
GAZİ ÜNİVERSİTESİ
Gazi
1926

NESNELERİN İNTERNETİ (IOT) PROTOKOLLERİNİN PERFORMANS KARŞILAŞTIRMASI

Yaseen NADİR

YÜKSEK LİSANS TEZİ
BİLİŞİM SİSTEMLERİ

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

EYLÜL 2021

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.

Yaseen NADİR

15/09/2021

NESNELERİN İNTERNETİ (IOT) PROTOKOLLERİNİN PERFORMANS KARŞILAŞTIRMASI

(Yüksek Lisans Tezi)

Yaseen NADİR

GAZİ ÜNİVERSİTESİ

BİLİŞİM ENSTİTÜSÜ

Eylül 2021

ÖZET

Nesnelerin İnterneti (IoT) kavramı topluma sunulmasından bu yana on yıldan fazla bir süre geçti. Bu araştırmada, Advance Messaging Queuing Protocol (AMQP), Message Queuing Telemetry Transport (MQTT), Constrained Applicatıon Protocol (COAP) olan üç IoT protokolünün performansını karşılaştırmayı hedeflemekteyiz. Seçilen protokollerin verimliliği, Verim ve Gidiş-Dönüş süresi (RTT) ile ilgili göstergeler kullanılarak değerlendirilmiştir. Üç protokolü değerlendirmek için python programlama dilinin SciPy kütüphanesini ve soket programlamayı kullanarak bir test ortamı önerilmiştir. Ele alınan senaryoda, bir IoT cihazı bir sunucuya veri gönderip ve yanıt beklmıştır. Paketler farklı boyutlar halinde gönderilmiştir. Oluşturulan veriler Pandas kütüphanesi kullanılarak toplanılmıştı ve CSV dosyalarına kaydedilmiştir. Ayrıca, verileri analiz etmek ve görselleştirmek için Anaconda kullanılmıştır. Deney testleri, buna göre farklı senaryolar için hangi protokolün en uygun olduğunu ortaya çıkarır. Sonuçlar, genel MQTT'nin diğer protokoller arasında en yüksek protokol verimliliğine ulaştığını göstermektedir.

Bilim Kodu : 11302
Anahtar Kelimeler : IoT, IoT Protokolleri, Performans, MQTT, AMQP, COAP
Sayfa Adedi : 49
Danışman : Doç. Dr. Mutlu Tahsin ÜSTÜNDAĞ

COMPARİSON OF IOT PROTOCOLS PERFORMANCE

(M. Sc. Thesis)

Yaseen NADIR

GAZİ UNIVERSITY

INFORMATICS INSTITUTE

September 2021

ABSTRACT

Internet of Things, it is been more than a decade since this concept was introduced to the society. In this research we have compared three IoT application protocols; Advanced Messaging Queuing Protocol (AMQP), Message Queuing Telemetry Transport (MQTT) and Constrained Application Protocol (CoAP). We have proposed a testbed using python programing language's library SciPy and socket programming to evaluate the three protocols. The selected protocols efficiency were evaluated using indicators related to Throughput and Round-Trip time (RTT). In the considered scenario an IoT device sends data to the server and waits for the response. The packets were sent in different sizes. The generated data were collected and saved in CSV files using Pandas library. Moreover, Anaconda was used to analyze and visualize the data. Experimentation tests reveal which protocol is best suited for different scenarios accordingly. Results show that overall MQTT achieves the highest protocols efficiency among other protocols.

Science Code : 11302
Key Words : IoT, IoT Protocols, Performance, MQTT, AMQP, COAP
Page Number : 49
Supervisor : Assoc. Prof. Mutlu Tahsin ÜSTÜNDAĞ

TEŞEKKÜR

Çalışmam boyunca tecrübe aktarımı, önerileri, kıymetli yardım ve katkılarıyla beni yönlendiren değerli Hocam ve Danışmanım Sayın Doç. Dr. Mutlu Tahsin ÜSTÜNDAĞ'a; her zaman yanımda olan ve desteğini esirgemeyen aileme ve en sevdiğim arkadaşım Areej Azim Uddin en içten teşekkürlerimi ve saygılarımı sunuyorum.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

RESİMLERİN LİSTESİ

Resim **Sayfa**

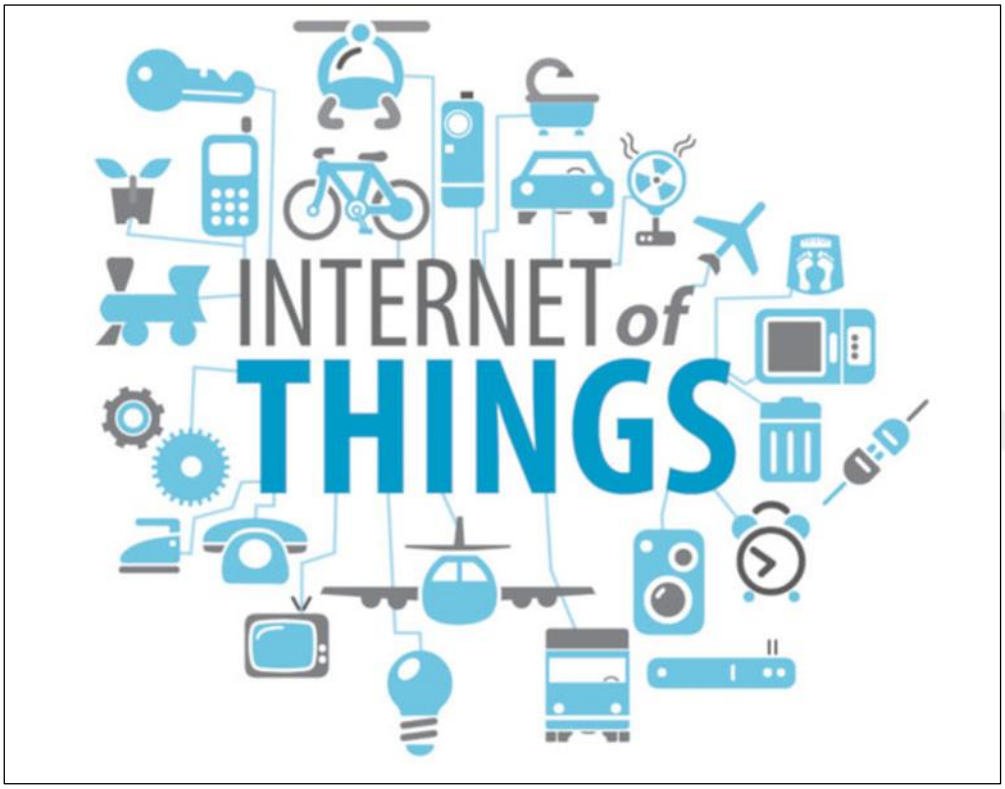
SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Kısaltma	Açıklama
AMQP	Gelişmiş Mesaj Kuyruklama Protokolü (Advanced Message Queuing Protocol)
CoAP	Kısıtlı Uygulama Protokolü (Constrained Application Protocol)
IETF	Internet Mühendisliği Görev Gücü (Internet Engineering Task Force)
IOT	Nesnelerin İnterneti (Internet of Things)
M2M	Makinadan Makinaya (Machine to Machine)
MQTT	Mesaj Kuyruklama Telemtri İletimi (Message Queuing Telemetry Transport)
NIC	Ağ Arayüz Kartı (Network Inteface Card)
QoS	Hizmet Kalitesi (Quality of Service)
REST	Temsili Durum İletimi (Representational State Transfer)
RFID	Radyo Frekans Tanımlayıcı (Radio Frequency Identification)
RTT	Gidiş Dönüş Süresi (Round-Trip Time)
SASL	Basit Kimlik Doğrulama ve Güvenlik katmanı (Simple Authentication and Security Layer)
TCP	İletim Kontrol Protokolü (Transmission Control Protocol)
TLS	Taşıma Katmanı Güvenliği (Transport Layer Security)
UDP	Kullanıcı Datagram Protokolü (User Datagram Protocol)
URI	Evrensel Kaynak Tanımlayıcı n(Universal Resource Identifier)

1. GİRİŞ

İnternet, bu dünya bir süredir var ama çoğunlukla insanların bir ürünü oldu, tüm veriler, görüntüler, kayıt, kitap oyunları ve ticaret insanlar tarafından insanlar için yaratıldı. İnternet şimdiye kadar icat ettiği en önemli ve dönüştürücü teknolojidir, hayatımıza dokunan ve dünyayı hızla değiştiren dijital kumaş gibidir. Benzer şekilde, yeni teknoloji ortaya çıkmıştır ve dünyayı yeniden değiştirmeye hazırdır, bu teknoloji insanları veya bilgisayarları bağlamakla ilgili değildir, "Nesnelerin İnterneti" nesnelerini bağlamakla ilgilidir.



Şekil 1.1. Nesnelerin interneti genel resmi
Kaynak: Wallen, 2016

Nesnelerin interneti, ağın günlük nesneleri kucaklayan gerçek dünyaya yayıldığı bir vizyonu temsil etmektedir. Fiziksel nesnelerin artık dijital dünya ile bağlantısı kesilmiyor, ancak bu nesneler uzaktan yönetilebilir ve Internet hizmetlerine fiziksel erişim noktaları olarak işlev görebilmektedir. Nesnelerin Ağında, nesneler, internete bağlanan ve veri üretmelerine, göndermelerine, almalarına ve değiştirmelerine olanak tanıyan şeyler algılama, birbirleriyle iletişim kurma ve birbirleriyle işbirliği yapma becerilerini ekleyerek deneyimi diğer nesnelerle paylaşmaya başlayabilmektedir. 1999- Neil Gross, Business Week ile ilgili olarak şunları söyledi: "Gelecek yüzyılda, dünya gezegeni elektronik bir cilde sahip olmayacak.

İnterneti hislerini desteklemek ve iletmek için bir iskele olarak kullanacak" (MaCabe, 2020). IoT, daha iyi bir yaşam kalitesi sağlamak için şimdiden önemli katkılarda bulunmuştur. Otoyollardan, yollardan, trafik ışıklarından dikiş makineleri, egzersiz bisikletleri, elektrikli diş fırçaları, çamaşır makineleri, elektrik sayaçları, fotokopi makineleri gibi cihazlara her şeyi bağlamak ayrıca vücutlarımızı saniyeler içinde doğrudan doktorlara veya sağlık kurumlarına bağlayan elektronik yamalar ve sensörler kullanarak birbirine bağlamkta bahsediyoruz. "Nesnelerin İnterneti" terimi, 1999 yılında şirketler arası bir RFID altyapısı tasarlamaya ve yaymaya başlayan Massachusetts Teknoloji Enstitüsü'ndeki (MIT) Auto-ID Center'ın çalışmasıyla popüler hale getirilmiştir. 2002 yılında kurucu ortağı ve eski başkanı Kevin Ashton Forbes Dergisi'nde "Şeyler için internete ihtiyacımız var, bilgisayarların gerçek dünyayı anlaması için standart bir yol" demiştir. Bu makale "Nesnelerin İnterneti" başlığını taşıyordu ve kelimenin tam anlamıyla belgelenmiş ilk kullanımıdır. Bununla birlikte, 1999'da zaten aynı düşünce, MIT Medya Laboratuarı'ndan Neil Gershenfeld tarafından popüler olan kıtapta "Şeyler Düşünmeye Başladığında" adlı aynı kavram kullanmış, "geriye dönük olarak, World Wide Web'in hızlı büyümesi, Net'i kullanmaya başladıkça sadece gerçek patlamayı başlatan tetikleyici yükü oldu" (Mattern ve Floerkemeier 2010).

IoT, internete bağlanabilecek algılama, yakalama, işleme ve iletişim yeteneklerine sahip binlerce son cihaz öngörmektedir. Düzenli olarak günlük hayatımıza katılan sensörlerin sayısı, örneğin akıllı telefonlarda, arabalarda ve binalarda sürekli olarak artmaktadır. Normalde, kaynak sınırlandırılmış araçlara, örneğin sensör göbeklerine veya akıllı telefonlara yerleştirilen bu sensörler, izledikleri veya birleştikleri maddelerin durumunu tahmin etmeyi sağlar. Bu verilerin İnternet aracılığıyla açık olması ihtimaline karşı, gerçek dünya nesnelerinin sanal portreleri olan Nesnelerin İnterneti'ne (IoT) eklerler. IoT cihazları, Internet'e bağlanmak için bir yerel alan ağı oluşturan, bir ağ geçidi üzerinden doğrudan bağlanabilir veya 2G / 3G / Uzun Süreli Evrim ve (5G) gibi hücresel teknolojiler kullanıyor olabilir. İkincisi, son cihazların genellikle Zigbee (IEEE 802.15.4 Standardına dayalı), Wi-Fi (IEEE 802.11 Standardına dayalı) gibi çeşitli radyo teknolojilerini kullanarak Makineden Makineye (M2M) ağları oluşturduğu durumdur, 6LowPAN (Zigbee) (Düşük Güçlü Kişisel Alan Ağları Üzerinden IPv6) veya Bluetooth (IEEE 802.15.1'e dayanarak) (Karagiannis, Chatzimisios, Vazquez-Gallego ve Alonso-Zarate, 2015). Bu araştırma IoT veri aktarımı için kullanılan üç ana Uygulama hizmetini inceleyecektir. Ayrıca, çoğunlukla IoT Verilerinin

aktarılması için kullanılan protokolleri analiz eder. IoT protokolleri arasında performans değerlendirmesi önerilen test yatağı kullanılarak yapılacaktır.

IoT'deki iletişim protokollerinin performansını değerlendirmek için birçok çalışma yapılmıştır. (Valera ve Tan, 2014), kesinlikle anlaşılan iki IoT protokolünün sunum sınavını yaptılar. Biri CoAP, diğeri MQTT. MQTT Yayın-Abone Olma modelidir ve CoAP (Constrained Application Protocol) iletişim için İstek Yanıt modeline bağlıdır, bu yüzden metodoloji tipik bir katman yazılımı kullandı. Bhattacharyya et al. (Bandyopadhyay ve Bhattacharyya, 2013) CoAp'ın istek cevap modeli ve MQTT yayın abone modeli ile bir karşılaştırma yapmıştır. IoT için başvuru katmanı kuralları, Kontogiannis, Chatzimparmpas ve Kokkonis (2015)'te araştırılmıştır; burada, yazarlar, sunulan trafik, paket kaybı olasılığı ve gecikme durumları üzerinde (CoAP) ve Message Queue Telemetry Transport (MQTT)'nın kantitatif bir incelemesini yapmıştır. Collina, Bartolucci, Coralli ve Corazza (2014). (Bellavista ve Zanni, 2016)'da yazarlar verimli IoT-bulut entegrasyonu için yenilikçi bir ölçeklenebilir dağıtık mimari önerdiler. Kullanılan iki protokol CoAP ve MQTT idi. 1000, 10000 ve 60000 mesaj iletimini tamamlamak için gereken zamanın performans ölçütü konulmuş ve yazarlara göre MQTT protokolü CoAP'tan çok daha hızlı olduğu bulgusuna erişmişlerdir.

Lavinia Nastasa, araştırmasında uygulama katmanı protokollerine odaklanmıştır; Güvenlik açısından CoAP, MQTT ve XMPP, üç protokolü ve bunların güvenlik açıklarını kısaca açıkladı ve çalışmaya göre, güvenlik ve işlevsellik açısından hiçbir çözüm için hiçbir tanesi en iyisi değildir (Nastasa, 2017). Alvin VALERA ve Hwee TAN, ortak uygulama programlama arayüzünü kullanarak ortak bir ara katman yazılımı önerdiler ve CoAP ve MQTT protokollerini test ettiler; farklı ağ koşulları farklı protokollerin performansını etkileyebilir, MQTT düşük paket kaybı için daha düşük gecikmeye sahiptir ve bunun tersi, CoAP masaj boyutu küçük olduğunda ve kayıp oranı eşit veya%25'ten az olduğunda güvenilir teslimat sağlamak için daha az trafik üretmektedir (Valera ve Tan 2014).

Bu araştırmada Advanced Message Queuing Protocol Protokolü, Message Queuing Transport Telemetry ve Constrained Application Protokolünün etkinliğini değerlendirmeyi amaçlıyoruz. Python programlama dilinin SciPy adlı kütüphanesi ve soket programlama kullanılarak bir test yatağı uygulaması geliştirilecektir. Bu test yatağı üç yapıdan oluşacaktır; IoT sensörleri, Ağ Geçidi ve Sunucu. Bu test yatağı üç yapıdan oluşacaktır; IoT sensörleri,

Ağ Geçidi ve Sunucu. Seçilen protokoller test ortamına uygulanacaktır, bu protokollerin etkinliği iki yönden değerlendirilecektir; Verim ve Gidiş-Dönüş Süresi (RTT). Ele alınan senaryoda, bir IoT cihazı bir sunucuya veri aktaracak ve yanıt bekleyecektir. Protokoller çeşitli trafik hacimleriyle test edilecektir. Sonuç olarak oluşturulan değerlendirme verileri Anaconda uygulaması kullanılarak analiz edilecek ve görselleştirilecektir.

Bu tez aşağıdaki gibi düzenlenmiştir: 2. Bölümde seçilen üç IoT protokolünün temel arka plan bilgilerini içeren bir kavramsal inceleme ve karşılaştırma sunmaktadir. 3. Bölümde bu çalışma yöntemi açıklanmakta, sonraki 4. Bölümde deneysel tasarımı kapsamaktadır, deney ortamını tasvir eder, takip edilen deneysel adımların bir açıklamasını sunar ve sonuçlar ve tartışma sağlar. Bölüm 5'te bu çalışma hakkında sonuçları ve gelcek çalışmalrı sunmaktadır.

2. UYGULAMA KATMANI PROTOKOLLERİ

Uygulama katmanı, kullanıcılar için hizmetler sağlar. Bu katmanın önemi, kullanıcıların ihtiyaçlarını karşılamak için yüksek kaliteli hizmetler sunma yeteneğine sahip olmasıdır. Uygulama katmanı protokolleri, mesajlaşma yeteneğini tanımlar ve hizmetin performansını etkiler.

2.1. Gelişmiş Mesaj Kuyruklama Protokolü (AMQP)

Gelişmiş Messaging Queuing Protokolü, mesaj odaklı ara yazılım için bir ikili ve açık standart uygulama katmanı protokolüdür. Eşler arası (noktadan noktaya yayınlama ve yayınlama ve abone olma) yönlendirme, mesaj yönlendirme, kuyruklama, İletim Kontrol Protokolü (TCP) gibi aktarım katmanı protokolüne altında yatan güvenilirlik, Basit Kimlik Doğrulama ve Güvenlik Katmanı (SASL) ve Aktarım Katmanı Güvenliği'ne (TLS) dayalı güvenlik sağlamak üzere tasarlanmıştır. Farklı satıcılar tarafından farklı dillerde yazılmış çok sayıda uygulama arasında birlikte çalışabilirlik sağlamaktadır. Gelişmiş Mesaj Kuyruklama Protokolü (AMQP) 2006 yılında çeşitli finans enstitüleri ve yazılım şirketleri tarafından yayınlanmıştır ve 2012 yılında OASIS1 tarafından standartlaştırılmıştşr Turowski, Bosse, Kubela ve Pohl (2018). Bu protokol iki seviyeye ayrılabilir; Mimari seviye ve Fonksiyonel seviye (Cui, 2017). Mimari düzeyde, AMQP iki katmana ayrılır: İşlevsel katman, uygulama adına harika işler yapan ve mesajlaşma yeteneklerini tanımlayan komutlar kümesini tanımlayan üst katmanıdır. Taşıma katmanı, uygulamadan sunucuya ve arkaya taşıyan alt katmanıdır, kanal çoğullama, çerçeveleme, içerik kodlaması, kalp atışı, veri gösterimi ve hata işleme bu katmanda yapılmaktadır.

Fonksiyonel düzeyde, AMQP'lerin ana bileşenleri istemciler ve sunuculardır. Sunucu şunlardan oluşur: değişim, mesaj kuyruğu ve ciltleme, bu üç bileşen kullanılarak mesaj yönlendirmesi gerçekleşmektedir. Değişim, yayıncıdan ileti alır ve uygun sıraya yönlendirmektedir. Mesaj Kuyruğu, iletileri, kastedilen tüketici istemcisi (abone) güvenli bir şekilde işleyene kadar saklamaktadır. Biding, borsalar ve kuyruklar arasındaki ilişkileri tanımlamaktadır.

2.1.1. Taşıma katmanı

AMQP aktarım belirtimi, AMQP ağında eşler arası mesaj aktarım protokolü sağlar ve yalnızca bir düğümden diğerine aktarımla ilgilidir. Düğümler arasında iletişim kurmaya başlamak için bir bağlantı kurulmalıdır. Bağlantı, iki tür uç noktaya sahip tek yönlü kanal sayısına bölünür: gelen ve giden uç noktalar, bu uç noktalar, gelen kanalları numarasına göre gelen çerçeveleri eşler ve giden çerçeveleri bitiş noktaları uygun giden kanal numarası ile işaretleyerek gönderilen çerçeveleri aktarmaktadır. Bununla birlikte, bir akran, akranın kapasitesini ve sınırlamasını tanımlayan AÇIN çerçevesini değiştirmeye başladığında başlayacaktır. Bir akran kapanma nedeni ile KAPATIN çerçevesini gönderdiğinde bağlantı kapanacaktır. Oturum adı verilen kanallar içindeki iletişim (Cui, 2017). Oturumlar, bağlantı iletişimi için bağlam görevi gören ilgili bağlantıların sayısını içeren iki düğüm arasındaki çift yönlü iletişimdir. Oturumlar, kullanılmayan kanal numarası ile atanan oturum bitiş noktası oluşturularak ve oturum bitiş noktasının giden kanalla ilişkisini bildiren bir BAŞLAYIN gönderilerek oluşturulur. Bir bağlantı kapatıldığında, kesildiğinde veya bir oturum bitiş noktası BİTİRİN çerçevesi göndererek sonlandırmaya karar verdiğinde oturum otomatik olarak sona ermektedir. Düğümler (kaynak ve hedefler) arasında mesaj iletmeye başlamak için düğümler arasında bir bağlantı kurulmalıdır. Bağlantı, iki düğüm arasındaki tek yönlü bir yoldur. İlgili bağlantıların grubu bir oturumla eklenebilir, ancak bir bağlantı aynı anda birden çok oturumla eklenmemelidir. Bağlantılar, kullanılmayan tanıtıcıya bağlı yerel bir terminal ile ilişkili bir bağlantı uç noktası oluşturularak ve yeni oluşturulan bağlantı yerel ve termin uç noktalarının durumunu körükleyen BAĞLAYIN çerçevesi gönderilerek, biri kaynak ve diğeri ise bağlantının yönüne bağlı olarak hedeflenmektedir. Bir akran, belirtilen bağlantı tutamacıyla AYRITIN çerçevesini göndererek bir bağlantıyı kapatır ve kapalı bayrağı doğru ayarlar, diğer taraftaki akran bağlantı uç noktasını yok eder ve kendi AYRITIN çerçevesiyle yanıt verir ve kapalı bayrağı doğru ayarlar.

Bağlantılar, bilgi ve protokol yöntemlerinin iletilmesinden sorumlu olan farklı çerçeve türlerine sırayla düzenlenen mesajlar ve bilgiler içerir. Bir bağlantıya herhangi bir çerçeve göndermeden önce, her düğüm protokol iletişiminin o iletişim için kullanılacak sürümünü belirlemek için protokol başlığını göndermeye başlamalıdır, 8 oktets dizisinden oluşan protokol başlığı 4 alana bölünmüştür, ilk alan “AMQP” nin büyük harfleri için ayrılan 4 sekizli, ardından 1 sekizli 0’lık protokol kimliğinden oluşur, ardından 3 imzasız bayt 1 majör için oktet, manor için 1 oktet ve sonra 1 oktet revizyon numarası protocol sürümü için oluşur.

Çizelge 2.1. Protokol başlığı düzeni

4 OKTETLER	1 OKTET	1 OKTET	1 OKTET	1 OKTET
“AMQP“	%d0	Majör	Küçük	Revizyon

Çerçeve 3 bölüme ayrılmıştır: Çerçeve başlığı, genişletilmiş başlık ve şöhret gövdesi; Çerçeve başlığı, başlığın başına gelmek üzere tasarlanmış 8 baytlık sabit boyutludur, boyut ve bilgi türü dahil olmak üzere çerçevenin geri kalanını analiz etmek için gerekli bilgileri içerir. Çerçeve başlıkları üç alandan oluşur: Boyut, DOFF ve türtür. Bayt 0-3 çerçeve boyutunu içerir, bu bölüm çerçeve başlığının toplam boyutunu, genişletilmiş başlığın ve çerçeve gövdesinini açıklamalıdır. DOFF, bayt 4, gövdenin çerçeve içindeki konumunu belirler. Tür, çerçeve başlığının bayt 5’i çerçevenin biçimini ve amacını belirtir ve geri kalan iki bayt kanal numarasını içerir. Genişletilmiş başlık, çerçeve başlığı ve çerçeve gövdesi arasındaki ortadaki değişken boyut alanıdır, muamele edilmek çerçeve türüne bağlıdır. Çerçeve gövdesi, çerçeve düzeninin son alanıdır, değişken boyuttadır ve çerçeve türünü derinleşir, gerçekleştirici ve ardından yük olarak tanımlar. Performatif, çerçeve gövde tipleriyle ve yükten kalan baytlarla tanımlanmalıdır.

Çizelge 2.2. AMQP çerçeve düzeni

<table>
<tr><th></th><th>0</th><th>1</th><th>2</th><th>3</th><th>Baytlar</th></tr>
<tr><td rowspan="2">Çerçeve Başlığı 8 bayt</td><td colspan="4">Boyut</td><td>0 Bayt</td></tr>
<tr><td>DOFF</td><td>Tip</td><td colspan="2">Kanal numarası</td><td>4 Baytlar</td></tr>
<tr><td>Genişletilmiş Başlık</td><td colspan="4">Tibe özgü (yoksayıldı)</td><td>8 Baytlar</td></tr>
<tr><td rowspan="2">Çerçeve Gövdesi</td><td colspan="4">Performatif</td><td rowspan="2">Baytlar DOFF*4</td></tr>
<tr><td colspan="4">Payload</td></tr>
</table>

Çerçevelerin dokuz gövde tipi vardır: Açın, başlayın, ekleyin, akış, aktarın, düzenleyin, ayırın, bitirin ve kapatın. Dokuz çeşit çerçeve sırayla kullanılacaktır: Bağlantıyı AÇIN, kanalda bir oturum BAŞLATIN, bu oturuma bir bağlantı BAĞLAYIN ve bağlantı durumunu güncellemek için FLOW’u kullanın, ardından bir mesaj göndermek için TRANSFER’i kullanmaktadır. DISPOSITION, uzak eşleri teslimat durumu değişikliklerini bildirmek için kullanılacaktır. Bir oturumdan bir bağlantıyı ayırmak için AYRITIN kullanılacaktır, benzer şekilde BİTİRİN bir oturumu sonlandırmak için kullanılır Son olarak KAPATIN bağlantıyı kapatır (Cui, 2017).

2.1.2. Fonksiyonel katmanı

Bu katman, Aktarım katmanının üzerinde oluşturulur. Fonksiyonel katmanda iki terim tanımlanmıştır: Çıplak Mesaj ve Açıklamalı Mesaj (OASIS, 2012). Çıplak mesaji gönderen tarafından sağlanan iletileri açıklar ve Açıklamalı mesaji alıcı düğümü tarafından alınan iletileri açıklar. Çıplak mesaji üç bölümden oluşur: Standart-özellikler, uygulama-özellikleri ve uygulama-verileri. Açıklamalı mesaj, baş açıklama bölümü, çıplak mesaj bölümü ve kuyruk açıklama bölümünden oluşur. Açıklamalı mesaj iki sınıfa ayrılır: hedefe giden mesajla birlikte giden ek açıklamadır ve bir sonraki düğüm tarafından tüketilen ek açıklamadır.

Çizelge 2.3. AMQP mesaj biçimi

			Çıplak Mesaji			
Başlık	Teslim-açıklama	Mesaj-açıklama	Özellikleri	Uygulama-Özellikleri	Uygulama-Verileri	Altbilgi
Açıklamalı Mesaj						

Çıplak mesaj sabittir, çıplak mesajlardaki bölümler herhangi bir ara düğüm tarafından değiştirilemez, silinmişlerse ara düğümler tarafından ilgilenilemez, ayrıca bu bölümler değiştirilemez. Bunun aksine, açıklamalı mesajdaki silinebilecek bölümler ve silinen bölümler varsayılan değerle dolu veya boş olarak kabul edilir. Başlık bölümü, mesajın ağ üzerinden aktarılmasıyla ilgili standart teslimat ayrıntılarını taşır, birkaç alan sağlar:

1- Dayanıklılık gereksinimini belirtir.

2- Mesaj önceliği, daha yüksek önceliğe sahip mesajlar, daha düşük önceliğe sahip mesajlardan önce teslim edilebilir.

3- TTL, milisaniye cinsinden yaşam süresi.

4- İlk edinen, eğer bu değer doğruysa, bu mesajın herhangi bir düğüm tarafından alınmadığı ya da tam tersi olduğu düşünülüyor.

5- Teslim sayısı, önceki başarısız teslimat teşebbüslerinin sayısı.

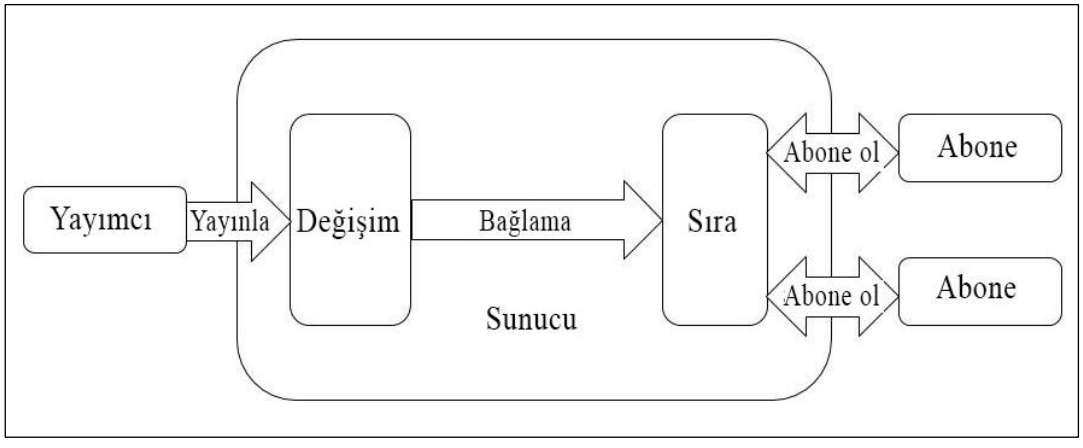
Teslim notu, standart dışı özelliklerin teslimi için kullanılır ve Mesaj notu, mesajın özellikleri için kullanılır, her iki not da gönderenden alıcıya bilgi iletir. Öncelikler, değiştirilemeyen bölüm standart öncelikler için kullanır, aşağıdaki alanlardan oluşuyordu:

- MESAJ KİMLİĞİ, genellikle ileti üreticisi tarafından ayarlanır ve benzersiz olmalıdır.
- KULLANICI KİMLİĞİ, mesaj üreticisinin kimliği.
- TO, hedefteki düğümü tanımlar.
- KONU, içerik ve amaç hakkında özet bilgisidir.
- CONTENT-TYPE, EXPIRATION-TIME, CREATIION-TIME.

Uygulama öncelikleri, yapılandırılmış uygulama verileri için kullanılan çıplak mesajın bir parçasıdır, bu veriler ara düğümler tarafından filtreleme için kullanılır. Uygulama verileri bölümü opak ikili veriler içerir. Altbilgi bölümü, yalnızca tüm çıplak mesaj görüldükten sonra hesaplanabilen mesaj teslimi hakkında ayrıntılar için kullanılır, ve ileti karmaları, HMAC'ler, imzalar ve şifreleme ayrıntıları sağlar.

2.1.3. Mesaj-Oryantasyon

AMQP, mesaj odaklı ara katman yazılımı (MOM) için tasarlanmış bir uygulama katmanı protokolüdür (Johansson, 2018). Mesajları yönlendirmek, kuyruklamak AMQP'nin ana tanımlayıcı özelliklerinden ikisidir. AMQP'nin toplam varlıkları aşağıdaki şemada gösterilecektir.



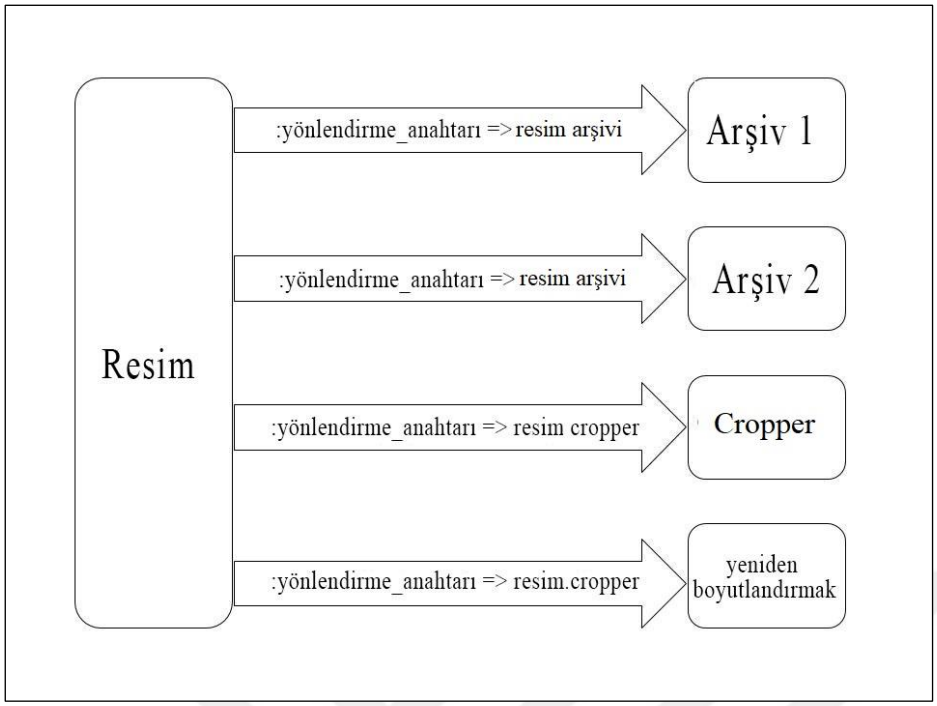
Şekil 2.1. AMQP mimarisi ve genel varlıkları

AMQP'deki ara katman yazılımı sunucusu veri üreticilerinden veri kabul eder ve iki ana görevi yerine getirir: Bunları farklı tüketicilere yönlendirir veya tüketici veri kabul edemediğinde bunları belleğe arabelleğe alır. Bu iki görev iki önemli bölüm tarafından yürütülmektedir: Değişim, üreticiden veri alır ve "Bağlamalar" adı verilen kuralları

kullanarak uygun mesaj kuyruğuna yönlendirir. Bağlama, borsalar, kuyruklar ve mesaj kuyruğu arasındaki ilişkileri tanımlar, mesajları saklar ve tüketiciyi istendiğinde mesajları çeker.

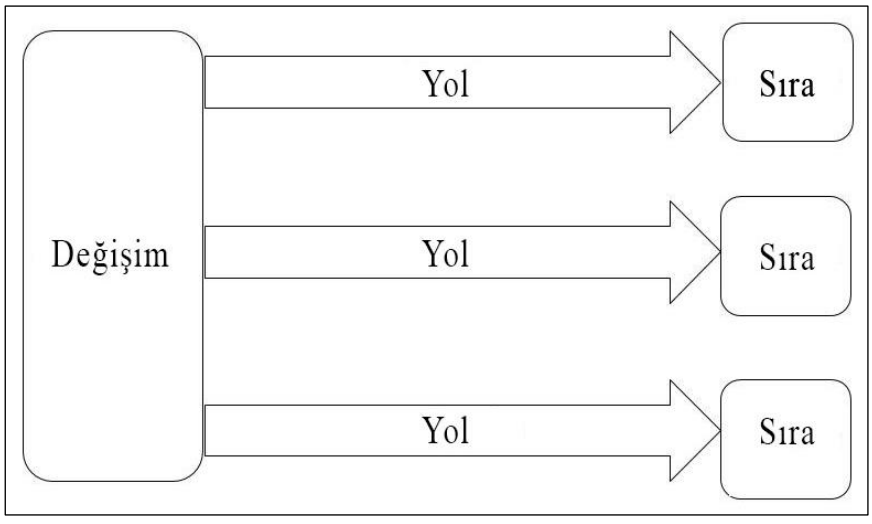
Mesajlar belirli özelliklerle üretilir, bazıları broker tarafından kullanılabilir ve geri kalanı broker için tamamen opaktır ve yalnızca hedefteki tüketiciler bunu kullanabilir. Ağlar güvenilir değildir, mesajlar işlenmeyebilir, bu nedenle AMQP'nin mesaj onayı vardır: Mesaj teslim edildikten sonra tüketici, brokere mesaj alındı bildirimlerini bildirir ve yalnızca bu durumda broker mesajları kuyruktan tamamen siler. Mesajları yönlendirememesi durumunda, broker bunu yayıncıya geri gönderir veya bırakır, yayıncı böyle bir vakanın nasıl ele alınacağına karar verir. Kuyruk, tüketici uygulaması tarafından tüketilecek iletileri tutan bir arabellektir, kuyruk, uygulamalar tarafından oluşturulabilir, paylaşılabilir, kullanılabilir ve yok edilebilir. Bağlama, değişim'ler ve kuyruklar arasındaki ilişkiyi tanımlar, değişim'ler mesajları sıralara yönlendirmek için kullanır. Bir kuyruğa bir değişim yolu mesajı yapmak için kuyruğun borsaya bağlı olması gerekir. Bir kuyruğa bir değişim yolu mesajı yapmak için kuyruğun değişimye bağlı olması gerekir. Bağlama isteğe bağlı yönlendirme anahtarına sahip olabilir, bazı değişim türleri yönlendirme anahtarını karşılaştırarak gelen mesajları uygun kuyruğa yönlendirir. Dahası, AMQP tüketicilere mesajları sıradan tüketmeleri için iki yol sunar: İlk "Push-API", belirli bir sıradaki mesajları tüketmek için ilgi uyaran uygulamalara gönderilen mesajı içerir. Bu yöntemdeki ikinci "Pull-API" tüketici uygulamaları mesajı gerektiği gibi getirir. Uygulama öncelikle daha esnek mesaj teslimi sağlayan gerekli varlıkları ve gerekli yönlendirme şemalarını tanımlar. Ancak, değişimler, bağlamalar, kuyruklar ve tüketiciler AMQP mimarisinde en önemli varlıklardır. Değişimler, üreticilerin mesajlarını alıp bir veya daha fazla kuyruğa ileten varlıklardır, ayrıca borsaların çeşitli nitelikler beyan etmesinin yanı sıra, en önemlisi: ad, dayanıklılık, otomatik olarak silme, argümandır. AMQP dört tür değişim sağlar: Doğrudan değişim, Fanout değişim, Konu değişim ve Üstbilgi değişim.

Doğrudan değişim, mesajların tek noktaya yayın yönlendirmesi olarak çalışır, bir kuyuruk, bağlayıcı anahtar K ile değiş tokuş yapmak için bağlanır, bir mesaj sadece R ve R = K yönlendirme anahtarıyla doğrudan değiş tokuşa ulaştığında, doğrudan değişim onu kuyruğa yönlendirecektir.



Şekil 2.2. Doğrudan değişim

En basit dağıtım türü olan Fanout değişim, yönlendirme anahtarını ve bağlayıcı anahtarı yok sayar ve mesajlarin yayın yönlendirmesini olarak çalışır, her zaman gelen mesajlarin kopyalarını değiş tokuşa bantlı tüm kuyruklara iletir.



Şekil 2.3. Fanout değişim

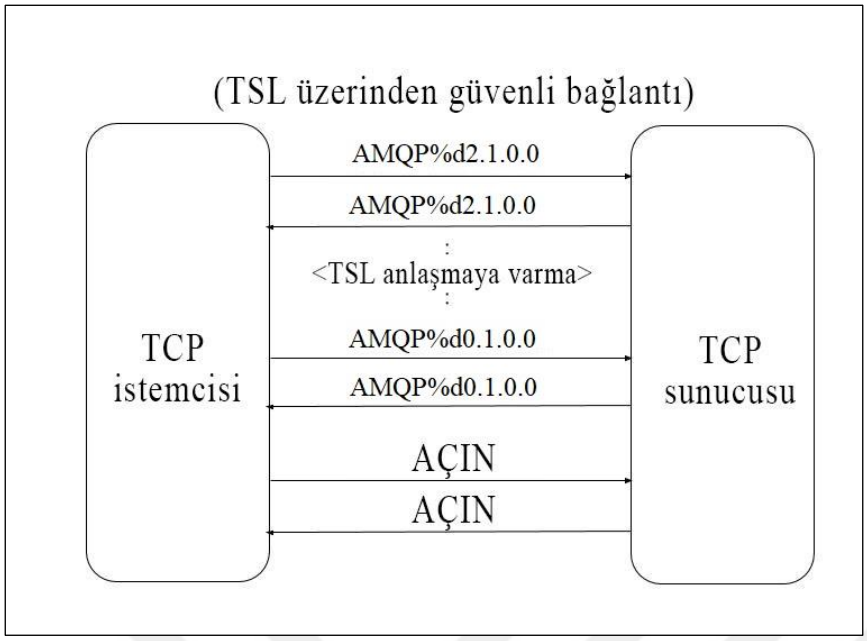
Konu değişim, çok noktaya yayın yönlendirme işlevi sağlar, anahtarları ve desenleri karşılaştırır, yönlendirme anahtarı bağlayıcı anahtarla eşleştiğinde ve yönlendirme modeli bağlayıcı desenle eşleştiğinde mesajlari bir veya çoklu kuyruğa iletir. Konu değişim türü

genellikle çeşitli yayınlama / abone olma modeli varyasyonlarını uygulamak için kullanılır (RabbitMQ, 2020).

Üstbilgi değişim, mesaji üstbilgisi olarak ifade edilen birden çok değere yönlendirme için tasarlanmıştır, üstbilgi değişim yönlendirme anahtarını tamamen yok sayar, üstbilgi değeri üstbilginin değeri bağlamanın değeriyle eşleşiyorsa mesaji ileti üstbilgisindeki özniteliklere göre yönlendirir. Bazen üstbilgi değişimler birden fazla mesaj üstbilgisi kullanarak kuyrukla bağlanabilir, böyle bir durumda "x-match" bağlayıcı argümanının kullanımı gelir, "x-match" iki moda sahiptir: x-match bağımsız değişkeni "all" olarak ayarlanmışsa değerler eşleşmelidir. Alternatif olarak, “x-match” u “any” olarak ayarlamak, sonra bir başlık değeri eşleştirmek yeterlidir.

2.1.4. Güvenlik

Bu bölümde AMQP tarafından sağlanan güvenlik yaklaşımları ele alınmaktadır. Güvenlik katmanı, AMQP’de AMQP ağı üzerinden yetkili, kimliği doğrulanmış ve şifrelenmiş iletişim kurmak için kullanılır. AMQP’de iki seçenek vardır: Aktarım katmanı Güvenliği (TLS) ve Basit Kimlik Doğrulama ve Güvenlik Katmanı SASLdır. Mesajlaşma istemcisi bağlantıları için TLS kullanımı, iletişimin mesajlaşma sunucusuna aktarılırken kurcalanmasına ve dinlenmesine karşı korunmasını ve sertifikalara dayalı kimlik doğrulamaları sağlar (Stack, 2020). TLS bağlantısı görüşmesi öncesinde, bir akran TLS iletişimi kurmak için protokol başlığını ortak akranına göndermeye başlamalıdır, bu protokol başlığı “AMQP” için 4 sekizli ve ardından 2 numaralı protokol kimliğinden oluşan 8 sekizli diziden oluşur, ardından Majör, Küçük ve Revizyon için 3 imzasız bayt gelir.

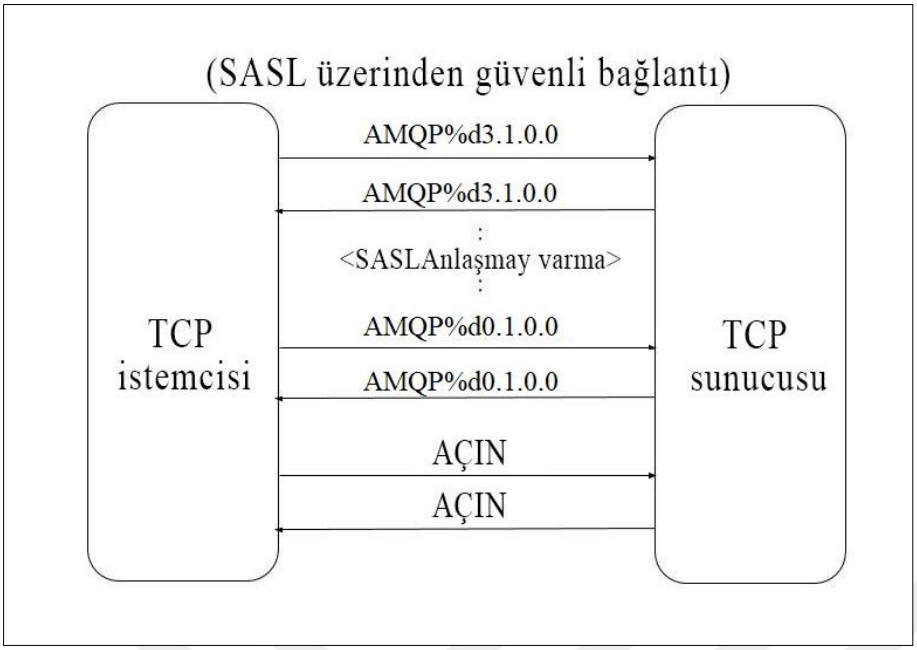


Şekil 2.4. AMQP'de TLS katmanı oluşturmak

TLS kullanımı görüşülürken birkaç kural uygulanmalıdır (OASIS, 2012):

- TCP istemcisi ve sunucu eşleri sırasıyla TLS istemcisi ve sunucu eşlerini ayarlamalıdır.
- TLS istemci eşi sunucu adı gösterge uzantısını KULLANMALIDIR.
- TLS istemcisi sertifikaların sunucu tarafından sunulduğunu doğrulamalıdır.
- Uygulamanın ortak eşinin yanıtını beklemeden bağlantıyı kapatmayı seçmesi mümkündür, bu kapanmaya tek yönlü kapatma denir.

AMQP'de kimlik doğrulama ve veri güvenliği için Basit Kimlik Doğrulama ve Güvenlik Katmanı (SASL) kullanılır, bu mekanizma kimlik doğrulama güvenliğinin artmasına izin veren basit kullanıcı adı ve parolanın ötesinde kimlik doğrulama sağlar. SASL katmanı oluşturmak için, her iki eşin de sırasıyla 8 oktetten oluşan ve "AMQP" için 4 oktet ve ardından 3 protokol kimliğinden oluşan 8 oktettir, ardından Majör, Küçük ve Revizyon için 3 imzasız bayt gelir.



Şekil 2.5. AMQP'de SASL katmanının oluşturulması

SASL iletişimi kurarken bazı kurallar dikkate alınmalıdır (OASIS, 2012):

- SASL çerçevesinde, çerçeve başlığının bayt 6 ve 7'si yok sayılır, böylece uygulama bunları 0 00 olarak ayarlamalı ve genişletilmiş DOFF başlığı 0 * 02 olarak ayarlamalıdır.
- Çerçeve gövdesi tam olarak bir AMQP tipine sahip olmalıdır.
- SASL anlaşmasında, SASL sunucularının desteklenen kimlik doğrulama mekanizmalarını duyurması gerekir.
- İletişim ortağı eş, desteklenen bir kimlik doğrulaması seçer ve SASL değişimini başlatır.

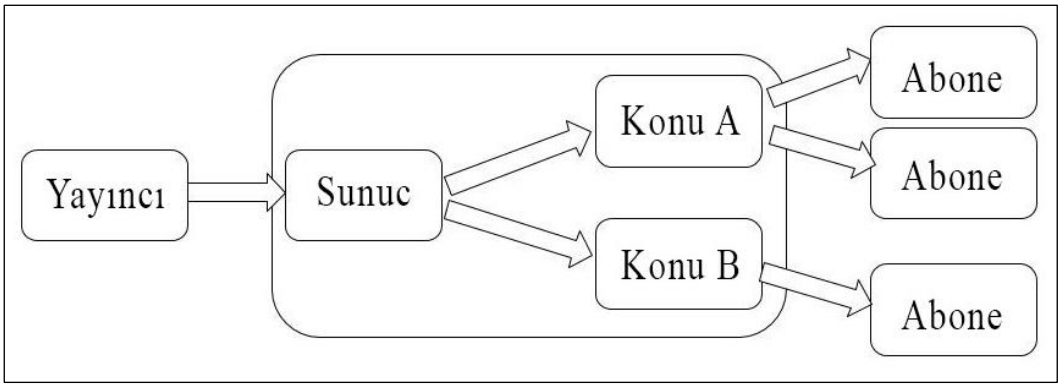
2.2. Mesaj Kuyruklama Telemtri İletimi (MQTT)

MQTT mesaj odaklı bir protokoldür , istemci sunucu yayınlama-abone olma ileti aktarım protokoldür. Son derece hafif, açık, basit, yüksek gecikme veya güvenilir olmayan ağ için uygulanması kolay olacak şekilde tasarlanmıştır. (Konstantinos ve diğerleri, 2016). MQTT protokolü, katıştırılmış ağ ve aygıtlar için düzenli, kayıpsız, çift yönlü iletişim sağlayan TCP/IP ağları üzerinden çalışmaktadir. MQTT, 1999 yılında Andy Stanford-Clark ve Arlen Nipper adlı iki mühendis tarafından IBM tarafından oluşturuldu ve 2013'te IBM, MQTT 3.11 sürümünü OASIS'e standartlaştırma için gönderilmişti (Janakiram, 2016).

MQTT'nin özetlenebileceği birkaç özelliği vardır:

- Bir-çok mesaj yönlendirmesi sağlayabilen yayınla-abone ol mesajlaşma modelini kullanmaktadır.
- MQTT uygulamaların ayrıştırılmasını sağlamaktadır.
- İleti teslimi için üç hizmet kalitesi (QoS).
- Yükün içeriğine uymayan mesaj aktarımı sağlamaktadır.
- Ağ trafiğini azaltmak için küçük bir taşıma yükü ve protokol alışverişi en aza indirilmiştir.
- Öngörülemeyen bağlantı kesilmesi olduğunda ilgili tarafları bilgilendirmek için bir mekanizmadır.

MQTT'nin yayımlama-abone olma modeli, İstemci-Sunucu modelini temel alır ve sunucu daha çok bir aracı veya ağ geçidine benzer, MQTT Aracısı, yayımlayan istemci ile abonelik yapan istemci arasında aracı görevi gören bir aygıttır. İstemcinin yayınladığı iletileri ilgili abonelere iletmek ve iletmekten sorumludur. Mesajlar belirli "konular" ile yayınlanır ve her müşteri çeşitli konulara abone olabilir. Aracı konuları saklar ve istemcilerin abone olma / abonelikten çıkma isteklerini işler. Müşteriler, abonelik isteklerini bir veya daha fazla konu içeren bir konu filtresiyle gönderir ve her konunun etiketi olarak bir konu adı bulunmaktadır. Tüm etkileşimler eşzamansızdır ve istemciler yalnızca sunucu ile doğrudan iletişim kurmaktadır.



Şekil 2.6. MQTT mimarisi

2.2.1. MQTT mesajı biçimi ve mesajı modeli

Çizelge 2.4. MQTT paket biçimi

Bitler	7	6	5	4	3	2	1	0
Bayt 1	Mesaj Tür				DUP	QoS		Kalan
Bayt 2	Kalan Uzunluk							
Bayt 3	Değişken Başlığı							
Bayt n								
Bayt n+1	Taşıma kapasitesi							
Bayt m								

MQTT, bir dizi kontrol paketi değiştirerek çalışmaktadır. Yukarıdaki çizelge MQTT mesaj biçimini açıklığa kavuşturmaktadır. Her MQTT kontrol paketi formatının 4 alana ayrılmış sabit bir başlığı vardır: Mesaj Türü, DUP, QoS ve Tutmak.

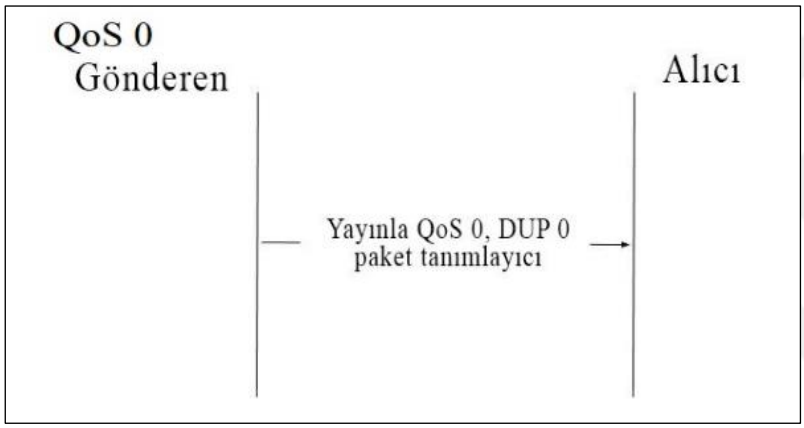
MQTT denetim paketi biçim üstbilgisinden Bayt 1'in 4-7 bitleri, mesaji türünü bildirmek için kullanılır, bu türleri kullanarak MQTT tüm mesaji yeteneklerini gerçekleştirebilir. Aşağıdaki çizelgede mesaj türleri gösterilmektedir.

Çizelge 2.5. MQTT mesaj türü

Tip	Değer	Akış Yönü	Açıklama
Reserved	0	Yasak	Ayrılmış
CONNECT	1	İstemciden Sunucuya	İstemcinin Sunucuya bağlanma isteği
CONNACK	2	Sunucudan İstemciye	Bağlan Onayı
PUBLISH	3	Çift yönlü	Mesajı yayınla
PUBACK	4	Çift yönlü	Yayınlama Bildirimi
PUBREC	5	Çift yönlü	Yayın alındı
PUBREL	6	Çift yönlü	Yayınlandı
PUBCOMP	7	Çift yönlü	Yayınlama Tamamlandı
SUBSCRIBE	8	İstemciden Sunucuya	İstemci abone isteği
SUBACK	9	Sunucudan İstemciye	Abone Kabulü
UNSUBSCRIBE	10	İstemciden Sunucuya	Aboneliği iptal etme isteği
UNSUBACK	11	Sunucudan İstemciye	Aboneliği iptal etme
PINGREQ	12	İstemciden Sunucuya	PING isteği
PINGRESP	13	Sunucudan İstemciye	PING yanıtı
DISCONNECT	14	İstemciden Sunucuya	İstemcinin bağlantısı kesilmek
Reserved	15	Yasak	Ayrılmış

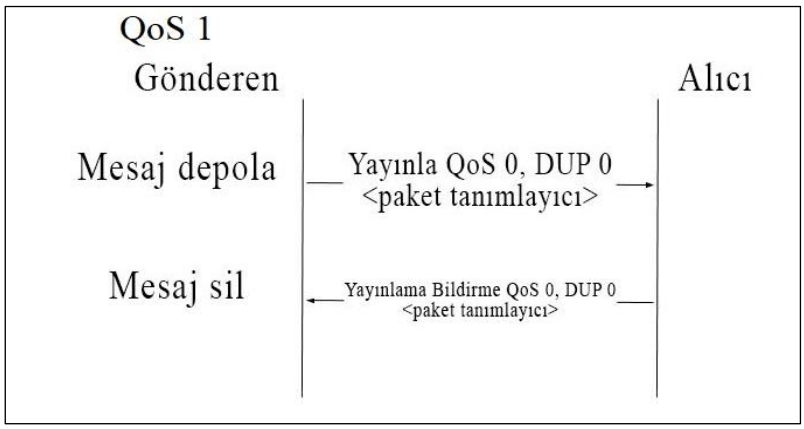
Bir sonraki alan DUP içeren sabit üstbilgide 3 bayt 1 bitidir, YAYIN ileti türünün dağıtımını çoğalmaktadır. Bir istemci veya sunucular MQTT YAYINLA paketi göndermeye çalıştığında DUP alanı 0 veya yyınla paketleri yeniden göndermeye çalıştığında istemci veya sunucu tarafından 1 olarak ayarlanmalıdır. YAYINLA paketindeki aşağıdaki alan QoS içeren sabit üstbilgide 1-2 bayt 1 bitidir, görevi mesajların teslim edilmesini sağlamaktır. MQTT, üç QoS seviyesine sahip uygulama mesajları sunar:

- QoS 0: Mesajın iletildiği ancak alıcı tarafından yanıt gönderilmediği ve gönderen tarafından gönderilme mesajı gönderilmediği "en fazla bir", kayıp meydana gelebilir.



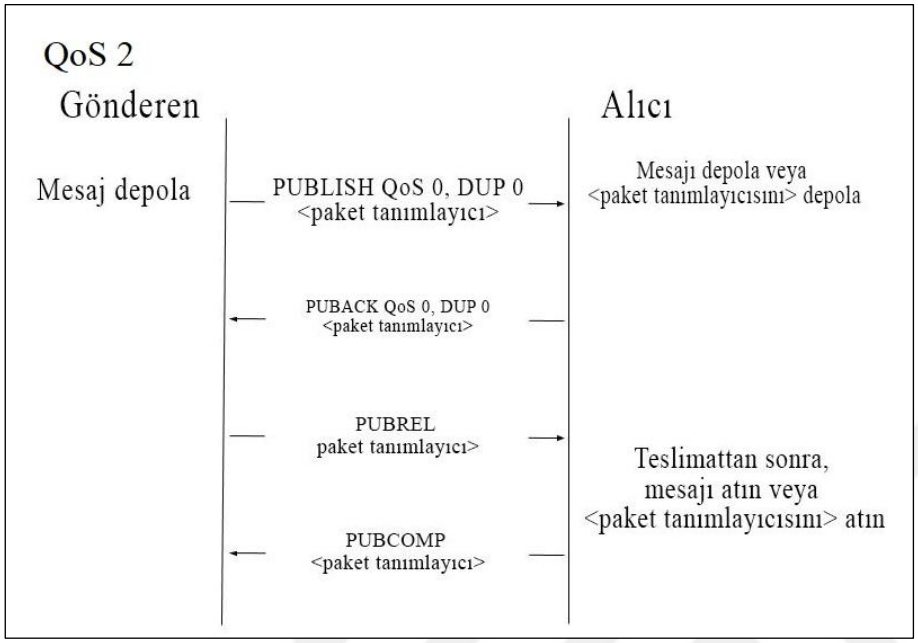
Şekil 2.7. QoS 0 mesajlaşma örneği

- QoS 1: "En az bir kez", bu QoS mesajın alıcıya en az bir kez gönderilmesini sağlar, çoğaltılabilir.



Şekil 2.8. QoS 1 mesajlaşma örneği

- QoS 2: “Tam olarak bir kez teslimat”, mesajın alıcıya kayıp veya çoğaltma olmadan gelmesi garanti edilen yerdir, en yüksek QoS seviyesidir.



Şekil 2.9. QoS 2 mesajlaşma örneği

Retain alanı, sabit başlıktan bayt 1’in 0 bitinde bulunan bir sonraki alandır. İstemciden sunucuya gönderilen PUBLISH paketinde alıkoyma işareti 1 olarak ayarlanır, bu durumda hizmetin iletileri depolaması gerekir ve abonelikleri konu adıyla eşleşen istemcilere teslim edilmesi için QoS olur. İstemcilerden sunucuya gönderilen PUBLISH paketinde flag bayrağı 0 olarak ayarlandıysa, sunucu ileti deposunu saklamamalı ve değiştirilmemeli, tutulan iletileri silmemelidir. Kalan Uzunluk bayt 2 ile başlar ve sabit üstbilgiden kalan bayttır, değişken üstbilgideki ve yükteki verileri içerir.

Değişken başlıklar sabit üstbilgi ve yük arasında konumlandırılır, bazı MQTT kontrol paketlerinin değişken bileşenleri vardır, değişken başlığın içeriği MQTT kontrol paketlerine bağlı olarak farklıdır. Paket Tanımlayıcısı alanı, bazı kontrol paketleri için değişken üstbilgide yaygındır. Aşağıdaki çizelgede, paket tanımlayıcı içeren paket türleri gösterilmektedir.

Çizelge 2.6. Denetim paketi türlerinde paket tanımlama alanı

Kontrol Paketi Tipi	Paket tanımlayıcı alanı
CONNECT	HAYIR
CONNACK	HAYIR
PUBLISH	EVET (QoS> 0 ise)
PUBACK	EVET
PUBREC	EVET
PUBREL	EVET
PUBCOMP	EVET
SUBSCRIBE	EVET
SUBACK	EVET
UNSUBSCRIBE	EVET
UNSUBACK	EVET
PINGREQ	HAYIR
PINGRESP	HAYIR
DISCONNECT	HAYIR

İstemci ve sunucu paket tanımlayıcısını uygunsuz bir şekilde ayarladı, ancak aynı paket tanımlayıcılarını kullanarak mesaj işlemlerine katılabilir. Bazı kontrol paketi türleri, paketlerinin son alanında yük gerektirir. Aşağıdaki çizelge bu kontrol paketi türlerini göstermektedir.

Çizelge 2.7. Yük gerektiren kontrol paketleri

Kontrol Paketi Tipi	Yük
CONNECT	Gereklidir
CONNACK	Yok
PUBLISH	İsteğe bağlıdır
PUBACK	Yok
PUBREC	Yok
PUBREL	Yok
PUBCOMP	Yok
SUBSCRIBE	Gereklidir
SUBACK	Gereklidir
UNSUBSCRIBE	Gereklidir
UNSUBACK	Yok
PINGREQ	Yok
PINGRESP	Yok
DISCONNECT	Yok

2.2.2. Güvenlik

Önemli olan, MQTT özelliklerinde güvenlik özelliği bulunmaması, her uygulamanın güvenliğinin tasarımına göre yapılmasıdır. MQTT protokolü TCP / IP'ye dayandığından ve uygun güvenlik ve bütünlük sağlamak uygulama sorumluluğu olduğundan, güvenlik özellikleri MQTT'nin üzerine uygulanmalıdır. Bu, Aktarım Katmanı Güvenliği TLS / Güvenli Yuva Katmanı SSL kullanılarak gerçekleştirilebilir. Kimlik doğrulama sertifikalara, gizlilik ise uygulamanın şifreleme mekanizmasına dayanır. İstemcinin sunucuya göre kimlik doğrulaması yapan MQTT, kullanıcı adı ve parolayı şifrelemek için TLS / SSL şifrelemesini kullanır; sunucu, istemci tarafından gönderilen SSL sertifikalarını kullanarak istemcinin kimliğini doğrulayabilir. Sunucunun istemci tarafından doğrulanması, istemci sunucu sertifikasını doğrulayabilir. Uygulama mesajlarının gizliliği, bazı TLS şifreleme paketleri, verileri şifrelemeyen NULL şifreleme algoritması içerir; bunun yerine, uygulamaların ağ üzerinden uygulama mesajının gizliliğini sağlamak için mesajının içeriğini şifrelemesi gerekir.

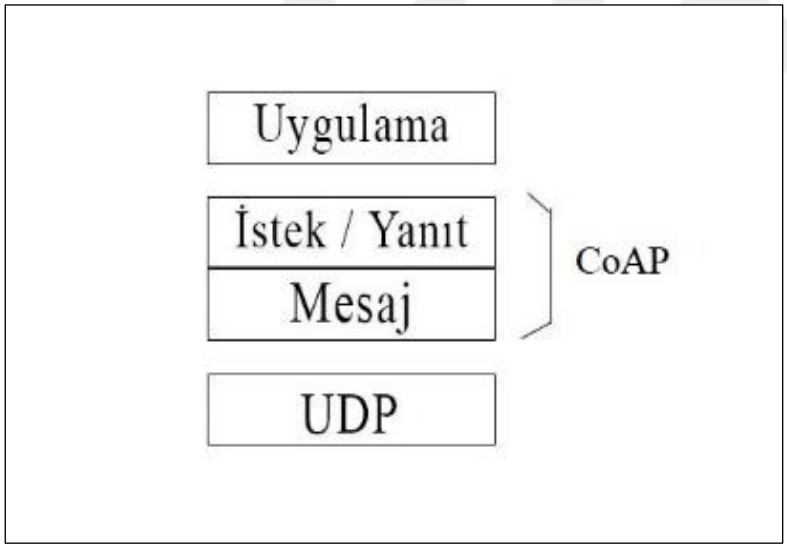
2.3. Kısıtlı Uygulama Protokolü (CoAP)

Kısıtlı Uygulama Katmanı, İnternet Mühendisliği Görev Gücü (IETF) tarafından oluşturulmuş ve standartlaştırılmıştır. CoAP, Temsili Durum Transferi (REST) temelli kısıtlı düğümler ve ağlar için özel bir internet uygulama protokolüdür, (Shelby, Hartke ve Bormann, 2014).

CoAP RESTful bir uygulama protokolüdür, ancak CoAP'ı net bir şekilde anlayabilmek için önce REST konseptinin tanıtılması gerekir. Representational State Transfer (REST), 2000 yılında doktora tezinde Roy Fielding tarafından tanımlanan bir web mimarisi tarzıdır, HTTP tabanlıdır ve web'deki bilgisayar sistemleri arasında iletişim standartları sağlayarak, sistemlerin birbirleriyle iletişim kurmasını kolaylaştırır, (Codecademy, 2020). RESTful sistemleri, vatansız ve istemci ve sunucuların ayrı kaygıları olarak sınıflandırılır. REST mimarisinde istemci içinde depolanan kaynaklara ulaşmak için sunucuya istek gönderir, bu her kaynağın adres olarak Tekdüzen Kaynak Tanımlayıcısı'na (URI) sahiptir. Bir istek genellikle aşağıdakilerden oluşur:

- Temel olarak 4 yöntem GET, PUT, POST ve DELETE olan bir HTTP yöntemi, her biri gerçekleştirilecek farklı işlem türlerini tanımlar.
- İstemcinin, sunucudan alabilecek içerik türünü göndererek istekle ilgili bilgileri iletmesine olanak tanıyan bir başlık.
- Kaynağın adresi: http://example.com/resources/URI.
- Veri içeren isteğe bağlı bir mesaj gövdesi.

Sunucu yanıtının olduğu durumda, içerik türü yanıtın başlığına eklenmelidir. Bu içerik türü, istemcinin istekte belirttiği seçeneklerden biri olmalıdır. Bu nedenle, tarayıcıda görüntülenen gösterim, referans alınan URI kaynağının mevcut durumunun yakalanmasıdır. REST'in aksine CoAP, tasarımı basit tutmak için UDP katmanı gibi datagram tabanlı taşımayı kullanır ve birincil hedefi sınırlı kaynaklara sahip cihazlar veya düşük bant genişliğine sahip ağlar arasında iletişim sağlamaktır, (Shelby ve diğerleri, 2014).

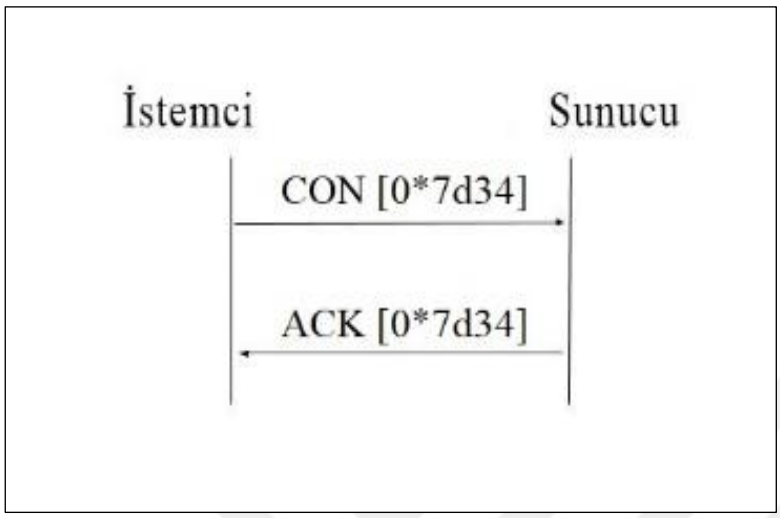


Şekil 2.10. CoAP katmanlaması

2.3.1. Mesaj modeli

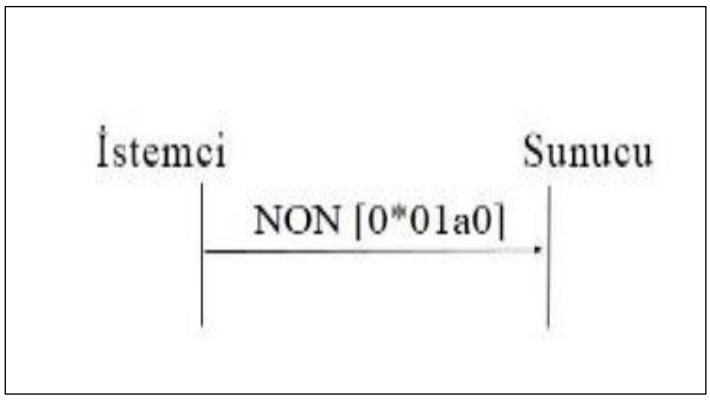
CoAP, REST'e benzer GET, PUT, POST ve DELETE yöntemlerini kullanır Shelby ve diğerleri (2014). CoAP'ın mesajlaşma modeli, dört mesaj alışverişi türüne dayanmaktadır: Onaylanabilir mesajı (CON), Onaylanmayan mesajı (NON), Onay(ACK) ve Sıfırla (RST). Bu dört mesaj, istek ve yanıt arasında paylaşılır. Her iletiler, başlıktaki İleti Kimliği kullanılarak çoğaltma algılaması sağlar. Onaylanabilir mesaj (CON) güvenilirlik sağlar,

gönderen alıcıdan aynı Mesaj Kimliği ile Onay (ACK) alana kadar mesaj tekrar yeniden iletilir. Alıcının CON mesajını işleyememesi durumunda Sıfırla (RST) mesajı ile cevap verecektir.



Şekil 2.11. Onaylanabilir mesajı

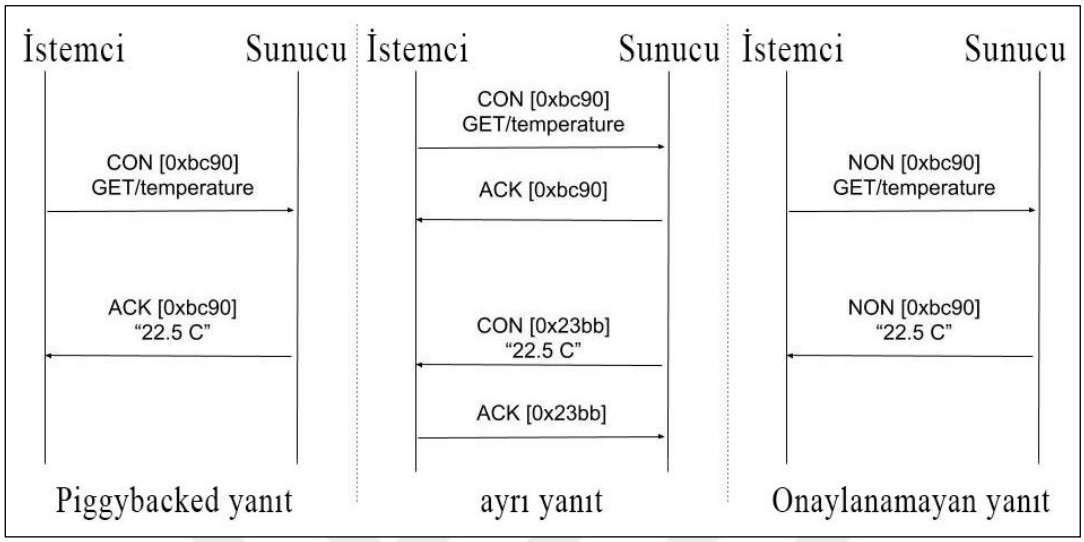
Güvenilir olması gerekmediğinde istek Onaylanmayan mesaj (NON) olarak gönderilebilir. Onay mesajı gerektirmez, ancak yine de çoğaltma tespiti için mesaj kimliği içerir. Alıcı, Onaylanamayan mesajı işleyemediğinde, Sıfırlama mesajı (RST) ile yanıt verebilir.



Şekil 2.12. Onaylanmayan mesajı

CoAP üç tür yanıt sağlar: Piggybackli yanıtı, istek Onaylanabilir ise ve kaynak hemen kullanılabilirse kullanılır, Onay mesajının içinde taşınır. Ayrı yanıt, sunucu hemen yanıt veremediğinde kullanılır, gönderene boş bir bildirim göndererek isteği yeniden iletmez. Kaynak hazır olduğunda, sunucu kaynağa yeni bir onay gönderir. Ayrıca, Onaylanamayan

mesajında bir istek gönderilmesi durumunda, yanıt Onaylanamayan yanıt olacaktır, cevap teyit edilemeyen mesajda yapılacaktır.



Şekil 2.13. CoAP isteği ve yanıtı

CoAP, önbellek ve proxy gibi bazı işlevleri destekler; Bitiş noktaları ve aracılar yaklaşan isteklere yanıt vermek için yanıtları önbelleğe alabilir, ait ağ bant genişliğini ve yanıt süresini azaltmayı amaçlamaktadır. Önbelleklemenin iki mekanizması vardır: Tazelik mekanizması, bu mekanizmanın amacı, gecikmeyi ve ağ gidiş-dönüşlerini azaltmak için isteklere gerek kalmadan saklanan yanıtları yeniden kullanmaktır. Yeni bir istek gerekli olduğunda, bu isteklere yanıt vermek için önbelleğe alınan yanıtları yeniden kullanan Doğrulama mekanizması kullanılabilir ve ağ bant genişliği kullanımını azaltır. CoAP önbelleği tek seferde HTTP yanıt koduna bağlıdır.

CoAP UDP üzerinden çalıştığından, çok noktaya yayın IP hedef adreslerinin kullanımını da destekleyerek çok noktaya yayın CoAP isteklerini etkinleştirir.

2.3.2. CoAP mesaj biçimi

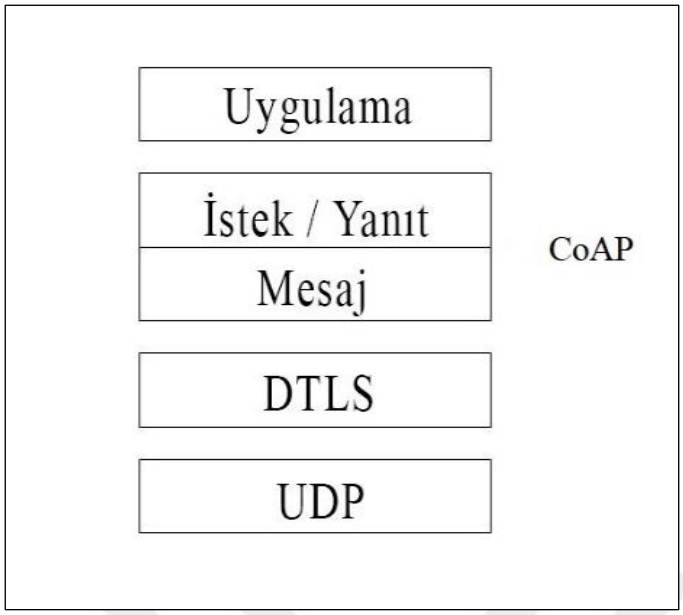
Çizelge 2.8. CoAP mesaj biçimi

Baytlar	0								1								2								3							
Bitler	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
Alanlar	versiyon		Tip		TKL				Kod								Mesaj kimliği															
	Token (varsa, TKL baytları)																															
	Seçenek																															
	1	1	1	1	1	1	1	1	Payload																							

CoAP ileti biçimi İkili biçimde, (kodlanır Shelby ve diğerleri, 2014).CoAP mesaj biçimi, beş alandan oluşan dört sabit boyutlu 4 bayt başlığıyla başlar. İlk alan, CoAP sürüm numarasını gösteren iki bit sürüm alanıdır. Sonrakı iki bit, mesajın türünü gösteren tür alanıdır: ONAYLANABİLİR MESAJI (0), ONAYLANMAYAN MESAJI (1), ONAY (2), SIFIRLAMA (3). Sonrakı, Toten alanının uzunluğunu gösteren dört bittir. Dördüncüsü, istek yöntemlerini ve yanıt kodunu gösteren kod alanıdır. Beşincisi, mesaj kopyalarını ve eşleşen mesaj türlerini tespit etmek için kullanılan on altı bit mesaj kimliği alanıdır. Başlığı, istek ve yanıtları ilişkilendirmek için kullanılan TKL ile gösterilen dört baytlık alan Simgesi alanı izler. Aşağıdaki alan isteğe bağlı seçenek alanı ve ardından isteğe bağlı faydalı yüktür.

2.3.3. Güvenlik

CoAP UDP üzerinden çalıştığından Datagram Aktarım Katmanı Güvenliği kullanılarak sabitlenir. Datagram Aktarım Katmanı Güvenliği (DTLS), veri kaydı tabanlı uygulamalar için gizlice dinleme, kurcalama veya ileti sahteciliğini önleyecek şekilde iletişim kurmalarını sağlayarak güvenlik sağlayan bir iletişim protokolüdür. DTLS protokolü, akış yönelimli Aktarım Katmanı Güvenliği (TLS) protokolünü temel alır ve benzer güvenlik garantileri sağlamayı amaçlamaktadır (Wikipedia, 2020).



Şekil 2.14. DTLS güvenli CoAP'ın soyut katmanlaması

CoAP, cihazları için dört güvenlik modu tanımlar:

- NoSec: Bu modda DTLS devre dışı bırakıldığında, uygun olduğunda alternatif düşük katmanlı güvenlik tekniği kullanılmalıdır.
- Ön Paylaşımlı Anahtar: DTLS etkinleştirilir, düğümler arasındaki bağlantı Ön Paylaşımlı anahtar kullanılarak şifrelenir ve uygulamanın TLS_PSK_WITH_AES_128_CCM_8 şifre paketini desteklemesi gerekir.
- Ham Ortak Anahtar: Bu modda DTLS etkinleştirildiğinde, bağlantı çift asimetrik anahtar kullanılarak şifrelenir, ancak X.509 sertifikası olmadan uyarı, TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 şifre paketini desteklemelidir.
- Sertifika: DTLS etkinleştirilir, bağlantı, konuya bağlanan ve bazı ortak güven kökleri tarafından imzalanan X.509 sertifikasına sahip asimetrik bir anahtar çifti kullanılarak şifrelenir. Gösterim, TLS_ECDHE_ECDSA_WITH_AES_128_CCM_8 şifre paketini desteklemelidir.

2.4. Protokollerin Karşılaştırılması

Çizelge 2.9. Protokollerin karşılaştırılması

	MQTT	AMQP	CoAP
Tanım	Yayınlama / Abonelik	Noktadan Noktaya ya da Yayınlama / Abonelik	İstek/Cevap
Önerilen	1999	2003	2013
Standardize	2013	2011	2013
Standartlar	OASIS MQTT Standartları	OASIS AMQP	IETF COAP Standartları
Uygulama Mimarisi	Sunucu	Sunucu	İstemci-Sunucu
İletişim Protokolü	TCP	TCP	UDP
Hizmet Kalitesi	3	3	Onaylı veya Onaysız mesaj seçeneği
Güvenlik / Şifreleme Yöntemleri	Basit- Kullanıcı Adı/Şifre modeli, Veri Şifreleme için SSL desteği	Oğrulama, Veri şifreleme için TLS desteği	DTLS

3. YÖNTEM

IoT uygulama katmanı protokollerine ilişkin yayınlanmış birkaç anket belgesi vardır. Ancak IoT teknolojisi muazzam bir hızla büyüdükçe, bu uygulama protokollerini farklı ölçütler ile test etmek önemliydi. Bu araştırmada da bu konuyla ilişkili olarak iletişimde kullanılan MQTT, AMQP ve COAP gibi protokoller incelenerek, farklılıkları ve performansları karşılaştırılmıştır. Seçilen protokollerin verimliliği, Verim ve Gidiş-Dönüş süresi (RTT) ile ilgili göstergeler kullanılarak değerlendirilmiştir.

Bu araştırma kapsamında, seçilen üç protokolün etkinliğini değerlendirmek amacıyla test yatağı yazılımı geliştirilmiştir. Test yatağı yazılımı Python programlama dilinin SciPy adlı kütüphanesi ve soket programlama kullanılarak geliştirilmiştir. Bu test yatağı üç yapıdan oluşmaktadır; IoT sensörleri, Ağ Geçidi ve Sunucu. Seçilen protokoller test ortamına uygulandı, bu protokollerin etkinliği iki metrik altında değerlendirilmiştir; Verim ve Gidiş-Dönüş Süresi (RTT). Ele alınan senaryoda, IoT cihazı sunucuya paketler gönderir ve yanıt bekler. Protokoller çeşitli trafik değerleriyle test edilmişti. Sonuçta oluşturulan değerlendirme verileri Pandas Library kullanılarak CSV dosyaları olarak saklanmıştı. Bu veriler Anaconda kullanılarak analiz edilmiş. Bu deneysel araştırma amacı, farklı ağ senaryoları için IoT protokollerinin performansını değerlendirmek ve karşılaştırmaktıdır. Ayrıca “Protokol verimliliği ağın farklı koşullarından etkileniyor mu?” ve “Gidiş Dönüş Süresi ağ koşullarından etkilenir mi?” bu soruların cevabları bulmaktdadır.

Test yatağının geliştirilmesi, seçilen protokollerin test yatağına uygulanması, verilerin toplanması ve analiz edilmesi ve sonuçların tümü bir sonraki bölümde yer almaktadır.

4. UYGULAMA VE BULGULAR

Bu bölümde, önerilen test ortamının uygulamasını tartışıyor ve önerilen uygulamanın performansını farklı yönlendirme protokolleri olan , AMQP, MQTT ve COAP ile değerlendiriyoruz. “Bulgular” alt bölümünde ise uygulama aracılığıyla toplanan verilere sonuçlar halinde yer verilmiştir.

4.1. Python Programlama Dilini Kullanarak Test Yatağı Uygulaması Geliştirme

Python, mükemmel tasarımı, temiz sözdizimi, öğrenmesi kolay, büyük bir çevrimiçi topluluk tarafından desteklenen ve onu IoT geliştiricileri için çekici kılan en iyi programlama dillerinden biridir. IoT’de, Python, cihazların yazılım geliştirmesinin yanı sıra arka uç geliştirme için mükemmel bir seçimdir. IoT uygulamaları için Python ile çalışmanın birçok avantajından vardır, bunlar her türlü platform için geniş bir kitaplık ve kod geliştirme hızıdır.

Taşıyan aygıt paketlerini filtreler. 3) Yalnızca yükü ayıklayarak Önerilen uygulama üç yapıdan oluşmaktadır. IoT sensör yapısı, ağ geçidi modül yapıları ve bulut uygulama sunucu yapılarıdır. İlk yapı, önerilen uygulama, üç farklı paket biçiminde paketler oluşturan IoT cihazının hafif bir sürümünü simüle ettiği, yönlendirme protokollerini, yani COAP, MQTT ve AMQP’yi temsil eden IoT sensörüdür. İkinci yapı, bir grup IoT sensöründen gelen farklı paketleri teslim alıp ve bunları bulut uygulama sunucusuna ileten ağların beklediği ağ geçidi modülüdür. Üçüncü yapı, ağ geçidinden sunucu, IoT değer zincirindeki Application Enablement Platform’u (AEP) temsil eder.

Simülasyon uygulaması python programlama dilinin söz dizimi ve kütüphaneleri kullanılarak geliştirilmiştir. Ancak, test tek bir PC’de yapılacaktır. Uygulamanın her yapısı için üç python dosyası oluşturulmuştur; bunlar IoT cihazı, ağ geçidi ve sunucudur. Seçilen protokollerin özellikleri python dosyalarına uygulanır.

En büyük zorluk, IoT cihazı ile ağ geçidi arasındaki bağlantıyı kurmaktı. Ancak Scapy kütüphanesi yardımıyla ethernet bağlantısı kurulmuştur. Scapy paket oluşturabilir, gönderebilir, yakalayabilir, istek ve yanıtları eşleştirebilir. Ağ geçidinin diğer tarafında, soket programlama kullanılarak TCP/IP bağlantısı kurulur. Soket programlama, birbirleriyle iletişim kurmak için bir ağdaki iki düğümü birbirine bağlamanın bir yoludur.

IoT cihaz dosyasının işlevi, paketler oluşturmak ve PC'nin ethernet'i üzerinden veri gönderip almaktır. Ağ geçidi, ethernet ve TCP/IP olmak üzere iki etki alanında çalışacak şekilde geliştirilmiştir. Fonksiyonları şunlardır; 1) NIC üzerindeki cihazlardan gelen paketi alır. 2) Tam ağ geçidi kimliğini cihaz paketini çıkarır ve uygulama sunucusuna giden UDP soketine kapsüller. 4) Kaynak aygıt kimliğini çıkarır ve bir dinleme soketini bağlantı noktası numarası olarak bağlar, bu sunucuyu aynı kaynak aygıt sayı kimliğine sahip bağlantı noktasına yanıt göndermeye zorlar. 5) Aynı soket üzerinden gelen paketleri teslim alır. 6) Alınan mesajın UDP başlığını çıkarır ve onu iot cihaz başlığına yerleştirir. 7) Yanıtı ilgili cihaza gönderir. 8) Yukarı ve aşağı akış için veri araştırması ve analizi yapmak için tüm bilgileri taşıyan Veri çerçevesi dosyaları oluşturulur. Son olarak, sunucu ağ geçidinden paketleri alır ve ACK mesajını gönderir.

4.2. Testin Adımları ve Veri Çerçevesi'lerin Oluşturulması

Simülasyonumuzda, önerilen uygulamanın performans değerlendirmesi aşağıdaki değerlendirme ölçütleri dikkate alınarak yapılmıştır:

- Yukarı akım işlem gecikmesi: IoT cihaz paketi başlığının kapsülünü açıp, yükü TCP / UDP başlığına kapsülleyip ve paketi bulut uygulama sunucusuna göndermek için yürütme süresidir.
- Aşağı akım işlem gecikmesi: TCP / UDP başlığının kapsülünü açıp, yükü IoT cihaz paketi başlığına kapsülleyip ve yanıta karşılık geleni IoT cihazına geri göndermek için yürütme zamanıdır.
- Ortalama işlem gecikmesi: Hem yukarı hem de aşağı yönde seyahat ederken IoT cihaz paketini işlemek için gidiş-dönüş işlem süresidir.
- Yukarı akım throughput: Ağ geçidinin IoT cihazından bulut uygulama sunucusuna yukarı akış yönünde işleyebileceği saniye başına toplam paket sayısıdır.
- Aşağı akım throughput: Ağ geçidinin bulut uygulama sunucusundan IoT cihazına aşağı akış yönünde işleyebileceği saniye başına toplam paket sayısıdır.

Seçilen üç protokol, 1024'lük sabit arabellek boyutu ve sekiz farklı trafik hacmiyle test edilmiştir; bu trafik hacımlar sırasıyle 25, 50, 100, 200, 300, 400, 500 ve 600'dür.

İşte bu testin adımları:

1- IoT, ağ geçidi ve sunucu, bu uçunu ait python dosyalarının çalıştırılması gerekiyor.

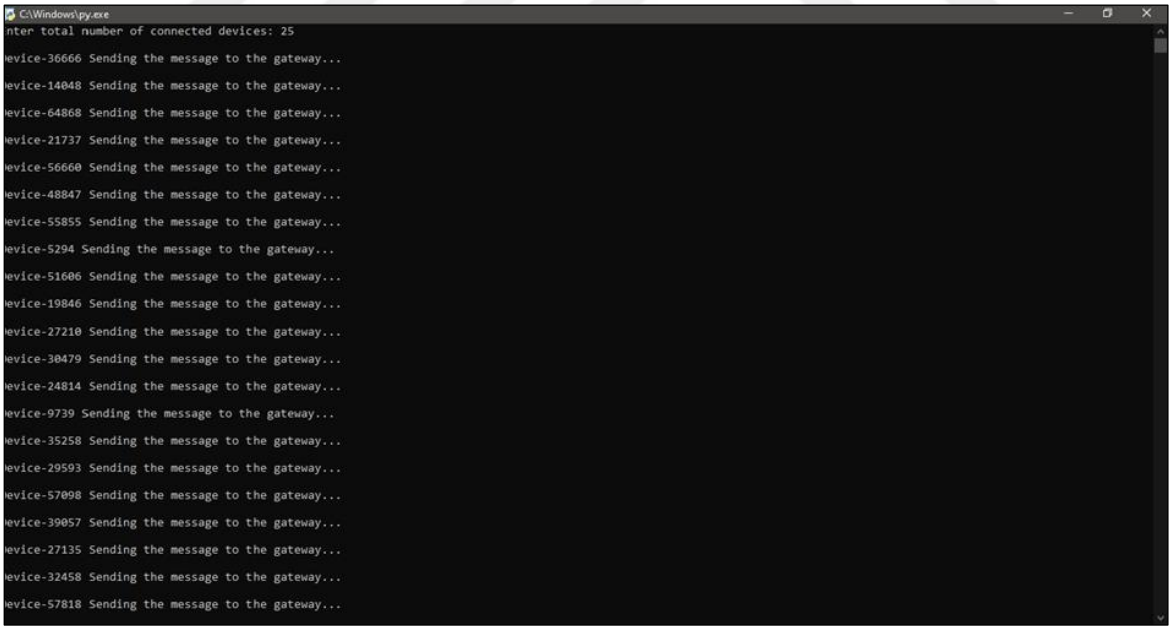
2- IoT cihazı “Enter the total number of connected devices:” yazısına karşılık ister, IoT cihazı “Hi server, I am device (gönderdiğin paket sayısının numarası yazılır)” yazan paketleri ethernet’e göndererek sunucu yanıtını bekler.

3- Ağ geçidi ethernetten paketleri teslim alır, sayacı başlatır, kaynak kimliğini çıkarır, buna TCP/IP başlığı ekler ve kaynak cihaz olarak port numarası ile sunucuya gönderir ve paketler ağ geçidinden ayrıldığında sayacı durdurur.

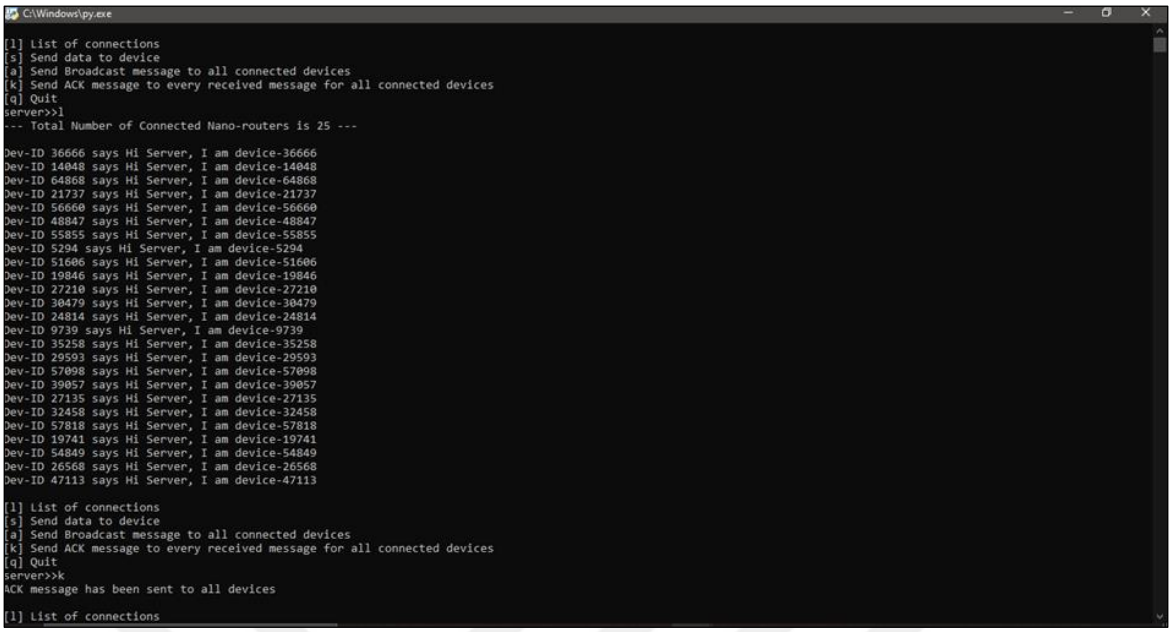
4- Sunucu paketleri alır ve bağlı her cihaza ACK mesajları ile cevap verir.

5- Ağ geçidi sunucudan cevabı alır, sayacı başlatır, cevabı ilgili cihaza gönderir ve sayacı durdurur ve simülasyonu bitirir. Sonunda, yukarı ve aşağı akış için veri araştırması ve analizi yapmak için tüm bilgileri taşıyan csv dosyaları olarak Veri çerçevesi oluşturur.

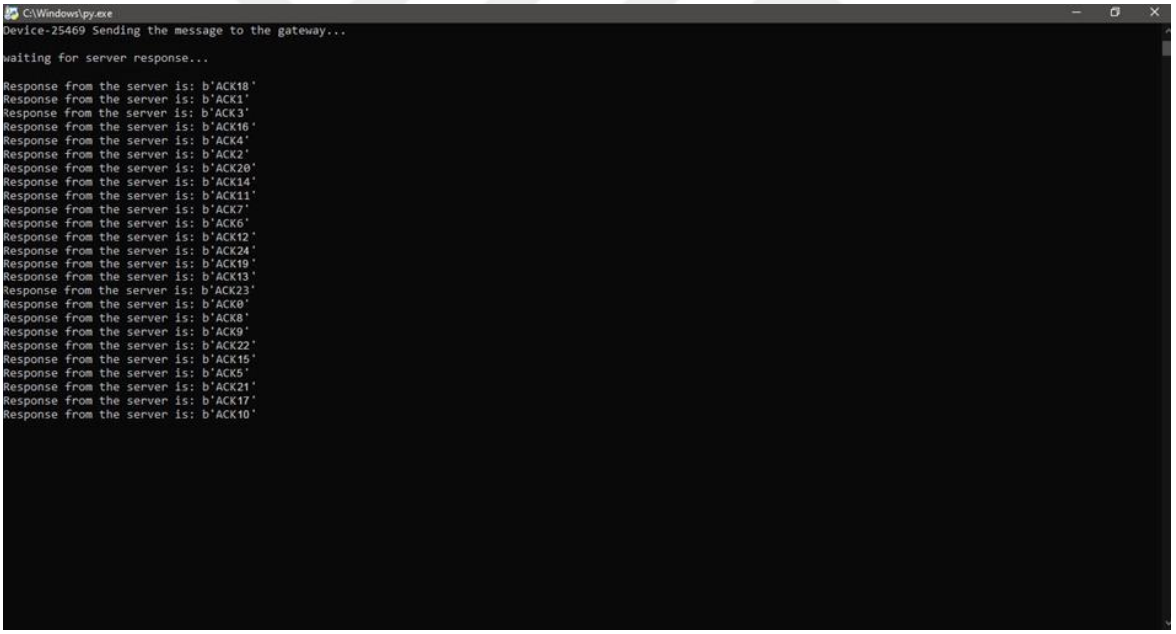
6- Her IoT cihazı ACK mesajını alır.



Resim 4.1. IoT cihazı paketleri gönderme aşaması



Resim 4.2. Sunucu ACK mesajını gönderme aşaması



Resim 4.3. IoT cihazının paketleri teslim alma aşaması

4.3. Analiz Etme

Simülasyon sona erdikten sonra, ağ geçidi, Panda kütüphanesi yardımıyla, yukarı ve aşağı akış için veri araştırması ve analizi yapmaları için tüm bilgileri taşıyan Veri çerçevesi'e csv dosyaları şeklinde kaydolur.

Bu araştırmada verilerin görsel olarak analiz edilmesi için Pandas, numpy, matplotlib ve seaborn gibi Anaconda'ya ait uygulama kütüphaneleri kullanılmıştır. Pandas csv dosyalarıyla ilgilenir, Numpy sayısal işlemlerle ilgilenir, Matplotlib ve seaborn rakamların çizilmesiyle ilgilenir. Kod blokları, yukarı akış, aşağı akış, Gidiş-Dönüş süresi işleme sürelerini, yukarı akış throughput ve aşağı akış throughput hesaplamak için geliştirilmiştir. Tüm işlemler Jupyter Notebook ortamında yapılır.

Csv dosyaları Panda kullanılarak Jupyter Notebook'a aktarılır. Şekil 19'da gösterildiği gibi "up start_time", "up_stop_time"dan çıkarılır ve yukarı akış işlem gecikmesi olan "up_exec_time"ı verir. Aynı işlem aşağı akış işleme gecikmesi için de geçerlidir, " dwn start_time"ı " dwn stop_time"dan çıkararak "dwn_exec_time"ı verir. Son olarak "up_exec_time", "dwn_exec_time"a eklenmesiyle belirli bir paket için RTT olan "GW_exec_delay" verir. Her bir "up_exec_time", "dwn_exec_time" ve "GW_exec_delay" öğelerinin ortalama sayısını hesaplamak ve ortalama Gidiş-Dönüş işlem gecikmesini milisaniye cinsinden görüntülemek için bunları 1000 ile çarpmalı ve sonuçlar bölümünde de görülebileceği gibi gerekliydi. Ayrıca çizelgeleri çizmeyi de kolaylaştırır. Bu işlem, her protokol için farklı trafik hacimleriyle yapılır.

Çizelge 4.1. RTT hesaplama

	Payload	up_start_time	up_stop_time	ACKS	dwn_start_time	dwn_stop_time	up_exec_time	dwn_exec_time	GW_exec_delay
0	b'Hi Server, I am device-37283'	7.872480	7.873358	ACK0	10.590836	10.603280	0.000878	0.012443	0.013321
1	b'Hi Server, I am device-34402'	7.873669	7.874369	ACK1	10.591627	10.607453	0.000699	0.015826	0.016525
2	b'Hi Server, I am device-28481'	7.925323	7.926100	ACK2	10.592138	10.607884	0.000777	0.015745	0.016522
3	b'Hi Server, I am device-34305'	7.926380	7.927062	ACK3	10.592668	10.608267	0.000682	0.015599	0.016280
4	b'Hi Server, I am device-7139'	7.927365	7.927904	ACK4	10.593174	10.610892	0.000539	0.017718	0.018257

Seçilen uygulama protokolleri, diğer throughput yönleriyle de değerlendirildi. Bir başka kod bloğu, hem yukarı akış verimini hem de aşağı akış verimini hesaplamak için geliştirilmiştir. bu blok, ilk "up start_time"ı son "up_stop_time"dan çıkarır ve bunu bağlı IoT cihazının sayısına bölersek ve yukarı akış throuputunu buluruz. Ayrıca ilk "dwn_start_time"ı son "dwn_stop_time"dan çıkarır ve bunu bağlı IoT cihazının sayısına bölersek ve aşağı akış throuputunu buluruz. Bu işlem, her protokol için farklı trafik hacimleriyle yapılır.

Çizelge 4.2. Throughput hesaplama

	Payload	up_start_time	up_stop_time	ACKS	dwn_start_time	dwn_stop_time
0	b'Hi Server, I am device-37283'	7.872480	7.873358	ACK0	10.590836	10.603280
1	b'Hi Server, I am device-34402'	7.873669	7.874369	ACK1	10.591627	10.607453
2	b'Hi Server, I am device-28481'	7.925323	7.926100	ACK2	10.592138	10.607884
3	b'Hi Server, I am device-34305'	7.926380	7.927062	ACK3	10.592668	10.608267
4	b'Hi Server, I am device-7139'	7.927365	7.927904	ACK4	10.593174	10.610892

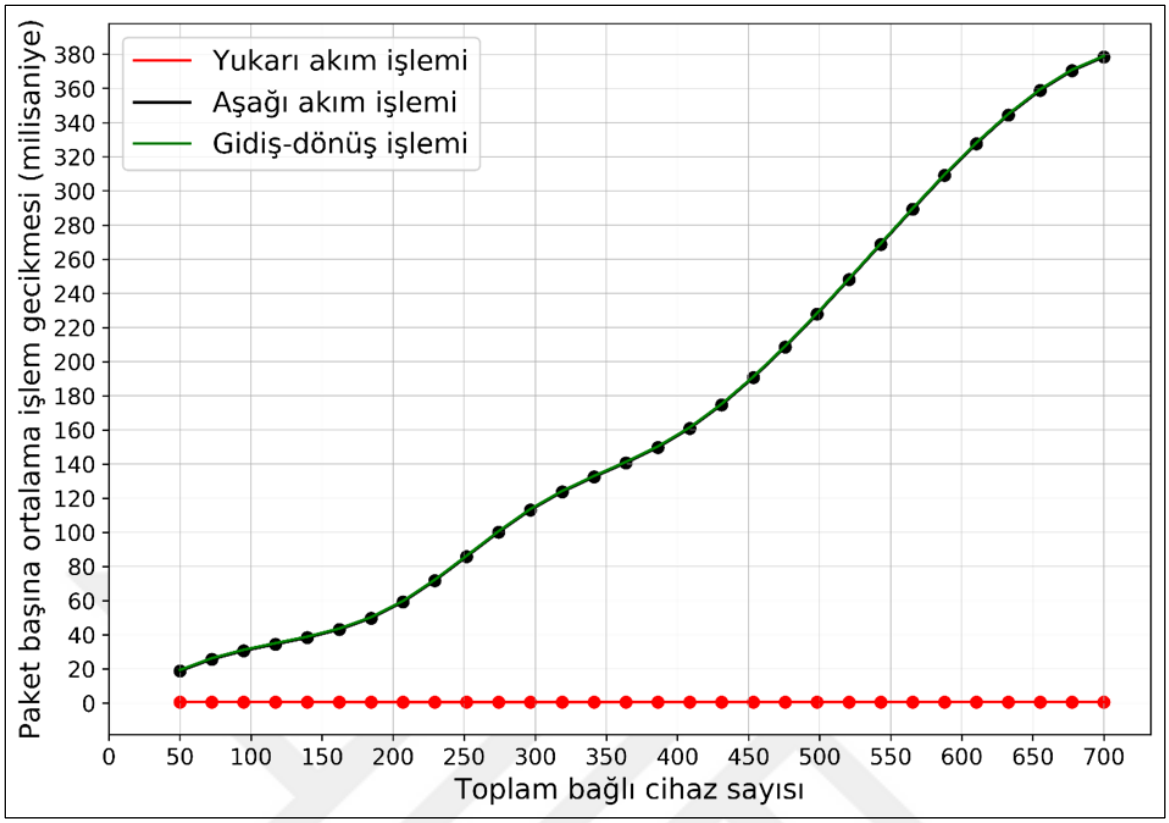
Seçilen uygulama protokollerin değerlendirmesi için yapılmış testlerin bulgularına Bulgular bölümünde yer verilmiştir.

4.4. Bulgular

IoT cihazları tarafından gönderilen paket sayısını değiştirerek performansı test ettik ve yukarı akış, aşağı akış ve gidiş-dönüş işlem gecikmesinin yanı sıra yukarı akış ve aşağı akış aktarım hızının ortalama işlem gecikmesini kontrol ettik.

4.4.1. Gelişmiş mesaj kuyruklama protokolü (AMQP)

Şekil 4.1. AMQP protokolünde, ortalama yukarı akış işlem gecikmesi, ortalama aşağı akış işlem gecikmesinden daha kısa olduğunu göstermektedir. Bu, AMQP paketlerinden paketin kapsülünü açıp ve TCP / UDP başlığı ile kapsüllemek, veri yükünün TCP / UDP başlığı ile kapsüllenip AMQP başlığına kapsüllenmesinden daha düşük işlem süresi gerektirdiği anlamına gelir. Ayrıca, IoT cihazlarından gönderilen toplam paket sayısının artırılması, ortalama yukarı akış işleminin gecikmesi üzerinde yüksek bir etkiye sahip değildir ama aşağı akış işleminin gecikmesi üzerinde yüksek bir etkiye sahip olması toplam gidiş-dönüş işlem süresi üzerinde doğrudan bir etkiye sahip olmasına neden olur.



Şekil 4.1. Ortalama işlem gecikmesi ile AMQP'deki bağlı cihazların sayısı arasındaki ilişki

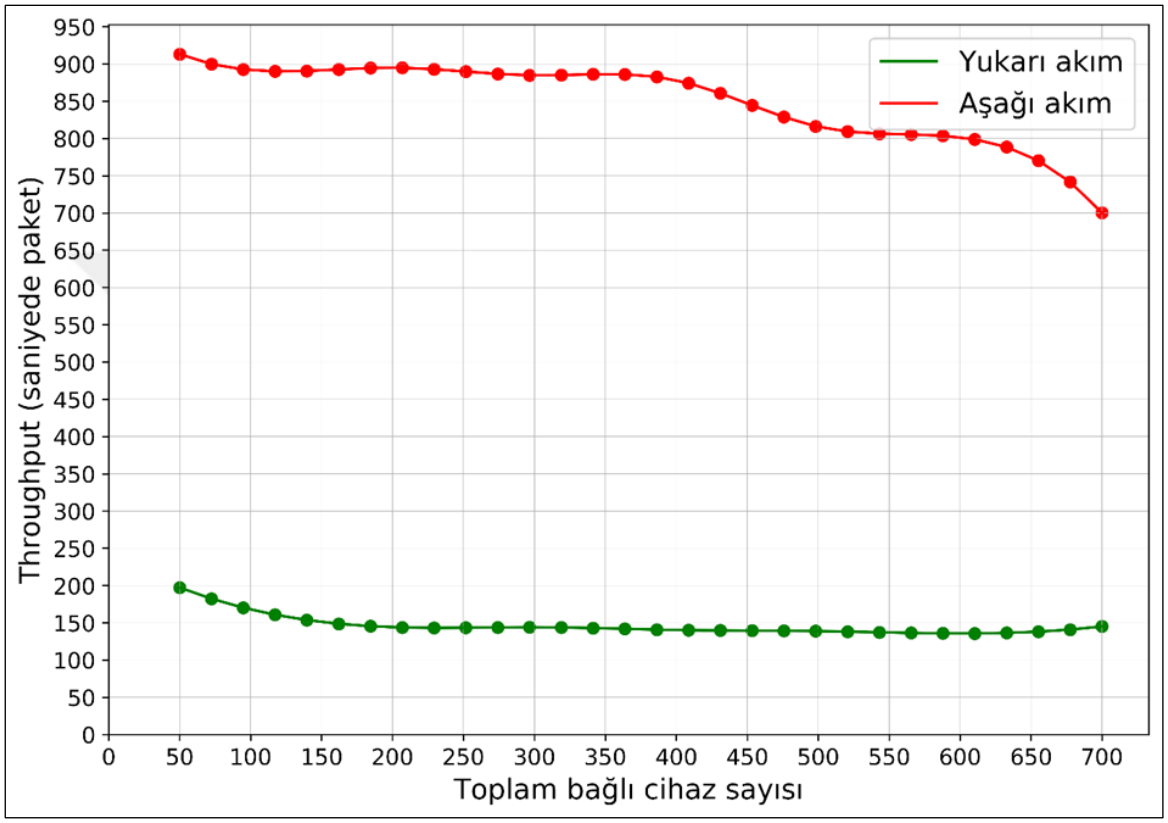
Şekil 4.2. AMQP yönlendirme protokolüne göre ağ geçidinin yukarı ve aşağı akış verimini göstermektedir. Aşağı akış veriminin, yukarı akış veriminden neredeyse dört kat daha yüksek olduğunu gösterir. Ayrıca, IoT cihazları tarafından gönderilen toplam paket sayısı arttıkça, ağ geçidinin yukarı ve aşağı akış hızı biraz azalır.

4.4.2. Mesaj kuyruklama telemetri iletimi (MQTT)

Bağlı IoT cihazlarının toplam sayısını manipüle ederek performansını test ettik, her cihaz ağ geçidine tek bir paket gönderir ve yukarı akış, aşağı akış ve gidiş-dönüş işlem gecikmelerinin ortalama işlem gecikmesinin yanı sıra yukarı ve aşağı akış aktarım hızını kontrol eder.

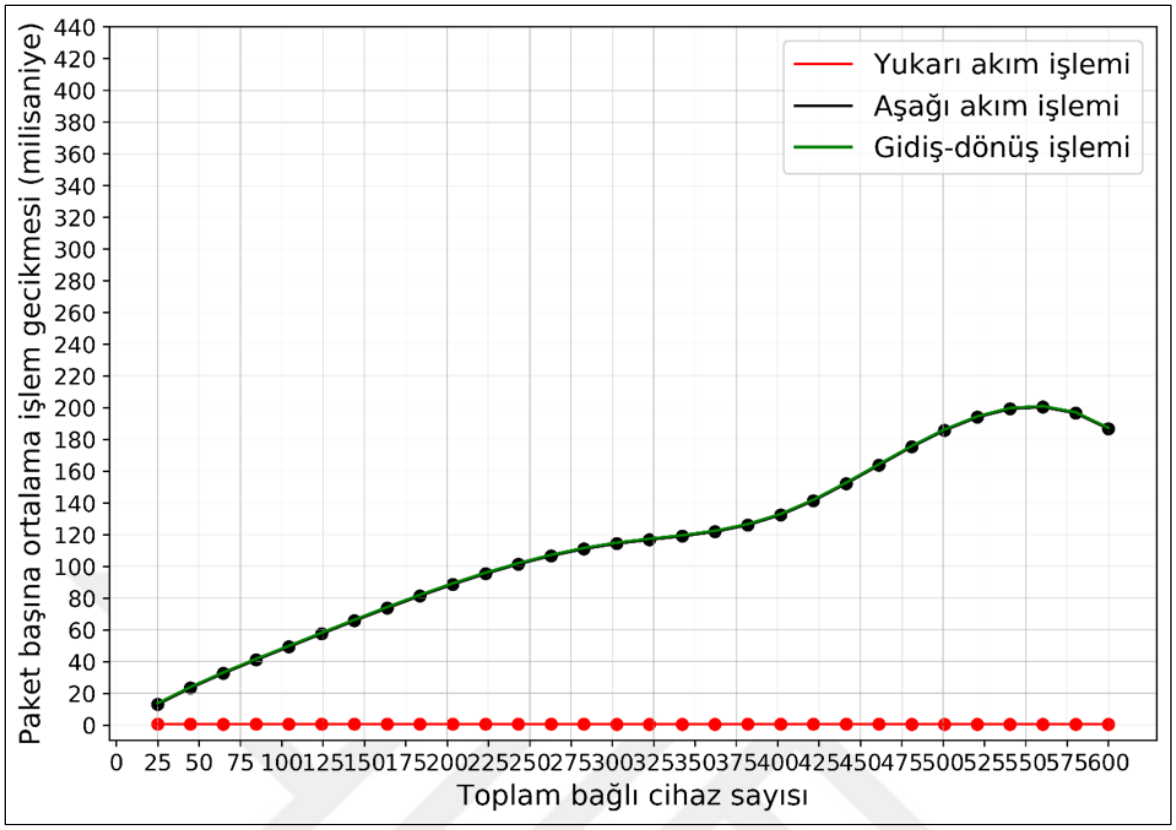
Şekil 4.3. MQTT protokolünde, ortalama yukarı akış işlem gecikmesi, ortalama aşağı akış işlem gecikmesinden daha kısa olduğunu göstermektedir. Bu, MQTT paketlerinden gelen paket kapsülünün açılması ve TCP / UDP başlığı ile kapsüllemek, veri yükünün TCP / UDP başlığının kapsülünün acılıp, MQTT başlığıyla kapsüllenmesine göre daha düşük işlem süresi gerektirdiği anlamına gelir.

Bunun yanı sıra, bağlı cihazların toplam sayısı arttıkça, yukarı akış işlem gecikmesi farklı sayıda bağlı cihaz için neredeyse sabittir, aşağı akış işlem süresi artarken, buna bağlı olarak toplam ortalama gidiş-dönüş işlem gecikmesi de artar. Bu, toplam bağlı cihaz sayısının ve aşağı akış işlem süresinin ölçeklenebilirlik performansı üzerinde doğrudan bir etkisi olduğu anlamına gelir.

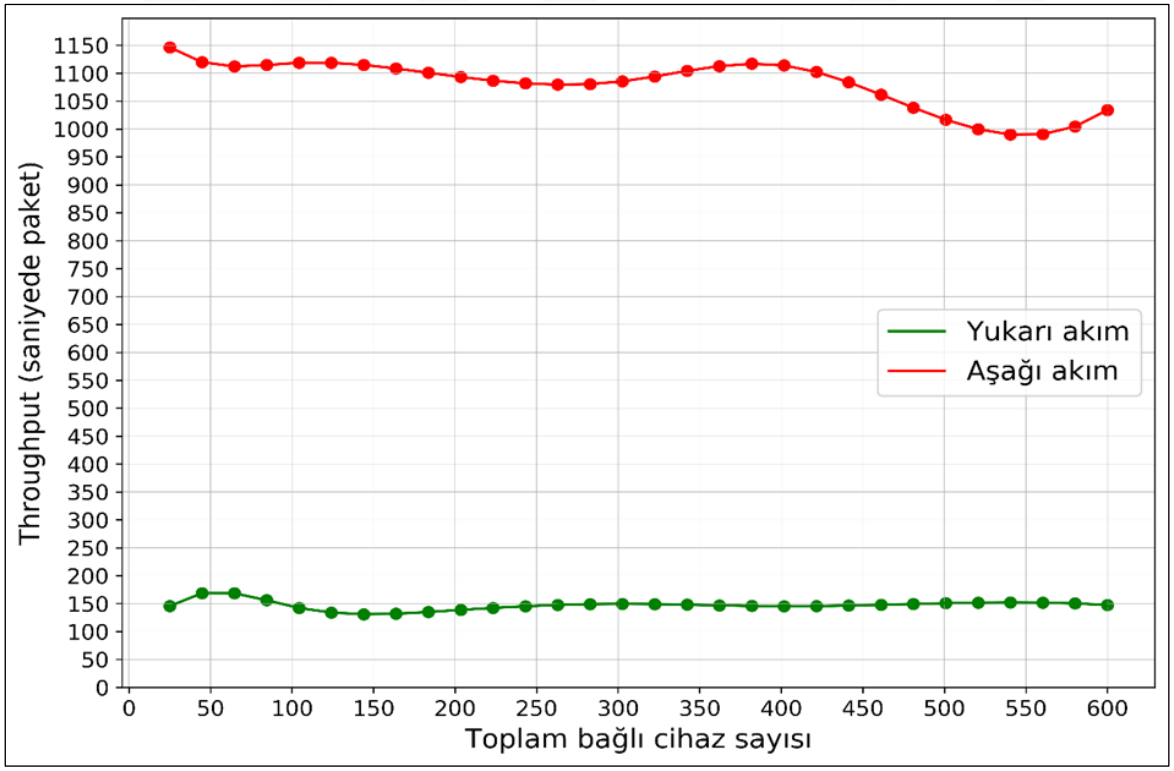


Şekil 4.2. Üretilen throughput ile AMQP'deki bağlı cihazların sayısı arasındaki ilişki

Şekil 4.4. MQTT yönlendirme protokolüne göre ağ geçidinin yukarı ve aşağı akış verimini göstermektedir. Aşağı akış veriminin, yukarı akış veriminden neredeyse yedi kat daha yüksek olduğunu gösterir. Ayrıca, IoT cihazları tarafından gönderilen toplam paket sayısı arttıkça, ağ geçidinin aşağı akış verimi biraz azalır.



Şekil 4.3. Ortalama işlem gecikmesi ile MQTT'deki bağlı cihazların sayısı arasındaki ilişki



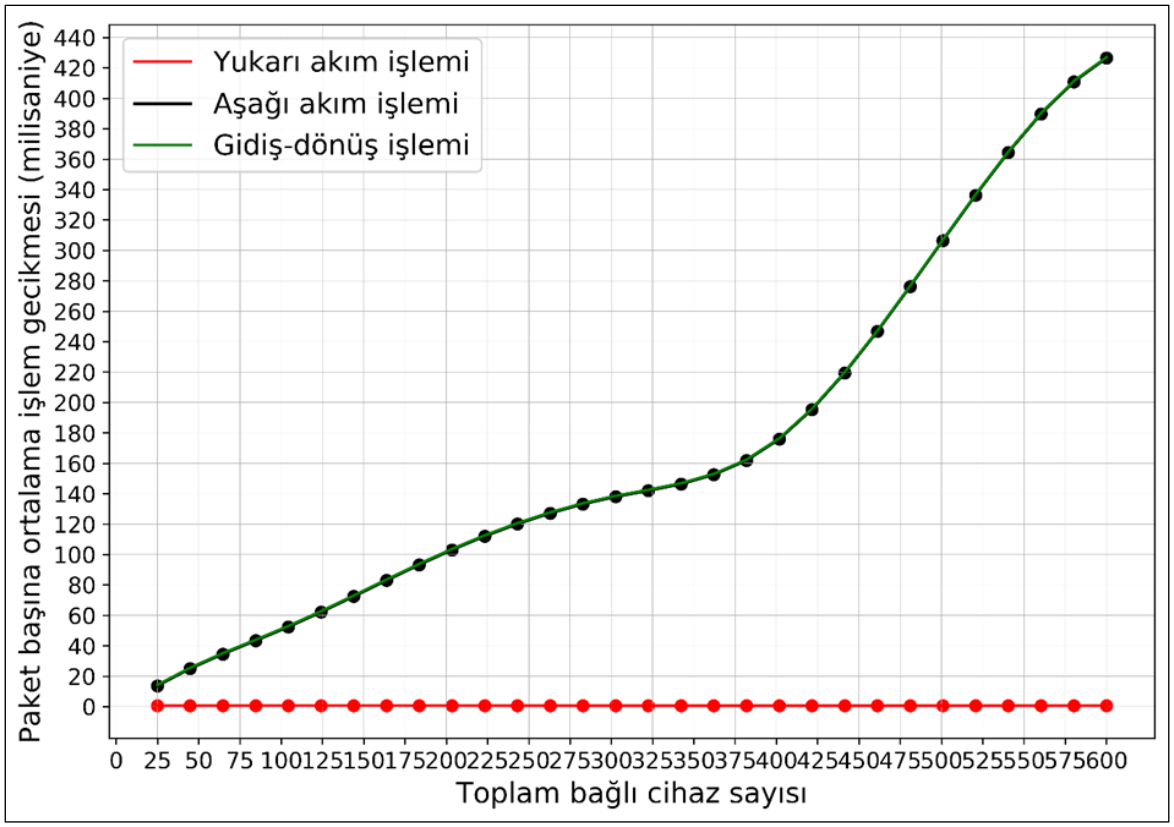
Şekil 4.4. Üretilen throughput ile MQTT'deki bağlı cihazların sayısı arasındaki ilişki

4.4.3. Kısıtlı uygulama protokolü (COAP)

Bağlı IoT cihazlarının toplam sayısını manipüle ederek performansını test ettik, her cihaz ağ geçidine bir paket gönderir, yukarı akış, aşağı akış ve gidiş-dönüş işlem gecikmesinin ortalama işlem gecikmesiyle birlikte yukarı ve aşağı akış veriminide kontrol eder.

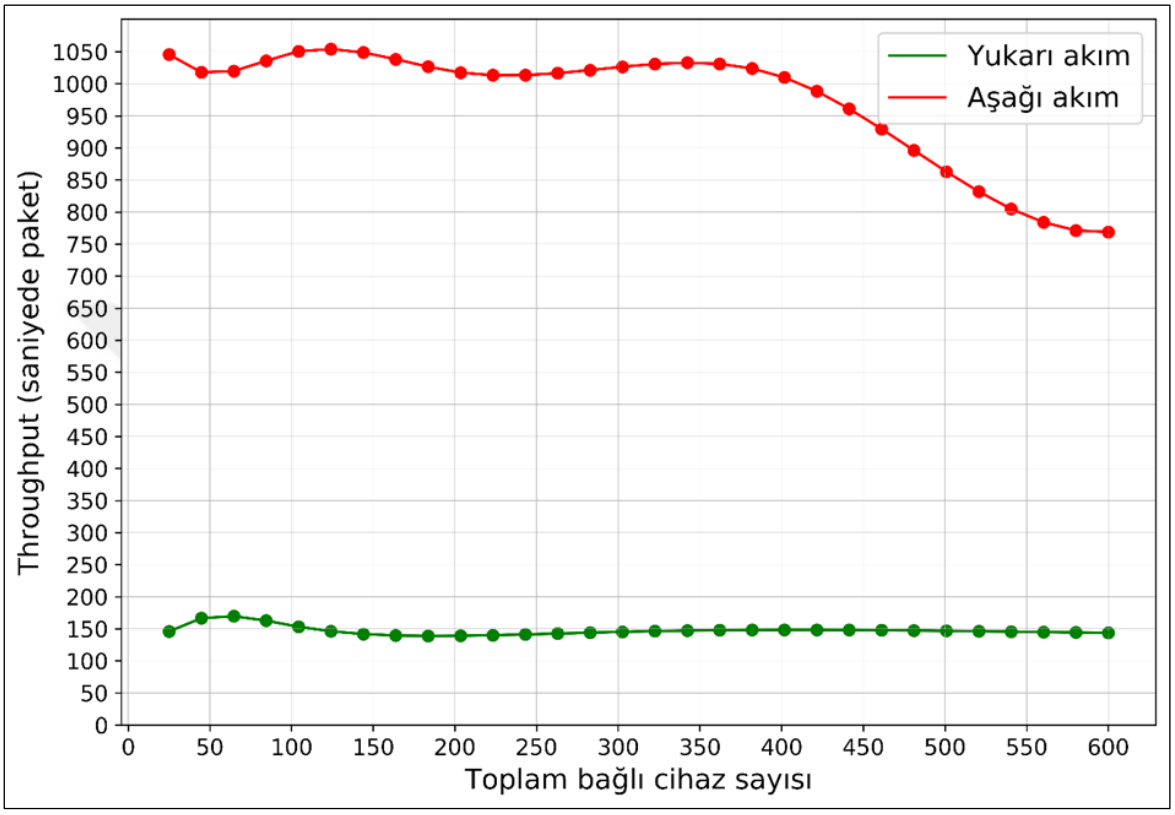
Şekil 4.5. COAP protokolünde, ortalama yukarı akış işlem gecikmesi, ortalama aşağı akış işlem gecikmesinden daha kısa olduğunu göstermektedir. Bu, COAP paketlerinden gelen paket kapsülünün acılması ve TCP / UDP başlığıyla kapsüllemenin, veri yükünün TCP / UDP başlığının kapsülünün acılması ve COAP başlığıyla kapsüllenmesine göre daha düşük işlem süresi aldığı anlamına gelir.

Ayrıca, bağlı cihazların toplam sayısı arttıkça, yukarı akış işlem gecikmesi farklı sayıda bağlı cihaz için hemen hemen sabittir, ancak aşağı akış işlem süresi artar, buna bağlı olarak toplam ortalama gidiş-dönüş işlem gecikmesi de artar. Bu, bağlı cihazların toplam sayısının ve aşağı akış işlem süresinin, ortalama gidiş-dönüş işlem gecikmesi üzerinde doğrudan bir etkiye sahip olduğu anlamına gelir.



Şekil 4.5. Ortalama işlem gecikmesi ile CoAP'deki bağlı cihazların sayısı arasındaki ilişki

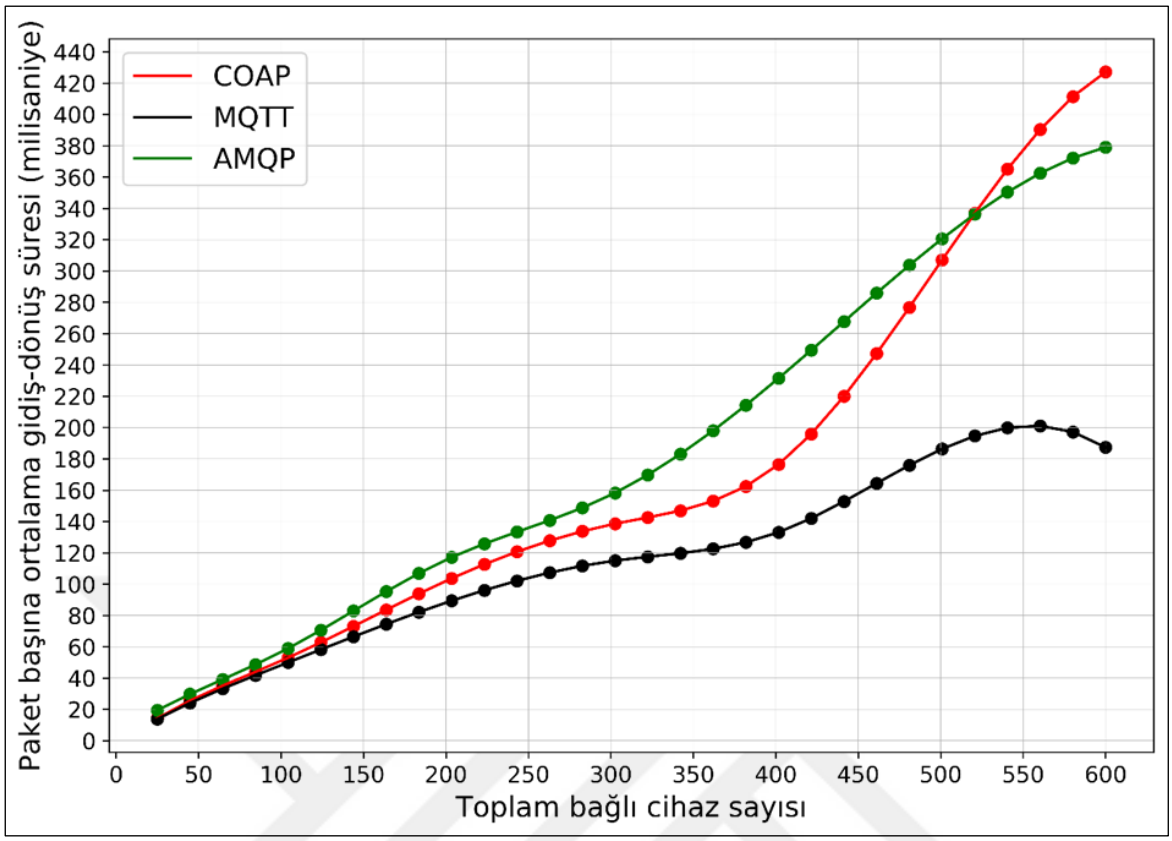
Şekil 4.6. COAP yönlendirme protokolüne göre ağ geçidinin yukarı ve aşağı akış verimini göstermektedir. Aşağı akış veriminin, yukarı akış veriminden neredeyse altı kat daha yüksek olduğunu gösterir. Ayrıca, IoT cihazları tarafından gönderilen toplam paket sayısı arttıkça, ağ geçidinin aşağı akış verimi azalır.



Şekil 4.6. Üretilen throughput ile CoAP'deki bağlı cihazların arasındaki ilişki

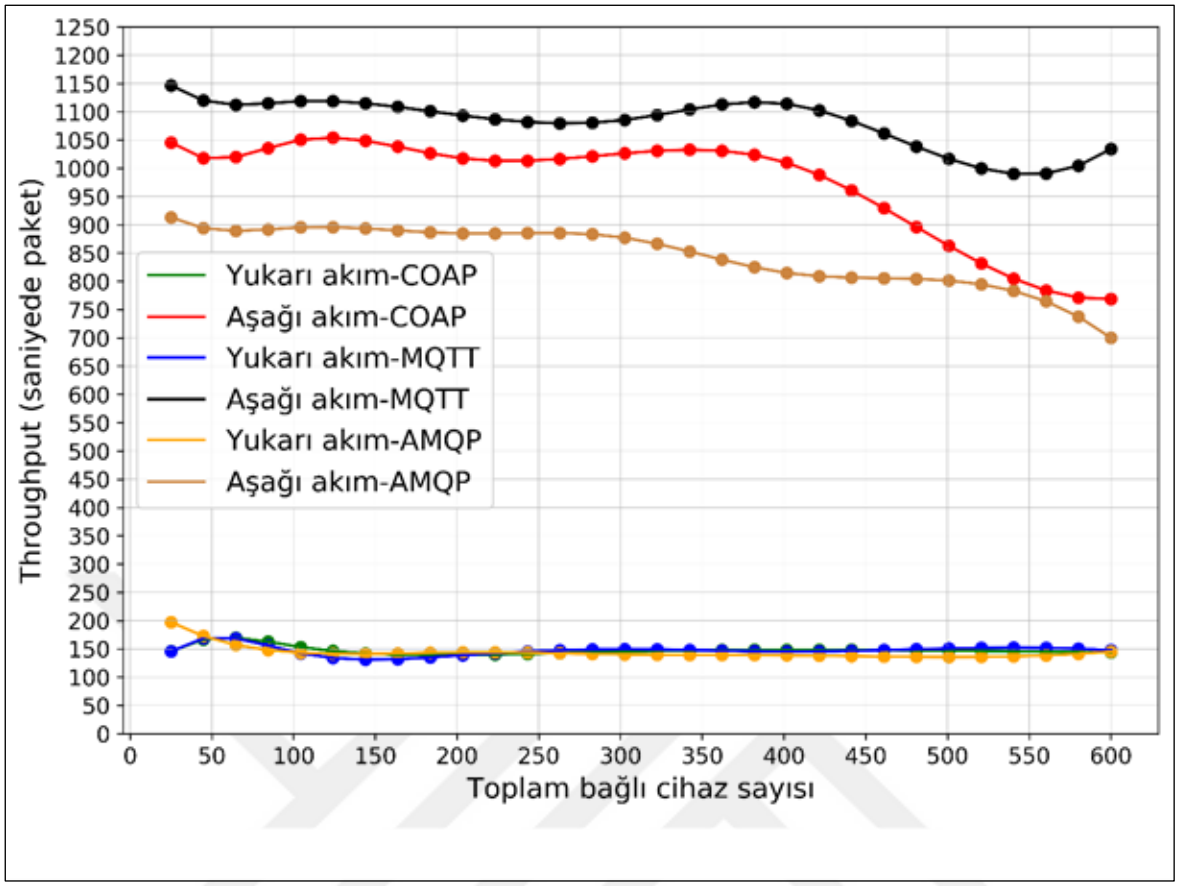
4.4.4. Yönlendirme protokolleri karşılaştırması

Şekil 4.7. ortalama işlem gecikmesi ile AMQP, MQTT ve COAP yönlendirme protokolleri için bağlı IoT cihazlarının sayısı arasındaki ilişkiyi göstermektedir. Örneğin 150 bağlı cihazla, kücük bir trafik hacmi oluşturduğumuz zaman belirgin bir şekilde AMQP yönlendirme protokolünün, diğer yönlendirme protokollerine göre en yüksek ortalama gidiş-dönüş işlem gecikmesine sahip olduğu görülür. COAP yönlendirme protokolü ikinci sırada gelir ve ardından MQTT, en düşük ortalama gidiş-dönüş işlem gecikmesine sahip olduğu tespit edilir. Yüksek trafik hacmi söz konusu olduğunda, örneğin 550 bağlı cihaz ve fazlasında, AMQP yönlendirme protokolü, COAP'tan daha düşük gidiş-dönüş işleme gecikmesi sağlar, ancak MQTT diğer yönlendirme protokolleri arasında hala en düşük gidiş-dönüş işlem gecikmesine sahiptir.



Şekil 4.7. Tüm yönlendirme protokolleri için ile bağlı cihaz sayısı arasındaki ilişki

Öte yandan, Şekil 4.8. AMQP, MQTT ve COAP yönlendirme protokolleri için bağlı IoT cihazlarının toplam sayısı ile yukarı ve aşağı akış veriminin arasındaki ilişkiyi göstermektedir. Birbirlerine neredeyse yakın olduklarından, tüm yönlendirme protokolleri için yukarı akış veriminde yüksek bir fark olmadığı açıktır. Aşağı akış verimindeyken, sonuçlar MQTT yönlendirme protokolünün en yüksek verime sahip olduğunu ve COAP'ın ikinci en yüksek verime sahip olduğunu, ardından AMQP'nin tüm yönlendirme protokolleri arasında en düşük verime sahip olduğunu göstermektedir.



Şekil 4.8. Tüm yönlendirme protokolleri için throughput ve bağlı cihaz sayısı arasındaki ilişki

5. SONUÇ VE GELECEK ÇALIŞMALAR

Önceki bölümlerde üç uygulama protokolü; Advanced Message Queuing Protocol, Message Queuing Telemetry Transprot ve Constrained Application Protocol tanıtıldı, karşılaştırıldı, test edildi ve değerlendirildi. Açık kaynak python programlama dili kullanılarak bir test ortamı yazılımı da önerildi. Bu bölümde, tüm değerlendirmenin sonucu, gelecek çalışmaları için önerilerle birlikte özetlenecektir.

5.1. Sonuç

Bu araştırmada, nesnelerin internetinin seçilen uygulama katmanı protokolleri, önerilen test ortamı yazılımı kullanılarak simüle edilmiş ve farklı hacim trafikleri ile testler, üç protokol performansı üzerinde çalıştırılmıştır. Sonuçlar, düşük hacimli gidiş-dönüş süresi için her protokol neredeyse aynı gidiş-dönüş işleme gecikmesine sahip olduğunu göstermektedir. Paket sayısı artarken, protokoller arasındaki fark da artar, yüksek hacimli gidiş dönüş süresinin bir sonucu olarak, MQTT diğer protokoller arasında en düşük ortalama gidiş-dönüş işleme gecikmesine sahiptir. Aynı şekilde, yukarı akış verim sonuçlarda her protokol neredeyse birbirine yakın olduğunu, tüm protokoller arasında yüksek bir fark olmadığını açıkça göstermektedir.

Testlerin sonuçlarına göre, MQTT protokolü, aşağı akış verimi açısından protokoller arasında en yüksek verim haline geldi. Her bir protokolün başlık boyutu protokolün etkinliğinde büyük rol oynadığı gözlemlenen en önemli noktalardan biridir.

2015 yılında Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan ve Colin Keng-Yan TAN tarafından yapılan “Ortak Bir Ara Katman Aracılığıyla MQTT ve CoAP Performans Değerlendirmesi” çalışmasına göre, MQTT mesajları daha düşük paket kayıp oranlarında CoAP mesajlarından daha düşük gecikmeye ve daha yüksek kayıp oranlarında CoAP mesajlarından daha yüksek gecikmeyle çalıştığını göstermektedir. Ayrıca, mesaj boyutu küçük olduğunda ve kayıp oranı %25’e eşit veya daha az olduğunda, CoAP, mesaj güvenilirliğini sağlamak için MQTT’den daha düşük ek trafik oluşturur (Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan and Colin Keng-Yan TAN 2015). Halbuki, bu çalışmada, tüm testler için sabit boyutlu bir yük ve üç protokolü test etmek için tek bir topoloji kullanıldı.

5.2. Gelecek Çalışmalar

Bu araştırmada, üç protokol önerilen test ortamına uygulanarak, üç uygulama protokolü gidiş-dönüş süresi ve iş hacmi açısından simüle edilmiştir. Her üç protokol için tek bir ağ senaryosu kullanılmış ve IoT sensörlerinden sunucuya gönderilen tüm paketler sabit yük ile yüklenmiştir.

Gelecekteki çalışma olarak, bu üç protokol üzerinde, farklı ağ senaryoları, farklı boyuttaki yükler ve paket kaybı gibi ek performans ölçütleri ile gerçek uygulama test ortamın altında denemesi gibi başka çalışmalar da yapılabilir. Elbette, her protokolün farklı uygulamaları bu açılardan farklılık gösterecektir ve burada her protokolün yalnızca bir uygulamasını dikkate aldık.

KAYNAKLAR

Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi,Mohammed Aledhari, Moussa Ayyash. (2015). Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE COMMUNICATION SURVEYS & TUTORIALS,, 11*(4), 36-45.

Bormann, C. (2012). Using CoAP with IPsec draft-bormann-core-ipsec-for-coap-00. *Journal of Network and Computer Applications, 7*(3), 50-62.

Contributors, T. (2008). AMQP Protocol Specification. *Version 0-9-1, 13 November 2008, 2*(1), 30-35.

Danish Bilal Ansari, Atteeq-Ur-Rehman, Rizwan Ali Mughal. (2018). Internet of Things (IoT) Protocols: A Brief Exploration of MQTT and CoAP. *International Journal of Computer Applications, 4*(3), 77 - 85.

Dinesh Thangavel, Xiaoping Ma, Alvin Valera, Hwee-Xian Tan and Colin Keng-Yan TAN. (2015). Performance Evaluation of MQTT and CoAP via a common middleware. *IEEE, 2*(1), 41-48.

E. Rescorla, N. Modadugu,. (2012). Datagram Transport Layer Security Version 1.2. *Internet Engineering Task Force, 3*(4), 221-235.

Friedemann Mattern, Christian Floerkemeier. (2010). From the Internet of Computers to the internet of things. *From Active Data Management to Event-Based Systems*, *19*(4)242–259.

George Kokkonis, Angelos Chatzimparmpas, Sotirios Kontogiannis. (tarih yok). Middleware IoT protocols performance evaluation for carrying out clustered data. *IEEE, 8*(4), 26-33.

Hamid M. Hasan, Bahaa K. mohammed. (2018). Evaluation of MQTT Protocol for IoT Based Industrial Automation. *IJESC, 8*(1), 188 - 192.

Istabraq M. Al-Joboury, E. H.-H. (2018). Performance Analysis of Internet of Things Protocols Based Fog/Cloud over High Traffic. *Jornal of foundzmental and applied sciences, 6*(2), 70-77.

Istabraq M. Al-Joboury, E. H.-H. (2018). Performance Analysis of Internet of Things Protocols Based Fog/Cloud over High Traffic, Vol 3rd. *Jornal of foundzmental and applied sciences, 4*(3), 215 - 223.

Iulia Florea, R. R. (tarih yok). Survey of Standardized Protocols for the Internet of Things. *2017 21st International Conference on Control Systems and Computer Science*, (s. 314 - 325).

Iulia Florea, R. R. (2017). Survey of Standardized Protocols for the Internet of Things. *IEEE, 8*(1), 10-14.

Joel L. Fernandes, Ivo C. Lopes, Joel J. P. C. Rodrigues, Sana Ullah. (2014). Performance Evaluation of RESTful Web Services and AMQP protocol. *IEEE, 13*(2), 81-90.

Jorge E. Luzuriaga, Miguel Perez, Pablo Boronat, Juan Carlos Cano, Carlos Calafate, Pietro Manzoni. (2015). A comparative evaluation of AMQP and MQTT protocols over unstable and mobile networks. *IEEE, 12*(3), 931-936.

Jyun-Yao Huang, Po-Hsin Tsai, I-En Liao. (2017). Implementing Publish/Subscribe Pattern for CoAP in Fog Computing Environment. *IEEE, 9*(4), 36 - 40.

Konstantinos Fysarakis, Othonas Soultatos, Ioannis Askoxylakis, Harry Manifavas. (2016). Which IoT Protocol? Comparing Standardized Approaches over a Common M2M Application. *IEEE, 9*(3), 334 - 347.

Konstantinos, F., Askoxylakis, I., Soultatos, O., Papaefstathiou, I., Manifavas, C., & Katos, V. (2016). Which IoT Protocol? Comparing standardized approaches over a common M2M application. *The task force, 3*(2), 61-66.

Lars Durkop, Bjorn Czybik, Jurgen Jasperneite. (2018). Performance Evaluation of M2M Protocols Over Cellular Networks in a Lab Environment. *International Conference on Intelligence in Next Generation Networks, 8*(3), 78 - 89.

Martin Stusek, Krystof Zeman, Pavel Masek, Jindriska Sedova, Jiri Hosek. (2019). IoT Protocols for Low-power Massive IoT: A Communication Perspective. *International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT, 2*(1), 261-268.

Matteo Collina, Marco Bartolucci, Alessandro Vanelli-Coralli, Giovanni Emanuale Corazza. (2014). Internet of Things Application Layer Protocol Analysis over Error and Delay prone Links. *Advanced Satellite Multimedia Systems Conference and the 13th Signal Processing for Space Communications, 4*(2), 390 - 404.

Matthias Pohl, Janick Kubela, Sascha Bosse, Klaus Turowski. (2018). Performance Evaluation of Application Layer Protocols for the Internet-of-Things. *International Conference on Enterprise Systems, 6*(1), 24-30.

Mehmet Ali EBLEME, Cüneyt BAYILMIŞ, Ünal ÇAVUŞOĞLU, Kerem KÜÇÜK. (2020). CoAP and Its Performance Evaluation. *Sakarya University Journal of Science, 24*(2), 78-85.

Nastasa, L. (2017). Security in the Interner of Things: A survey on Application Layer Protocol. *IEEE, 5*(2), 45-56.

OASIS. (2012). OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0, Vol 2nd. *OASIS Standard*, 85-96 .

OASIS. (2014). MQTT Version 3.1.1. *OASIS Standard, 8*(2), 14-18.

OUAKASSE Fathia, RAKRAK Said. (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,, 7*(2), 211- 218.

Pallavi Sethi and Smruti R. Sarangi. (2017). Internet of Things: Architectures, Protocols, and Applications. *Journal of Electrical and Computer Engineering, 9*(5), 114-120.

Paolo Bellavista, Alessandro Zanni. (2016). Towards Better Scalability for IoT-Cloud Interactions. *IEEE 2nd International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow (RTSI), 8*(4), 57 - 64.

Priyanka Thota, Yoohwan Kim. (2016). Implementation and Comparison of M2M Protocols for the internet of things. *Intl Conf on Applied Computing and Information Technology, 3*(3), 22- 30.

Ronny Klauck and Michael Kirsche. (2012). Bonjour Contiki: A Case Study of a DNS-Based Discovery Service for the Internet of Things. *International Conference on Ad-Hoc Networks and Wireless, 7*(1) 45-55.

Soma Bandyopadhyay, Abhijan Bhattacharyya. (2013). Lightweight Internet Protocols for Web Enablement of Sensors using Constrained. *14*(2), 91-102.

Soma Bandyopadhyay, Abhijan Bhattacharyya. (2013). Lightweight Internet Protocols for Web Enablement of Sensors using Constrained. *International Conference on Computing, Networking and Communications, Workshops Cyber Physical System, 4*(2), 148 - 156.

Sotirios Kontogiannis, Angelos Chatzimparmpas, George Kokkonis. (2015). Middleware IoT protocols performance evaluation for carrying out clustered data. *IEEE, 16*(6), 35-47.

Sreeraj S, Suresh Kumar N, G Santhosh Kumar. (2017). A Framework for predicting the performance of IoT protocols, a Use Case based approach. *IEEE, 10*(1), 577 - 585.

Stefan Mijovic, Erion Shehu, Chiara Buratti. (2016). Comparing Application Layer Protocols for the Internet of Things protocols via Experimentation. *International Forum on Research and Technologies for Society and Industry Leveraging a better tomorrow, 2*(1), 152-157.

Turowski, K., Bosse, S., Kubela, J., & Pohl, M. (2018). Performance Evaluation of Application Layer. *IEEE, 2*(1), 31-36.

Valera, Alvin Cerdena; Tan, Hwee Xian. (2014). Performance evaluation of MQTT and CoAP via a common middleware. *IEEE, 13*(2), 36-40.

Vasileios Karagiannis1, Periklis Chatzimisios1, Francisco Vazquez-Gallego2, Jesus Alonso-Zarate2. (2015). A Survey on Application Layer Protocols for the Internet of Things. *Vasileios Karagiannis1, Periklis Chatzimisios1, Francisco Vazquez-Gallego2, Jesus Alonso-Zarate2, 1*(1), 1-10.

Vivek Hareshbhai Puar, Chintan M. Bhatt, Duong Minh Hoang, and Dac-Nhuong Le. (2018). Communication in Internet of Things. *Information Systems Design and Intelligent Applications, Advances in Intelligent Systems and Computing 672,, 5*(2), 273- 282.

Z. Shelby, K. Hartke, C. Bormann. (2014). The Constrained Application Protocol (CoAP). *Internet Engineering Task Force, 5*(1), 27-36.

Zoran Babovic, Jelica Protic,. (2016). Web Performance Evaluation for Internet of Things Applications. *IEEE, 6*(1), 194-203.

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı : NADİR Yaseen

Uyruğu :

Eğitim Bilgileri

Derece	**Eğitim Birimi**	**Mezuniyet Tarihi**
Yüksek Lisans	Gazi Üniversitesi / Bilişim Sistemleri	Devam ediyor
Lisans	Limkokwing Üniversitesi / Bilişim İletişim Teknoloji	2016
Lise	Dar Al-hadith	2011

Yabancı Diller

Rohyngca, Arapça, İngilizce ve Türkçe

Yayınlar

1. Comparison of IoT Protocols Performance, International Journal of Innovative Science and Research Technology Volume 6, Issue 5, 256 – 260.

GAZİLİ OLMAK AYRICALIKTIR…