NESNELERİN İNTERNETİNDE COAP PROTOKOLÜ İLE KABLOSUZ ALGILAYICI AĞLARIN GÜVENLİĞİNİN SAĞLANMASI

Uğur KANTEKİN

YÜKSEK LİSANS TEZİ

Bilgisayar Mühendisliği Ana Bilim Dalı
Tezli Yüksek Lisans Programı

Danışman: Dr. Öğr. Üyesi Serap ÇEKLİ

İstanbul
T.C. Maltepe Üniversitesi
Fen Bilimleri Enstitüsü
Haziran 2018

JÜRİ VE ENSTİTÜ ONAYI

Uğur KANTEKİN'in " Nesnelerin İnternetinde Coap Protokolü İle Kablosuz Algılayıcı Ağların Güvenliğinin Sağlanması" başlıklı tezi 22/6/2018 tarihinde aşağıdaki jüri tarafından değerlendirilerek "Maltepe Üniversitesi Lisansüstü Eğitim ve Öğretim Yönetmeliği"nin ilgili maddeleri uyarınca, Bilgisayar Mühendisliği Anabilim Dalında Yüksek Lisans/Doktora tezi **oy birliğiyle** / ~~**oy çokluğuyla**~~ olarak kabul edilmiştir.

Unvanı. Adı ve Soyadı	İmza
Üye (Tez Danışmanı) : Dr. Öğr. Üyesi Serap ÇEKLİ	.....
Üye : Prof. Dr. Mesut RAZBONYALI	.....
Üye : Dr. Öğr. Üyesi Ali AKMAN	.....

Prof.Dr. İlter BÜYÜKDIĞAN

Enstitü Müdürü

maltepe üniversitesi	ETİK İLKE VE KURALLARA UYUM BEYANI	Doküman No	FR-178
		İlk Yayın Tarihi	01.03.2018
		Revizyon Tarihi	
		Revizyon No	00
		Sayfa	1/1

Revizyon Takip Tablosu

REVİZYON NO	TARİH	AÇIKLAMA
00	01.03.2018	İlk yayın.

ETİK İLKE VE KURALLARA UYUM BEYANI

22/06/2018

Bu tezin bana ait, özgün bir çalışma olduğunu; çalışmamın hazırlık, veri toplama, analiz ve bilgilerin sunumu olmak üzere tüm aşamalarından bilimsel etik ilke ve kurallara uygun davrandığımı; bu çalışma kapsamında elde edilmeyen tüm veri ve bilgiler için kaynak gösterdiğimi ve bu kaynaklara kaynakçada yer verdiğimi; çalışmanın Maltepe Üniversitesinde kullanılan "bilimsel intihal tespit programı" ile tarandığını ve öngörülen standartları karşıladığını beyan ederim.

Herhangi bir zamanda, çalışmamla ilgili yaptığım bu beyana aykırı bir durumun saptanması durumunda, ortaya çıkacak tüm ahlaki ve hukuki sonuçlara razı olduğumu bildiririm.

Uğur KANTEKİN

TEŞEKKÜR

Öncelikle tez çalışmalarımın her aşamasında fikirleriyle bana yol gösteren ve tez danışmanlığımı üstlenen değerli hocam Dr. Öğr. Üyesi Serap ÇEKLİ' ye teşekkürü borç bilirim. Ericsson Araştırma Geliştirme ve Bilişim Hizmetleri A.Ş ailesine tez konusundaki yönlendirmelerinden, yardımlarından ve gösterdikleri anlayıştan dolayı teşekkür ederim. Çalışmamın her aşamasında yanımda olup bana moral veren değerli aileme sonsuz teşekkürlerimi sunarım.

Uğur Kantekin

Haziran 2018

ÖZ

NESNELERİN İNTERNETİNDE COAP PROTOKOLÜ İLE KABLOSUZ ALGILAYICI AĞLARIN GÜVENLİĞİNİN SAĞLANMASI

Uğur KANTEKİN
Yüksek Lisans
Bilgisayar Mühendisliği Ana Bilim Dalı
Danışman: Dr. Öğr. Üyesi Serap ÇEKLİ
Maltepe Üniversitesi Fen Bilimleri Enstitüsü, 2018

Günümüzde Nesnelerin İnterneti (IoT) teknolojisinin günlük hayatımızda kullanımı gittikçe yaygınlaşmaktadır. Kablosuz Algılayıcı Ağ (KAA) aygıtların çeşitliliği, düğüm sayısı, uygulama ortamlarının kısıtlı, kısa mesafeli vericiye ve düşük kapasiteli donanıma sahip olması gibi nedenlerden dolayı birçok saldırıya karşı korunaksızdır. Bu çalışma kapsamında, KAA düğümlerinin arasında güvenli haberleşmeyi sağlamak amacıyla kullanılan haberleşme protokolü Kısıtlayıcı Ağ Haberleşme Protokolü (CoAP) ve Veri Bloğu Aktarım Katmanı Güvenliği (DTLS) protokolü birlikte kullanılarak haberleşme modeli güvenli CoAP (CoAPs) tasarlanmıştır. Ayrıca DTLS protokolü başlık bilgisi sıkıştırılarak ve kimlik doğrulama süreleri kısaltılmasıyla protokolün daha performanslı çalışması sağlanmıştır. Algılayıcı düğümlerden alınan sıcaklık ve nem gibi çevresel faktörlerden elde edilen veriler DTLS protokolü aracılığıyla 128 bit şifrelenerek ana istasyondaki düğüme gönderilmiştir. Bu çalışmayla, gerçekleştirilen güvenlik sistemi ile DTLS ve CoAP protokolü arasındaki tümleştirme işlemi benzetim ortamında gösterilmiştir. Bu çözümün IoT içinde uygulanan diğer haberleşme protokollerine göre karşılaştırılmıştır ve başarım sonuçları sunulmuştur.

Anahtar Sözcükler: Nesnelerin İnterneti, CoAP, DTLS, CoAPs

ABSTRACT

WIRELESS SENSOR NETWORK SECURITY WITH COAP PROTOCOL FOR INTERNET OF THINGS

Uğur KANTEKİN
Master
Thesis Advisor: Dr. Öğr. Üyesi Serap CEKLİ
Maltepe University Graduate School of Science Engineering, 2018

Nowadays, the use of Internet of Things (IoT) technology becomes increasingly prevalent in our daily life. The wireless sensor networks (WSN) are vulnarable to many attacks because of the device variety, number of nodes, restricted application environment which have short range transiver and low capacity hardware. In this study, the secure CoAP (CoAPs) as a communication model is designed by using the Datagram Transmission Layer Security (DTLS) protocol with Constrained Application Protocol (CoAP) which is used to provide the secure communication of WSN nodes. Moreover, the identity verification time of the DTLS protocol has been decreased providing that header part has been compressed therefore the protocol operates with higher performance. The sensor node data which is obtained from the environmental factors such as temperature and humidity has been send to the sink node as the base station with the help of the DTLS protocol by using 128-bit encryption. In this study, the integrity process of the implemeted security system with the DTLS and CoAP protocols is showed in the simulation environment. The implemented security solution has been compared with the other communication protocols which are applicable to IoT and the performance results are presented.

Keywords: Internet of Things, CoAP, DTLS, CoAPs

İÇİNDEKİLER

ÇİZELGELER LİSTESİ

ŞEKİLLER LİSTESİ

KISALTMALAR

ACK	Kabul edilebilir	Acknowledgement
AMQP	İleri Düzey Mesaj Dizisi Protokolü	Advanced Message Queuing Protocol
CoAP	Kısıtlayıcı Uygulama Protokolü	Constrained Application Protocol
CoAPs	Güvenli Kısıtlayıcı Uygulama Protokolü	Constrained Application Protocol Secure
CON	Onaylanabilir	Confirmable
DTLS	Veri Bloğu Aktarım Katmanı Güvenliği	Datagram Transport Layer Security
DoS	Servis Reddi	Denial of Service
DDS	Veri Dağıtım Sistemi	Data Distribution System
HTTP	Hiper-Metin Transfer Protokolü	Hypertext Transfer Protocol
IPSec	İnternet Protokol Güvenliği	Internet Protocol Security
IoT	Nesnelerin İnterneti	Internet of Things
JSON	JavaScript Nesne Biçimi	Javascript Object Notation
IDS	Saldırı Tespit Sistemi	Intrusion Detection System
IPsec	İnternet Protokol Güvenliği	Internet Protocol Security
IETF	İnternet Mühendisliği Görev Gücü	Internet Engineering Task Force
IPv4	İnternet Protokolü Sürüm 4	Internet Protocol Version 4
IPv6	İnternet Protokolü Sürüm 6	Internet Protocol Version 6
KAA	Kablosuz Algılayıcı Ağ	Wireless Sensör Network
MAC	Mesaj Doğrulama Kodu	Message Authentication Code
M2M	Makineler Arası İletişim	Machine To Machine
MQTT	Telemetri Mesaj İletim Protokolü	Message Queue Telemetry Transport
MMS	Maksimum Bölüm Boyutu	Maximum Segment Size
PKI	Açık Anahtar Altyapısı	Public Key Infrastructure
RST	Sıfırlama	Reset
SSL	Güvenli Soket Katmanı	Secure Socket Layer
UDP	Kullanıcı Veribloğu İletişim Kuralları	User Datagram Protokol
URI	Tekdüzen Kaynak Tanımlayıcısı	Uniform Resourse Identifier
TCP	Taşıma Kontrol Protokolü	Transmission Control Protocol
TLS	Taşıma Katmanı Güvenliği	Transport Later Security
6LoWPAN	Düşük Güçlü Kablosuz Kişisel Alan Ağları	Low Power Wireless Personal Area Networks

ÖZGEÇMİŞ

Uğur Kantekin

Bilgisayar Mühendisliği Anabilim Dalı

Eğitim

Derece	*Yıl*	*Üniversite, Enstitü, Anabilim*
Ls.	2015	Maltepe Üniversitesi Bilgisayar Mühendisliği (İngilizce)
Lise	2009	Tuzla Anadolu Teknik Lisesi

İş/İstihdam

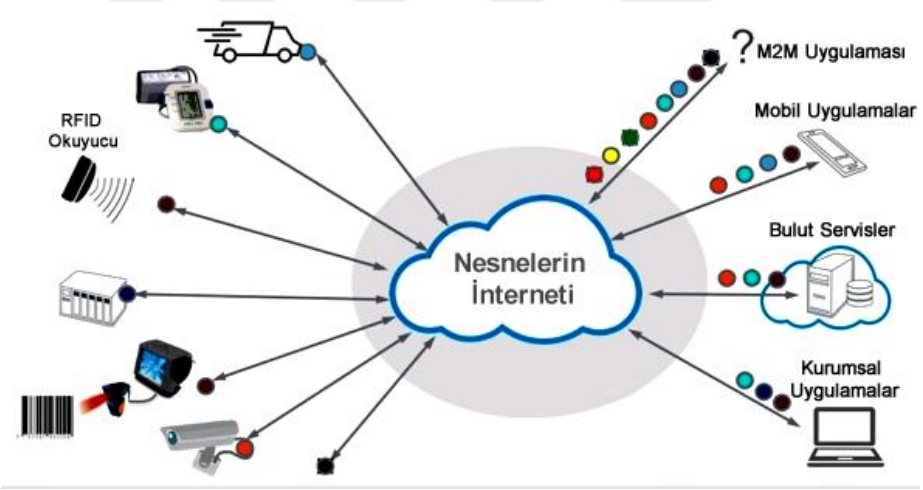
Yıl	*Görev*
2010 - 2011	Web Geliştiricisi - İnteramedya İletişim Grubu
2015 -	Yazılım Mühendisi - Ericsson Türkiye

Kişisel Bilgiler

Doğum yeri ve yılı	: Şişli/İstanbul 1990	Cinsiyet: Erkek
Yabancı diller	: İngilizce	
GSM / e-posta	: 0530 841 2385 / ugur@ugurkantekin.com	

BÖLÜM 1. GİRİŞ

Kablosuz Algılayıcı Ağ (KAA) ortamlarının gelişmesiyle beraber gelen düşük maliyetli algılayıcı düğümlerin güvenliği yeni araştırma alanı oluşturmuştur. Bu ağlar sıcaklık, ses, titreşim, basınç, hareket veya hava kirliliği gibi fiziksel veya çevresel koşulları izlemek ve ortaklaşa verilerini ağ üzerinden ana istasyondaki düğüme iletmek için konum olarak dağıtılan özerk algılayıcılardan oluşur. Her bir KAA düğümü, izlenecek alanlara rastgele veya düzenli aralıklarla bırakılarak çeşitli ölçüm bilgileri toplayabilmektedir. KAA uygulama alanları en çok askeri, sağlık ve akıllı şehir uygulamaları, endüstriyel üretim süreçlerinin izlenmesi ve sağlık izleme sistemlerinde görülmektedir.



Şekil 1.1. Nesnelerin İnterneti (IoT)'de Güvenlik

KAA'lar her bir düğümün bir algılayıcıya bağlı olduğu yüzlerce veya binlerce düğümden oluşmaktadır. Bu özellikleri bakımından düşük maliyetli, kablosuz haberleşme ortamı, kaynak kısıtları, çalışma zamanındaki işlem kapasitesi, algılayıcı aygıt sayısının fazlalığı, enerji verimliliği, konumlandırma ihtiyacı, gerçeklenen ortamın eski olması ve maliyet yüksek olması gibi nedenlerden dolayı KAA birden fazla güvenlik sorunuyla başa çıkmak zorun kalmıştır. Bu ağlar saldırılara karşı korunaksız, çok düşük kapasiteli ve kısa mesafeli vericiye sahiptir. IoT' de düşük güç tüketimi ile çalışan

cihazlar (6LowPAN) için mevcutta bulunan pek çok haberleşme protokolü bulunmaktadır.

Makinalar arası iletişim (M2M) uygulamalarının kullanımı ile kimi zaman birden fazla protokol aynı anda kullanılmak zorunda kalınmıştır. KAA düğümlerinin haberleşmesinde kullanılan protokol güvenliğinin sağlanması konusunda ciddi riskler bulunmaktadır. Dolasıyla, tercih edilen protokollerin yanında ortamın koşuluna göre ek güvenlik duvarı veya Saldırı Tespit Sistemi (IDS) kullanılması gerekmektedir. Uygulamaların kullanılabilmesi için yalnız benzetim ortamında kalmaması, algılayıcı düğümler üzerinde de uygulamasının yapılması gerekmektedir. Son yıllarda IoT platformlarının gelişmesi ile birlikte yapılan araştırmalar internet üzerinden haberleşmeyi sağlayan protokol tasarım örnekleri günden güne artmaktadır. Tez kapsamında incelenen yapıya benzer olan literatürdeki çalışmalar aşağıdaki gibidir.

Florida International üniversitesinden Mahmudur Rahman, Bogdan Carbunar ve Umut fitness uygulama çalışmalarında kullanılan adım sayar, kalp ritmi takip cihazı, kronometre ve nem ölçer gibi cihazlardan anlık değerleri toplayarak IoT cihazı vasıtasıyla uzaktaki bir web sunucusuna kriptolu şekilde iletmektedir. Bu çalışmada şifreleme algoritması olarak FirstBeat, IoT cihazı olarak ise Arduino cihazı kullanılmıştır. Ayrıca web sunucudan IoT cihazına ait olan koordinat verileri alınarak bulunduğumuz noktayı GPS vasıtası ile izlenebilmektedir [1].

Berkeley üniversitesinden Chris Karlof 2014 yılında araştırmalarda genellikle fiziksel katmanda kullanılan protokollerinden TinySec ve MiniSec güvenlik çözümlerinden algılayıcı düğüm üzerinde uygulanmıştır. IEEE 802.15.4 ise KAA için geliştirilmiş olmasına rağmen düşük güç tüketimi, düşük maliyet ve esnek oluşundan dolayı KAA'da kullanılmaktadır. Diğer güvenlik protokolleri düğüm üzerine uygulanmamıştır. TinySec ve MiniSec veri gizliliğini garanti etmek için 80 bit anahtar boyutlu Skipjack algoritmasını kullanmıştır. Yapılan araştırmalar göstermektedir ki veri gizliliği için anahtar boyutunun en az 128 bitlik olması gerekmektedir. TinySec mesaj tekrar yayınlama ataklarını önleyemezken, MiniSec'te verinin bütünlüğü garanti edilememektedir. Ayrıca bu protokoller KAA için güvenlik gereksinimlerinden yeniden kullanılabilirlik ilkesini sağlayamamaktadır. Kullanılabilirlik gereksiniminin

karşılanmaması demek, o protokolün DoS saldırılarına karşı dayanıksız olması anlamına gelmektedir [2].

KTH Teknoloji Enstitüsünden Rikard Höglund 2014 yılında yapılan çalışmada kısıtlı kapasiteli cihazlarda kullanılacak CoAP için düşük seviyeli bir kimlik doğrulama uzantısı oluşturulmuştur. Kısa Mesaj olarak adlandırılan bu uzantı Kimlik Doğrulama (SMACK), güvenli bir yöntem gerektiren cihazlarda kullanılabilir uzantının temel amacı cihazların sadece sınırlı güç kullanırken mesajların doğrulanması düşük batarya sahip düğümler ve uyku saldırılarında karşı koruma sağlanmıştır [3].

Luleå üniversitesinden James King 2015 yılında gerçekleştirdiği çalışmasında yerel ağ içerisinde ayrı bir IoT ağı oluşturmuştur. IoT arayüzü olarak ise Arduino aygıtı kullanılmış ve çeşitli medikal cihazlar ve algılayıcılardan alınan veriler toplanarak şifrelemiştir. Şifreleme algoritması olarak AES-128 ve AES-256 algoritmalarını kullanmıştır. Arduino cihazı üzerinden şifrelenen verileri yerel ağ üzerinde anlık olarak izlenebilmektedir. WAN üzerinde ise belirli bir server'ın erişmesi için NAT bağlantısı açılmıştır. Bu sayede anlık veriler server üzerinde tutularak veri analizinin yapılması sağlanmıştır [4].

Bannari Amman Teknoloji enstitüsünden M.Suresh havaalanları için yapılan park alanına algılayıcı yerleştirmiş ve bu algılayıcılardan aldığı verileri Arduino yardımı ile okumuştur. Arduino aygıtının ethernet portu üzerinden bilgisayar ile paylaşmıştır Ethernet üzerinden paylaşılan verilerde HTTP protokolü kullanılmıştır. Verilerin yerel bilgisayarın web arayüzünde izlenebilmesi için Arduino programı üzerinde görülen web arayüzü tasarlanmıştır. Ayrıca kullanıcıların kendi kullanıcı ve şifreyi ile bağlanabilmesinin sağlayan kimlik denetimi oluşturulmuştur [5].

Bu çalışmada amaç, KAA' ların paket güvenliğini sağlamak ve dışarıdaki diğer ağlardan gelen isteklerde veriye doğrudan ulaşılmasını engellemektir. Bunu yaparken düşük maliyetli ve hızlı biçimde verileri almanın yöntemi olan Kısıtlayıcı Uygulama Protokolü (CoAP) protokolü ile birlikte Güvenli Veri Bloğu Aktarım Katmanı (DTLS) gelen isteklere sertifikasyon işlemini uygulayarak verilerin güvenli bir şekilde iletilmesinin sağlanması için bir yöntem önerilmiştir. Böylece sertifikasyon bilgisine sahip olmayan diğer ağlardaki istemcilerin algılayıcılara erişimi engellenmiştir. Bu çözümlerin her biri önerilen güvenlik çözümleri benzetim ortamlarında uygulanmıştır ve gerçek donanım üzerinde bir IoT kurulumunda denenmiştir.

İkinci bölümde, (CoAP) kullanılarak IoT' de haberleşmesinde bir ağ oluşturulması, CoAP protokolünün yapısının incelenmesi, gerçek KAA aygıtlarında uygulanması ve üzerinde Contiki işletim sistemi üzerindeki COOJA benzetim ortamını kullanarak algılayıcı düğümler arasındaki haberleşmenin CoAP protokolü ile gerçekleştiği gösterilmiştir.

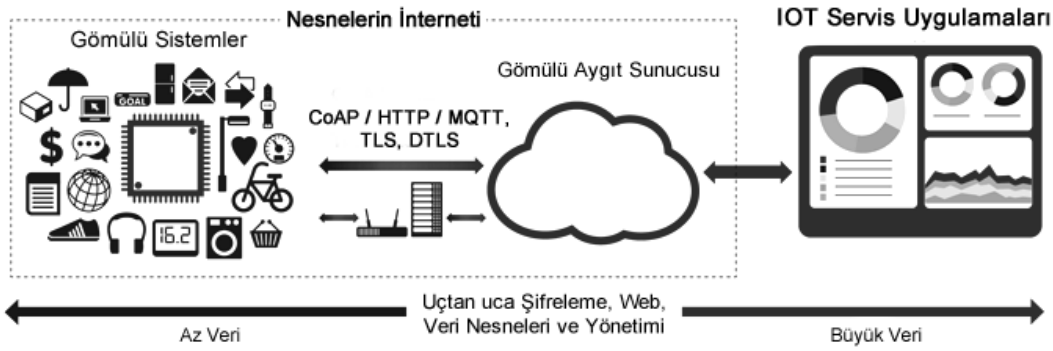
Üçüncü bölümde, KAA düğümleri arasında güvenli iletişim kurarken haberleşme maliyetini düşürmek amacıyla DTLS sıkıştırma yöntemi geliştirilmiştir. Benzetim ortamında ve denemler sonucunda, sıkıştırılmış DTLS ile KAA aygıtların oturumlarının daha kısa ve hızlı olduğu görülmüştür. Böylece güvenli paket transferi sağlanırken iletim süresinin düşürülebilmesinin mümkün olduğu görülmüştür.

Son bölümde ise Contiki işletim sistemini kullanarak Cooja benzetim ortamı üzerinde CoAP ve DTLS tümleştirme işlemi gösterilmiştir. Geliştirilen CoAPs ve CoAP protokollerinin haberleşme performans hızı sonuçları sunulmuştur.

BÖLÜM 2. YÖNTEM

IoT' deki aygıtların birçoğu birbiriyle iletişim halinde olduklarından dolayı her zaman saldırıya açıktır. Kablosuz İnternet güvenliği karşılıklı veri alışverişi yaparken, bu tip bağlantılarda güvenlik halen sorun olmaktadır. Birden fazla bağlı nesneye sahip olan akıllı ev veya büyük bir üretim firma organizasyonunda, bu zorluklar daha büyük olacaktır. Güvenlik güncellemesi nesnelere uygulanırken verimsiz bir plan, IoT sisteminde en büyük problem olacaktır.

Bütün sistemlerin şifrelerinde savunmasız kısımlar ortaya çıkabilmektedir. Ancak geliştirme süreçlerinde güvenliği ve eksiksiz bir güvenlik yaşam döngüsüne sahip algoritmalar kullanıldığında daha az güvenlik sorunu ortaya çıkacaktır. Bununla beraber, tüm yazılım firmaları, kendi kullanıcılarını korumak amacıyla güvenlik eklentileri piyasaya sürmek ve güvenliği riskli kısımlara hızlı müdahale etmek için hazır olmalıdır. Kablosuz ağ sistemine bağlı nesnelerin güvenli bir şekilde tasarlanması, IoT ekosisteminin başarısı için çok önemlidir. Burada, bağlı cihazları üreten şirketlerin ilk önceliği güvenlik değildir. Aygıtları geliştiren mühendislerin, tasarım ve nesnelerin birbiriyle iletişim kurabilme çalışmalarına öncelik vermesi nedeniyle, güvenlik ikinci planda kalabilmektedir.



Şekil 2.1. Nesnelerin İnterneti (IoT)'de Güvenlik

Birbirine bağlı nesnelerin güvenlik açığının olmasının diğer nedeni ise aygıtların üretim maliyeti gelmektedir. Bu aygıtların güvenli bir şekle dönüştürmek üreticilerin ve yeni tasarlanılacak güvenlik sistemlerinin geliştirilmesi, güvenli aygıtların bakımını sağlamak için yeni eklenti yazılımlarının oluşturulması ve güvenli olmayan ağlar için sızıntı testi yapılması gibi birden fazla adımdan geçmesi gereklidir. Bu nedenlerle, bazı üreticiler güvenlikten tamamıyla vazgeçebilirler.

IoT altyapısını oluşturan yapıların kontrol sistemlerinde ölçeklenebilirlik ve hız açısından performans ihtiyacı duymaktadır. Her bir nesnenin internet ortamında tanınabilmesi, sadece ona ait bir internet adresinin olmasını gerektirir. Çizelge 2.1.' de görüldüğü üzere IPv4 formatının yakın gelecekte adresleme kapasitesi dolacağı için IP adresleme düzeni IPv4'den IPv6 geçilmiştir [6].

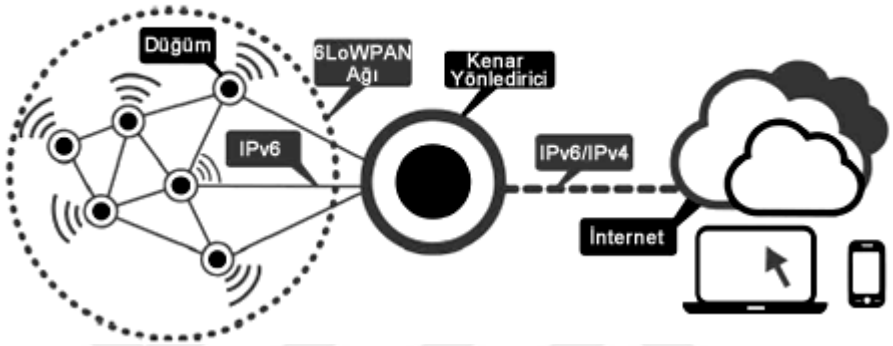
Çizelge 2.1. IPv6 ile IPv4 protokoller arasındaki fark

Özellik	InternetProtokol Sürümü 4 (IPV4)	Internet Protokol Sürümü 6 (IPV6)
Tarih	1981	1999
Kaynak ve Hedef Adresleri	32 bit (4 Bayt)	128 bit (16 Bayt)
IPSec Desteği	İsteğe bağlı	Zorunlu
Adres Formatı	192.149.2525.76	2041:0000:130F:0000: 0000:07C0:853A:140B
Yazım Kuralı	192.149.0.0/24	2041:0000:130F:0000::/48
Adresleme Kapasitesi	2^{32} = ~4,294,967,296	2^{128} = ~340,282,366, 920,938,463,463,374, 607,431,768,211,456

IoT haberleşme protokolleri temel olarak iki kategoride toplanmaktadır [7].

- İstemci/Sunucu (Client / Server)
- Yayınla/Abone ol (Publish / Subcribe)

İstemci/Sunucu yönteminde İstemci, Sunucuya bağlantı isteğinde bulunur. İstenilen verilerin tamamı Sunucu'da olduğu için İstemci tarafından yapılan isteğe yanıt verir. Örneğin; İstemci algılayıcı düğümden herhangi bir nem veya sıcaklık bilgisi okumak istediğinde Sunucu istemcinin önceden var olup olmadığını ve aygıtın adres bilgisini öğrenmesi gereklidir. Şekil 2.2.' de verildiği gibi IPv6 ile KAA'ların haberleşmesi için İstemci / Sunucu yöntemine ait gösterim verilmiştir.



Şekil 2.2. Nesnelerin İnterneti (IoT)'de Güvenlik

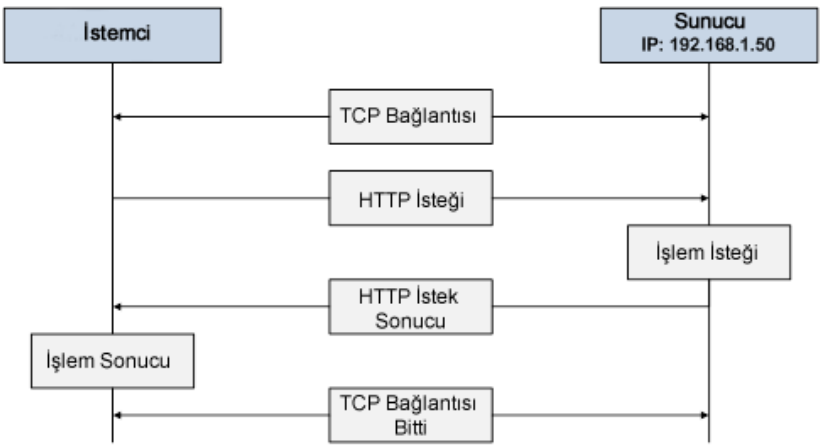
Yayınla/Abone ol yönteminde ise arada bir arabuluculuk vardır ve konu çerçevesinde arabuluculuk görevini üstlenmektedir. İstemciler (tüketici) Sunucu' dan gelen verileri üye oldukları konu çerçevesinde aradaki arabuluculuk üzerinden okuyabilmektedir. Örneğin; bir cihaz sıcaklığı her dakika örnekleyebilmekte ve ölçüm sonuçlarını her saat başında yayınlayabilmektedir. Bu bilgiyi kullanacak olanlar arabulucu düğüm üzerinden okurlar ve veriyi üreten ile doğrudan ilişki kuramazlar. Eğer İstemci/Sunucu yönteminde haberleşme altyapısı detaylı analiz edilmişse çok iyi sonuç verir. Sunucunun bir IP adresi bulunur ve ilgili bir portu dinleyebilir. İstemci bu porta bağlanarak istekte bulunabilir. Yayınla/Abone ol yöntemi alt yapıda belirsizlikler var ise daha iyi sonuç verebilir.

Örneğin; Arabulucu mesafesi aralığı büyük düğümün ve aygıtın ağ adresi yenileniyor ise veya sürekli ağa bağlı değilse sadece bu gibi durumlarda aradaki arabulucu iletişimin sürmesini yönetebilir. IoT destekli basit bir cihaz için belli bir adrese bağlanıp elindeki verileri yayınlanmak üzere ona aktarması kolaydır.

2.1. IOT Haberleşme Protokolleri ve Standartları

2.1.1. Hiper Metin Transfer Protokolü (HTTP)

İstemciler ile bilginin sağlandığı sunucular arasında bir bağlantı oluşturan HTTP (Hiper Metin Transfer Protokolü) aynı zamanda bu kaynaklar arasındaki bilgi alışverişinin kurallarını belirleyen bir dildir. Bu kuralların ilki, verinin güvenli bir şekilde aktarımıdır. Gönderilen mesajların gizli olması gerektiği için Şekil 2.3' de görüldüğü gibi TCP altyapısını kullanarak gönderilir. Aktarılan veriler sadece dosyayı içermesi gerekmediği için ismi hiper metin olarak oluşturulmuştur. Bundan dolayı İstemci ile Sunucu arasında bilgi alışverişi olmadan önce TCP bağlantısı kurulması sağlanabilmiştir [8]. Web siteleri büyük küçük birçok dosyadan oluşmaktadır ve bir istek geldiğinde bu dosyaların hızlıca aktarılması gerekmektedir. Bir web tarayıcısı bir web sayfasını istediğinde, web sunucusuna bir istek gönderilir. Sonra web sunucusu da bu istek mesajına bir cevap gönderir. Bu mesajların (hem istek hem cevap) birer başlık ve birer gövde kısmı bulunmaktadır. Başlık kısmında mesajla ilgili meta bilgiler, gövde kısmında ise mesajın içeriği ile ilgili bilgiler yer almaktadır. HTTP bir istek/cevap protokolü olduğu için ve her cevaptan önce bir istek olacak şekilde tasarlanmıştır.

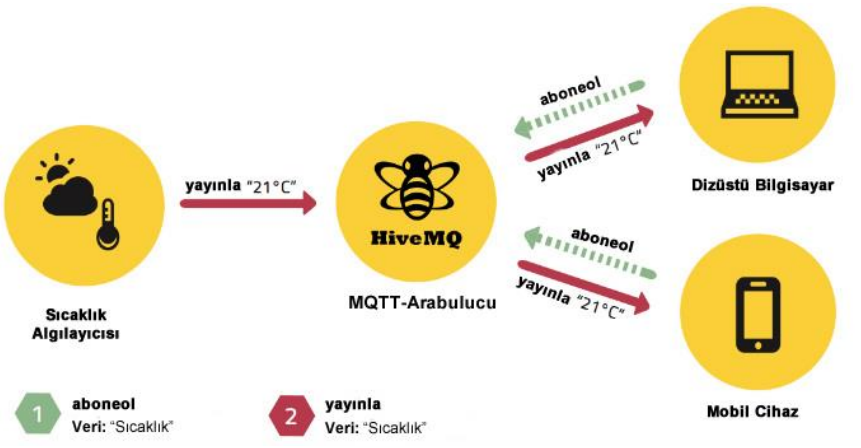


Şekil 2.3. HTTP protokolü çalışma mimarisi

2.1.2. Telemetri Mesaj İletim Protokolü (MQTT)

Telemetri Mesaj İletim protokolü Şekil 2.4 ‘de görüldüğü gibi yayınlama ve abone olma mantığına dayanan bir yöntemle çalışmaktadır. Daha çok uzak haberleşme ağlarındaki cihazların Merkezi Denetleme Kontrol ve Veri Toplama **(**SCADA) sistemlerinde algılayıcı düğümlerden verileri toplamak için geliştirilmiştir. MQTT protokolü HTTP’ ye benzer bir şekilde JSON ve ikili formatta veri içerebilmektedir. MQTT protokolü HTTP kadar yaygın olmasa da akıllı IoT uygulamalarında kullanım oranı artmaktadır ve arabulucu tabanlı projeler de geliştirilmektedir. Dolasıyla HTTP ve MQTT protokolleri birçok IoT ekosisteminde kullanımı gittikçe genişlemiştir.

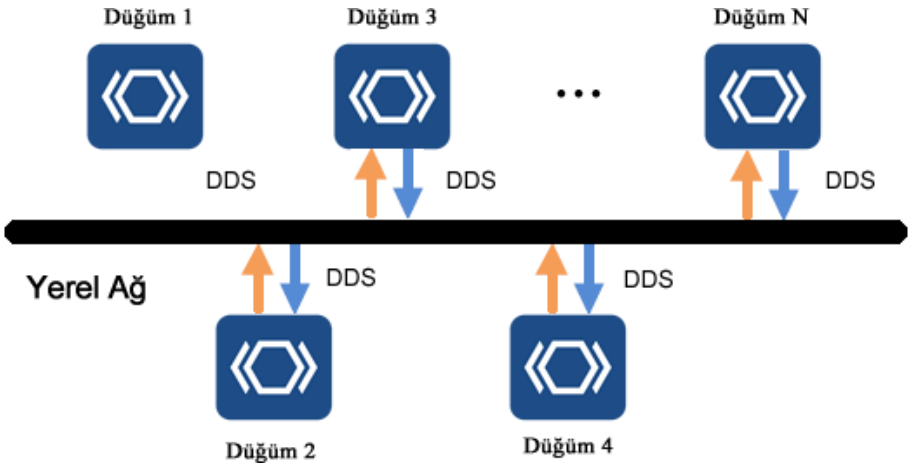
Bu protokolü, ortamdaki istemci ve sunucuların altyapısal yetersizliği ve uygulanacak ortamla ilgili altyapı bilgisi az ise kullanımı önerilmelidir. Aynı zamanda verilerin yayınlanabileceği bir MQTT arabulucusunun bulunması gerekmektedir. MQTT güvenli bir haberleşme yolu üzerinden Güvenli Taşıma Katmanı (TLS) çalışmaktadır. Aktarım katmanı protokolü kısmında MQTT TCP'yi kullanır ve dolayısıyla bağlantının tüm gerekli olduğu senaryolar için mükemmeldir. MQTT, CoAP ile karşılaştırıldığında daha yavaş bir gönderme döngüsüne sahiptir. Çoğu IoT uygulamasında MQTT üzerinden HTTP ile doğrudan tümleştirme yapılabilir veya veri tabanına doğrudan istediğiniz verileri gönderebilmesi için esnek yapıya sahiptir [9,10,11].



Şekil 2.4. MQTT protokolü çalışma mimarisi

2.1.3. Veri Dağıtım Hizmet Protokolü (DDS)

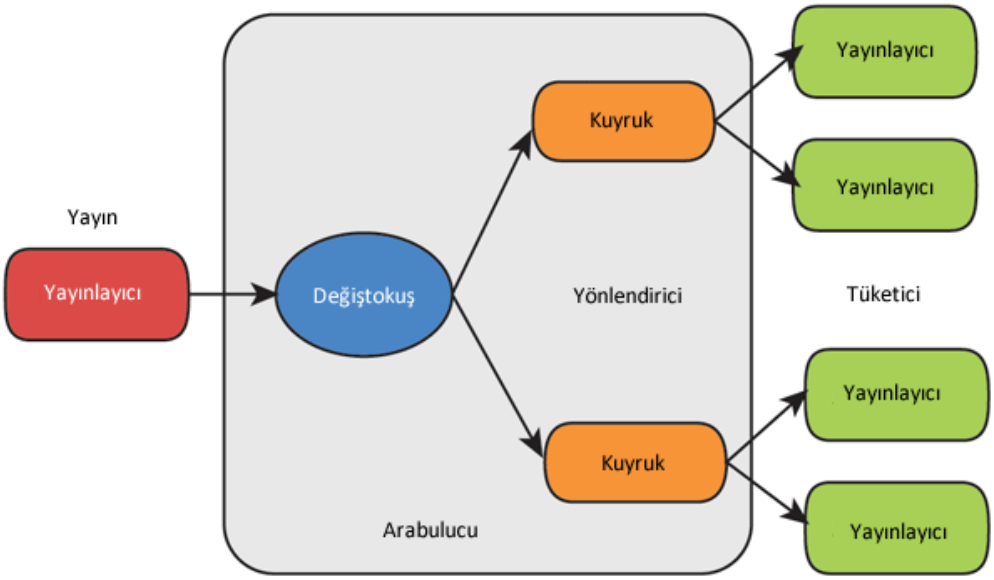
Veri Dağıtım Hizmet Protokolü (DDS), Nesne Yönetim Grubu (OMG) tarafından tasarlanmış bir ara katman protokoldür. DDS protokolü de yayınla/abone ol mimarisini desteklemektedir ve açık kaynaklı bir standarttır [12]. MQTT'den farkı ortada arabuluculuk rolüne ihtiyacı yoktur ve dağıtık şekile sahip mimaridir. Bu protokolü düğümler arasındaki haberleşme kurarken UDP (Kullanıcı Veri bloğu İletişim Protokolü) altyapısını kullanır [13]. Bu protokol IoT'de daha etkin iletişimi sağlamak amacıyla merkezi sistemden yönetimi ortadan kaldırmıştır. Böylece milisaniye mertebesindeki sürelerin altında haberleşebilmektedir. Bu yüksek hız kapasitesi nedeniyle DDS ondaki protokolünü M2M haberleşme ihtiyaçlarında kullanılmaktadır. Özellikle haberleşirken ana sunucu yerine, ağ üzerinde dağıtık bir şekildeki haberleşme yöntemini kullanır. Aşağıda Şekil 2.5'de protokolün sistem mimarisi yerel ağdaki düğümlerden bir tanesi çalışmaz veya bozulur ise normal durumda çalışan düğümler üzerinden iletişim kurabildiği görülmüştür.



Şekil 2.5. Dağıtık sistem mimarisi

2.1.4. İleri Düzey Mesaj Dizisi Protokolü (AMQP)

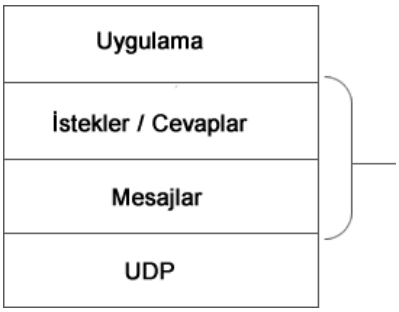
İleri Düzey Mesaj Dizisi Protokolü (AMQP) yayınlama/ abone ol protokol mantığıyla çalışmaktadır. Özellikle bankacılık sektöründe kullanılır ve diğer alanlarda kullanımı sınırlıdır. Haberleşme modelinde sağlamlık ve işlem türü haberleşmeyi desteklemesi onun en değerli özelliğidir. Temsili Durum Transferi (REST) iletişim yöntemini desteği yoktur. IoT haberleşmesinde kullanılan en yavaş çalışan protokollerden biridir ve alt yapısında TCP protokolünü kullanır [14]. Bunun nedeni AMQP protokolü mesajların kaybı olmamasına çok fazla önem vermesidir. IoT dünyasında belli bir yeri vardır ve farklı üstünlüklere ve zayıf yönlere sahiptir. Aşağıdaki Şekil 2.6' de çalışma sistemindeki gibi ağdaki iletilen mesajların tamamını takip etmeye çalışır, her bir iletilen paket bağımsızlık ilkesine göre iletilir, istenildiği şekilde eriştiğinden emin olmaya çalışan bir protokoldür. AMQP protokolü en çok veri merkezleri ile mesajlaşmada sistemlerinde kullanılmaktadır [15].



Şekil 2.6. İleri Düzey Mesaj Dizisi Protokolü (AMQP) çalışma mimarisi

2.2. Kısıtlayıcı Uygulama Protokolü (CoAP)

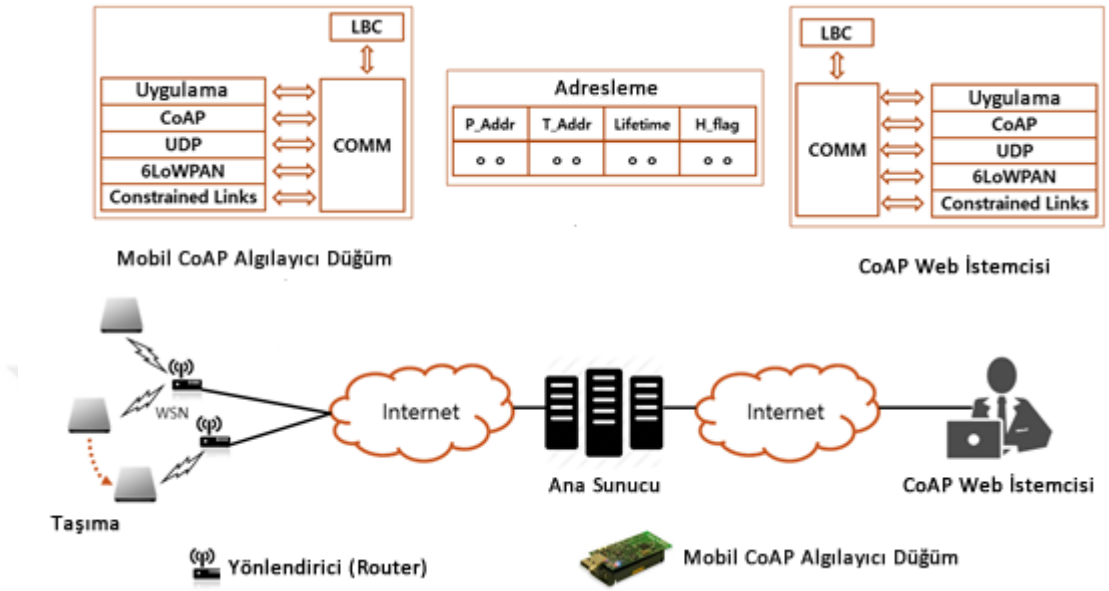
Kısıtlayıcı Uygulama Protokolü (CoAP), İnternet Mühendisliği Görev Gücü (IETF) standardı tarafından geliştirilmiş uygulama katmanı protokolüdür. Özellikle CoAP kısıtlı cihazlar ile birlikte kablosuz ağ algılayıcıları çalışmalarında 2014' te RFC 7252 şeklinde standart olarak kullanılmıştır [16]. Web teknolojilerinde kullanılan Temsili Durum Transferi (REST) servisleri akıllı telefonların özelliklerinden yoksun kısıtlı cihazlarla haberleşmede özel bir IoT protokolü gerektirir ve CoAP bunu destekleyen bir protokoldür. Temel REST komutlarını HTTP ile beraber kullanabilen (GET, DELETE, POST, PUT) metotlarını kullanabilen, basit bir yapıya sahip, CoAP İstemci ve CoAP Sunucu arasında en kolay veri haberleşme yoludur [17]. Aynı zamanda kısıtlı ve düşük kapasiteli veri tutabilen düğüm aygıtlar üzerinde ve bant genişliği düşük olan ağlarda çalışabilen CoAP, daha hızlı haberleşmek içinde UDP altyapısını kullanarak Şekil 2.7'de görüldüğü gibi çalışmaktadır. Mesajlaşma modelinde kullanılan servisler ve kaynakların bulunması için önceden belirlenmiş yerleşik konumda içindedir ve Tekdüzen Kaynak Tanımlayıcısı (URI) adresleme tipi gibi anahtar yapıyı barındırır [18]. CoAP'ın İstemci/Sunucu mesajlaşma modeli HTTP' nin İstemci/Sunucu modeline benzer ancak CoAP M2M çalıştığı için CoAP hem istemci hem de sunucu rolünü üstlenmiştir.



Şekil 2.7. CoAP protokol katmanları

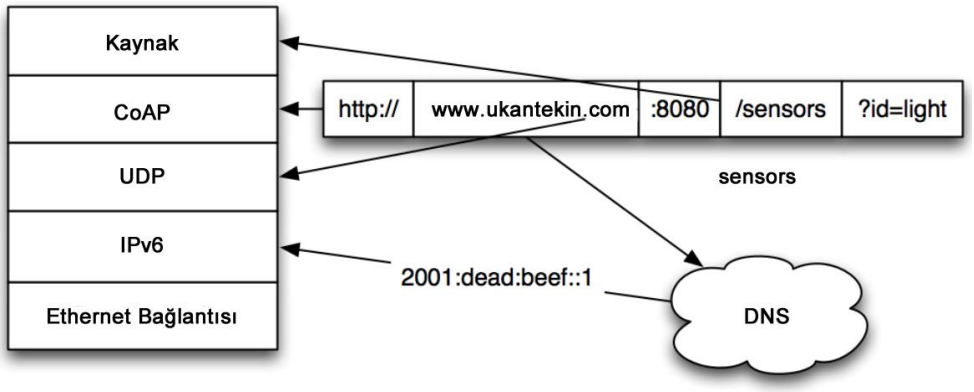
CoAP protokolü HTTP gibi durumsuz(stateless) çalışma mantığından farklı olarak asenkron olarak çağırma yapılarak gerçekleştirilir. Bu işlemdeki sürecin içinde seçenek doğruluğunu bulmak için mesajlar katmanlarının modelde mantıksal kullanımı ile gerçekleştirilebilir. IoT ile düşük kapasiteli cihazlar daha önemli bir hale gelmiştir. Bundan dolayı, var olan teknolojiler bu düşük kapasiteli cihazlara uyarlanmakta ve mümkün olmadığında yenileri geliştirilmektedir.

Kısıtlı donanıma aygıtlar için CoAP mimarisinin çalışma mantığında kullanılan Şekil 2.8' de düğümler arasında haberleşme sağlayan ana yönlendirici düğümler bulunmaktadır.



Şekil 2.8. CoAP mimarisi

Önbelleğe alma yerine CoAP vekil sunucusu ile CoAP cihazı arasında değiş tokuş edilen mesaj sayısını azaltmaya olanak tanımaktadır. Böylece bir Web uygulaması tarafından bir CoAP kaynağına erişmeye çalışıldığında Şekil 2.9'da görüldüğü gibi daha hızlı bir şekilde URI üzerinden erişilebilmektedir [19]. URI, URL'in işaret ettiği kaynak konumundan sonra gelen ilgili kaynağın ayırıcı adresini belirtir yani web üzerinde belli bir kaynağa (internet sitesi, belge, resim vb.) ulaşmak için kullanılan karakter dizisi ve metindir.



Çizelge 2.2. URI çalışma sistemi

2.2.1. Mesajlaşma Tipleri

CoAP protokolünde dört ana mesaj tipi vardır. Çizelge 2.2'deki gibi mesajlaşma bitleri Onaylanabilir (CON), Onaylanamaz (NON), Kabul edilebilir (ACK) ve Sıfırlama (RST). CON mesaj tipi karşılığında bir ACK mesajı alınması gereklidir. ACK mesaj bilgisi gelen yanıtla birlikte gönderilebileceği gibi ayrı ayrı olacak şekilde alınabilmektedir. NON mesajların karşılığında ACK gönderilmesine gerek yoktur. ACK mesajları CON kısmına denk gelicek şekilde gönderilebilmektedir [20].

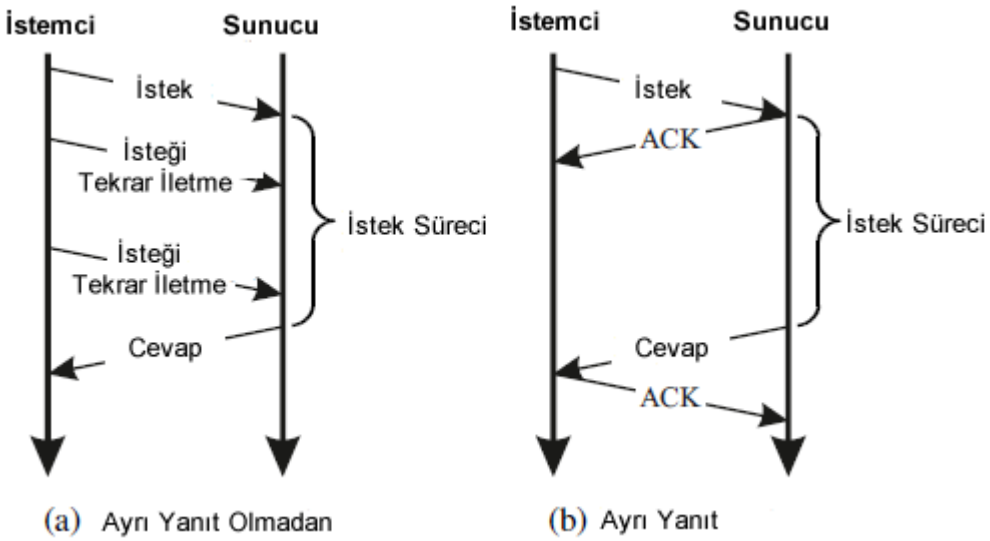
Çizelge 2.2. Mesajlaşma çalışma sistemi

CON	00(0)	ACK	10(2)
NON	01(1)	RST	11(3)

RST mesaj tipi, bir problem oluştuğunda veya istekler başarısız olduğunda istek sonucunda (CON veya NON) gönderilmektedir. Ayrıca boş bir RST mesajı bir aygıtın çalışıyor durumunda olduğunu anlamak için yerel ağda (ping) kullanılabilmektedir. CoAP başlık boyutu 4 Bayt ve her atılan mesaj bilgisinde 16 bitlik Mesaj Kimlik Numarası (Message ID) bulunmaktadır. Mesaj Kimliği için gönderilen 250 tane mesaj yeterlidir. Mesajların doğru olup olmadığını CON mesaj biti ile kontrol edilir. CON mesajları, bir ACK mesajı alınana kadar zaman aşımı mekanizması kullanılarak yeniden gönderilir. Bir cihaz bir isteğin içeriğini yerine getiremeyeceği zaman ACK mesajı yerine RST mesajı gönderir. Ancak doğru olmayan mesaj için NON olarak gönderilmektedir. Tekrar olarak bu gönderilen mesajların başarısız olması halinde RST mesaj biti gönderilmektedir.

2.2.2. İstek / Yanıt Çalışma Sistemi

CoAP istek/yanıt sisteminde metot kodları veya dönen yanıt kodları gösteren Şekil 2.10' da CoAP paket mesajlarını üzerinde taşır ve istemciden gelen bir istek CON veya NON mesajlaşma tipinin bilgisi üzerinde taşıyabilmektedir. CON mesajına verilebilecek cevap oluşmuş ise oluşturulan ACK mesajını beraber gönderilen paket içine eklenebilir. Ancak CoAP Sunucusu istekleri hemen yanıtlayacak şekilde hazır konum değilse paket içeriğine varsayılan değerde bir ACK ekler ve belli süre sonra paketi gönderir. Yanıt hazır olduğu zamanda yeni bir CON mesajı atanır ve yanıt bu mesaj ile birlikte gönderilir. Bu yanıt tipinde ayrı ACK ve ayrı yanıt yoktur, ikisi birleştirilmiştir. Bu yanıt verme türüne "ayrılmış cevap" denmektedir [21]. Bu koşulda istemci kendisine CON mesaj içerisinde gelen yanıt karşılığında ACK bilgisi göndermek zorundadır. Aksi taktirde sunucu sürekli olarak aynı yanıtı dönecektir. CoAP yanıtları isteklerle eşleştirmek için bir jeton (Token) kullanır. Jeton kavramı mesaj kimliğinden farklı bir kavramdır [22,23].



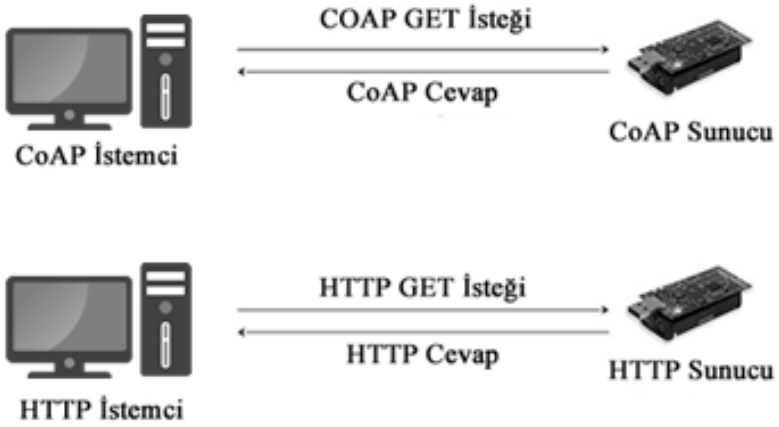
Şekil 2.10. CoAP istek/yanıt çalışma sistemi

Yukarıdaki Şekil 2.10'de görüldüğü gibi istemcinin sunucu ayrı yanıtlarla, istemciye, uzun işlem sürelerinde kaçınılması sağlanmıştır. Gereksiz tekrar gönderimler olmaktadır. Ve istekler NON olarak gönderilmişse, yanıt NON veya CON olacak şekilde gönderilmelidir. Çizelge 2.3'de CoAP Sunucusu tarafından kullanılan REST komutlarını HTTP' dekine benzer şekilde kullanmaktadır.

Çizelge 2.3. Temel REST komutları

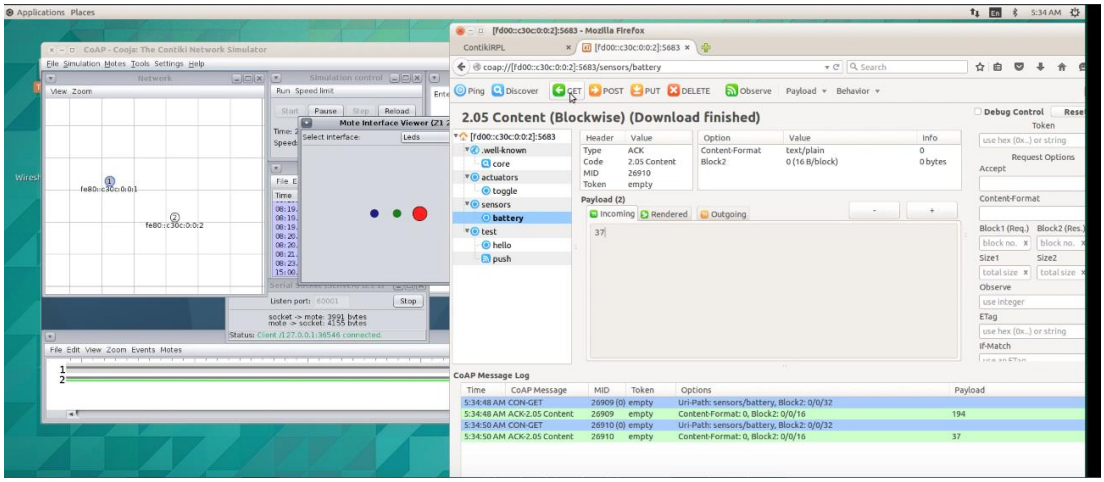
GET	Paket verisini listeleme ve veri görüntülemek için kullanılır.
PUT	Paket verisinin saklanmasını sağlar.
POST	Paket verisine değer göndermek için kullanılır.
DELETE	Paket verisi siler.

Ayrıca veri iletiminde noktadan noktaya yayın (Unicast) sağlayan bir protokoldür. Bu iletişim yanı sıra, birden fazla alıcıya veri gönderilmesi esnasında bant genişliği tasarrufunu sağlayabilmek için akışın sadece bir adet kopyasının ağa gönderilmesi esasına dayanan tekniği (multicast) kullanmaktadır [24].



Şekil 2.11. HTTP ve CoAP protokolleri karşılaştırılması

CoAP protokolü ile düğüm adresinden REST servis komutu GET komutu Şekil 2.12'de gösterildiği gibi düğümünden alınan sıcaklık bilgisi Firefox tarayıcısının Cooper Firefox için CoAP kullanıcı İstemcisi, URI şeması için bir adresi yükler ve kullanıcıların IoT cihazlarına göz atmasını ve etkileşimde bulunmasını sağlamaktadır.



Şekil 2.12. TelosB düğümünden REST ile GET metoduyla sıcaklık değerini alınması

2.2.3. Mesajlaşma Düzeni

CoAP protokolünün mesaj formatı aşağıda verilen Çizelge 2.4' de gösterilmiştir [25].

- **Sürüm (Ver):** CoAP başlık formatındaki 2 bitlik tamsayı şeklinde oluşturulmuş sürüm numarasını belirtmektedir.
- **Mesaj Tipi (T):** Mesaj tipleri CON (0), NON (1), ACK (2) ve RST (3) şeklinde 2 bitlik işaretli tam sayıdan oluşturulmuştur.
- **Jeton Boyutu (TKL):** Tanımlanan jeton sistemdeki değişken karakter uzunluğunu 4 bitlik bir tamsayıdan belirtir.
- **Kod:** CoAP Sunucusu üzerinden geçen tüm işlemlerin sonucunu gösterir. Örneğin; istemciden gelen başarılı gelen bir isteği, başarılı yanıtı veya hata yanıtını gösterebilir ve 8 bitlik tamsayıdan oluşmaktadır.
- **Mesaj No:** Mesaj tiplerini denkleştirme işlemini ve tekrar tespit etmek için kullanılır. Jetonun içindeki tanımlanan değer istekleri ve yantları ilişkilendirmek için kullanılır.

Çizelge 2.4. CoAP mesaj formatı

0								1								2												3	
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1
Ver		T		TKL				Kod								Mesaj No															
JETON (TOKEN)																															
SEÇENEK (OPTION)																															
1	1	1	1	1	1	1	1	YÜK- TAŞIMA İŞARETÇİSİ (PAYLOAD)																							

CoAP yanıtı içinde bazen seçenek parametresi bulunmaz bazende birden fazla seçenek da barındırabilir. Seçenekler alanını yük-taşıma(Payload) işaretçisinin içinde 1 bayt'lık alan ayrılmaktadır. İşlemin sonuna geldiğimizde ise isteğe bağlı yük-taşıma alanı gelmektedir. Paket mesajında yük-taşıma varsa, bu yük-taşımadan önce 8 tane bitten oluşan yük-taşıma işaretçisi gelir. Yük-taşıma bölümü dediğimiz yer sonra UDP protokolüne kadar uzanabilmektedir.

Kısıtlayıcı Uygulama Protokolü gereklidir çünkü geleneksel protokoller, sınırlı kaynağa sahip cihazlar için uygun değildir. KAA ağ konumu uç noktada kalan düğümler ağları paket aktarımı daha "kayıplı" olma eğilimindedir. Ancak düşük güçlü cihazların yıllarca pillerle veya enerji toplama yoluyla çalışmaya devam etmesi ve mümkün olduğunca az enerji harcaması gerekir. CoAP, düşük güçlü KAA aygıtları gibi basit kısıtlı kaynakların internet üzerinden etkileşimli biçimde iletişim kurmasını sağlayan bir yazılım protokolüdür. Kullanıcı Datagram Protokolünü (UDP) destekleyen cihazlarda çalışır ve küçük mesaj boyutları, mesaj yönetimi ve düşük güç tüketen, düşük bellekli cihazlar için ideal olan hafif mesaj yükü özelliklerine sahip bir uygulama katmanı sunmaktadır [26].

CoAP'ın amacı düşük kapasiteli (6LoWPAN) cihazlar için veri yükü getirmeden ve uygulamalar arası haberleşmede basit iletim yolu oluşturmasıdır. Başka bir özelliğide paket başlık (header) bilgi kısmındaki içeriği sadeleştirirek kısaltılmıştır. Dolasıyla gelen isteklerin güvenliği konusunda HTTP'ye benzer şeklinde kullanılan Güvenli Veribloğu Taşıma Katmanı (DTLS) kullanarak iletişim güvenliğini tahsis edebilmektedir. Genellikle CoAP kullanımına baktığımızda HTTP ve MQTT tercihi arasında bir yerde kalmaktadır. Ancak bant genişliği problemi olan yerel ağda iseniz HTTP için sorun yaratmaya başlamış ise CoAP' a kullanımı daha uygundur [27].

Paketler birbirinden bağımsız olarak, bir mesaj istek, yanıt veya boş olabilmektedir. Mesajın tipi CoAP kaynak türüyle birlikte verilmektedir. Bir isteğin sonucunu belirtmek için bir yanıt kodu döndürülür. CoAP istemcisi ve sunucu Düğüm Yük Denetim Sistemi (FSU)' nun yanı sıra Çizelge 2.5' de ve Çizelge 2.6' daki sistem parametrelerini kullanmaktadır. Örneğin, FSU tarafından verilen CoAP CON + GET isteği için bir KAA düğümü, CoAP sunucusundan nemi almak için düğümünde çalışan bu, CoAP yanıtını ACK mesajına cevap verme yöntemine "ayrılmış yanıt" denmektedir [28].

Çizelge 2.5. CoAP algılayıcı düğümün kaynak parametreleri

Kaynak	GET	POST	Açıklama
/st	X		Sıcaklık
/sh	X		Nem
/sv	X		Gerilim
/r	X		Sıcaklık, Nem ve Gerilim
/l		X	LEDler
/ck		X	AES Şifrelenmiş Anahtar

Çizelge 2.6. CoAP kaynak erişim parameterleri

Resource	GET	PUT	Açıklama
/ni	X		Düğüm hakkında bilgi verir ve 6LoWPAN /RPL ağı üzerinden entegrasyonu sağlar.
/warntemplow		X	Uyarı Sıcaklık düşük
/warntemphi		X	Uyarı Sıcaklık yüksek

2.3. CoAP ile Diğer IoT Haberleşme Protokollerinin Karşılaştırılması

Sınırlandırılmış Uygulama Protokolünü tasarlarken, IETF' nin amacı, onun ölçeklenmesi ve genişletilebilir olmasını sağlamaktı ve temel alınan iletişim paradigması sayesinde IOT protokolleri arasında en hızlı ve etkin haberleşmeyi sağlamaktadır. Diğer IoT protokolleri arasında Çizelge 3.7' de çoğunluğu en az on yıldan fazladır kullanılan HTTP, WebSocket, Genişletilebilir İleti Gönderme ve Durum Protokolü (XMPP) ve Telemetri İletimi (MQTT) bulunmaktadır [29].

Çizelge 2.7. CoAP kaynak erişim parameterleri

Özellikler	HTTP	AMQP	DDS	CoAP
Gerçek Zamanlılık	ms	msler	µs	ms
Haberleşme	P2P	P2P	P2P	P2P
Adresleme	kullanici @alan	Hiyerarşik	Route Filtreleri	URI
Taşıma Katmanı	TCP	UDP	UDP	DTLS, UDP
Sistem Yükü	Basit	Karmaşık	Karmaşık	Basit

Diğer protokollerin CoAP'a göre bozuk ölçeklendirilmiş yapılardan oluşması güvenli veri alışverişi yapmaya elverişsiz bir ortam sunmaktadır. REST altyapısına bağlı olarak CoAP' ın ölçeklenebilir olması diğer protokollere göre en büyük avantajıdır. CoAP kaynağı sınırlı cihazlar ve ağlar için iyi bilinen web RESTFUL modelini kullandığından cihaz mümkün olduğunca sunucu tarafından yönetilir ve istemci ile ilgili bir içerik (context) ve oturum bilgisi (stateless) tutmaz. Bundan dolayı CoAP' ın düşük güçte çalışan aygıtların haberleşmesi için kullanılması en önemli özelliğidir.

2.4. Veri Bloğu Aktarım Katmanı Güvenliği (DTLS)

Güvenli Veri Bloğu Aktarım Katmanı (DTLS) protokolü belirtilen ağ trafiğini güvenli hale getirmek için oluşturulmuş verileri şifreleme işlevini üstelenen bir protokoldür. DTLS, SSL ve TLS' in standartlaştırılmış halidir [30]. Güvenlik göz önünde bulundurulduğunda CoAP protokolündeki güvenlik hizmetleri açısından bütünlük, kimlik doğrulama ve gizlilik olmak üzere üç ana unsur vardır. DTLS hepsini barındıran ve yönetimi kendi içinde alt yapısal temellinde TCP protokolünü örnek almaktadır. COAP protokolü uygulama katmanında bir önceki katmanda paket aktarımı yaparken UDP kullanmaktadır. DTLS protokol olarak iki ana problemi çözmektedir. Bunlar yeniden sıralama ve paket kaybı sorunlarıdır. Bu problemleri çözmek için protokolde üç kontrol noktası oluşturulmuştur [31].

a.) Paketlerin yeniden iletiminin sağlanması
b.) El sıkışma içinde sıra numarası atanması
c.) İletilecek paket hedefinin tekrar algılanması

Düşük kapasiteli IoT cihazları için TinyOS işletim sisteminde TinySec, DTLS ve 802.15.4 ile kullanılmaktadır. Ancak IETF RFC 6347 'de rol modeline göre aksine DTLS güvenilir mesaja bağlı değildir [32]. Dolasıyla diğer aygıtlarla haberleşmeye geçtiklerinde kimlik doğrulama ve asimetrik şifreleme kullanılır ve daha sonra karşılıklı anlaşma işlemi yapabilmesi için bir simetrik anahtar üzerinde anlaşma sağlamalıdır [33]. Güvenli olmayan verilerin taşınması, TLS mümkün olduğu kadar TLS kayıplı mesaj aktarımı nedeniyle protokolde yapılan ana değişiklikler kullanıldığında paket kaybı, mesajların yeniden düzenlenmesi ve mesaj boyutlar gibi konularında yönetimi konusunda başarılıdır. Çizelge 2.8' de görüldüğü üzere ağ katmanı güvenlik protokollerinden farklı olarak uygulama katmanındaki DTLS uçtan uca iletişimi korumaktadır. Uçtan uca iletişim koruması, saldırganın güvenliği ihlal edilmiş bir düğümden geçen tüm metin verilerine erişmesini engellemektedir. DTLS ayrıca alt katman güvenlik protokollerinde meydana gelen kriptografik ek yük sorunlarını da önlemektedir [34].

Çizelge 2.8. Standartlaştırılmış güvenlik çözümleriyle IoT yığını

IoT Katmanları	IoT Haberleşme Protokolleri	Güvenlik Protokolü
Uygulama	CoAP	Yok
Ulaştırma	UDP	DTLS
Ağ	IPV6, RPL	IPsec, RPL güvenlik
Fiziksel	MAC (IEEE 802.15.4)	80215.4 güvenlik

2.4.1. Taşıma Katmanı Güvenliği (TLS)

Taşıma Katmanı Güvenliği (TLS) ve önceden web dünyasının en çok kullandığı Güvenli Soket Katmanı (SSL), aygıtların bulunduğu ağlarda şifrelenmiş bir şekilde paket göndermesini sağlamak amacıyla geliştirilmiş şifreleme protokolüdür. TLS protokolü X.509 adında OpenSSL sertifikalarını kullanır ve bundan dolayı karşı tarafla iletişime geçeceklerinde önce kimlik doğrulanmasını asimetrik şifre ile sağlanır. Daha sonrasında taraflar karşılıklı olarak simetrik anahtar üzerinde anlaşır. Daha sonra oturum anahtarı dediğimiz yapılar şifrelenmiş veri şifrelemek için tekrar kullanılabilir [35,36,37].

Çizelge 2.9. Yayınlanan kronolojik DTLS sürümleri

Protokol	Yıl
DTLS 1.0	1999
DTLS 1.1	2006
DTLS 1.2	2008
DTLS 1.3	2015 (Taslak)

Veri gizliliği, atak yapan düğümlerin veriyi elde edememesini garanti altına alabilir ancak verinin yetkisi olmayan kullanıcılar tarafından değiştirilmesini engelleyemez. Veri bütünlüğü iletişimde mesajın değiştirilmemesini garanti etmektedir. Bir servis reddi (DoS) saldırısında algılayıcı mesajları bozularak ağda haberleşmeyi sekteye uğratabilir. Ayrıca, doğrudan doğruya bir KAA ortamında atak yapan saldırgan düğümlerin dışında mesajlar paket aktarımında veri kaybından dolayı hata alabilir. Bundan dolayı veri bütünlüğünü sağlamak için veya Dairesel Kodları (Cyclic Codes) ya da Mesaj Doğrulama Kodları (MAC) denilen mesaja eklenen bir sayıların kullanılması zorunludur [38].

Paket aktarımı sırasında mesaj bütünlüğünü mesaj kimlik doğrulama kodları için sağlamaktadır. Kimlik doğrulama günümüzde birçok alanda anlık mesajlaşma, elektronik posta, ağ tarama ve İnternet üzerinden sesli iletişim gibi uygulamalarda yaygın olarak kullanılmaktadır. Bu protokolün en önemli özelliği uçtan uca içeriğin gizliliğinin korunmasıdır. Bu yüzden kısa süreli oturum anahtarı, uzun süreli gizli simetrik anahtardan türetilmemelidir. Çizelge 2.10'daki X.509 sertifikalarının seçimi sonucunda TLS protokol katmanları ile sertifika yöneticileri ve açık anahtar alt yapısı, sertifika ve sahibi arasındaki ilişkinin doğrulanmasının yanı sıra oluşturulması, imzalaması ve sertifikaların geçerliliğinin yönetilmesi için gereklidir. Bu, güvenilirlik ağı yoluyla kimlik doğrulamasından daha faydalı olduğu halde, 2013'deki küresel izleme ifşası sertifika yöneticilerinin ortadaki adam saldırısına (man-in-the-middle attack) izin verdiğini bundan dolayı güvenlik bakımından zayıf bir nokta olduğunu bilinir hale getirmiştir [39].

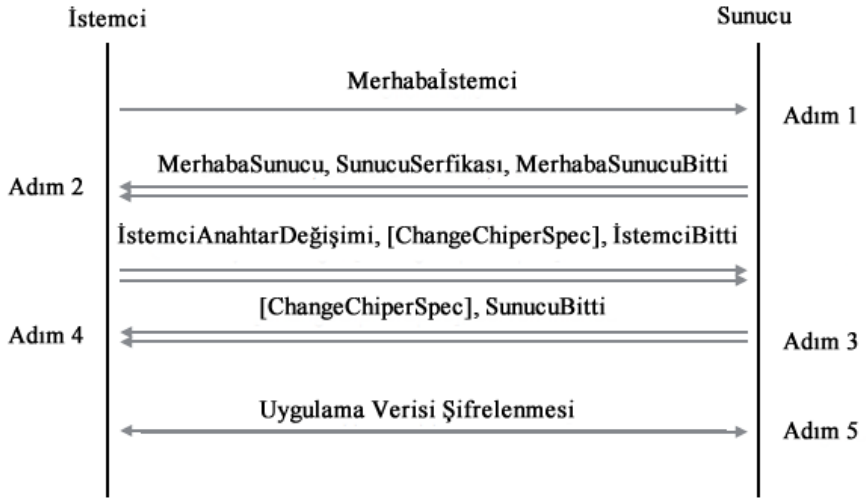
Çizelge 2.10. TLS protokol katmanları

El Sıkışma Protokolü	ChangeChiperSpec Protokolü	Uyarı Protokolü	Uygulama Protokolü
Protokol Kaydı			
TCP			

Genellikle internet sitelerinde SSL sertifikası kullanılır ve TLS uygulama katmanında ağ bağlantıları verisini şifreler. OSI modelde eşdeğer olarak, TLS / SSL oturum katmanında başlatılır ve sunum katmanı çalıştırılır. Oturum katmanı asimetrik şifrelemenin kullanıldığı bir el sıkışma işlemine sahiptir. Buradaki amaç, oturum için bir paylaşılan anahtar ve şifreleme ayarlarını oluşturmaktır. Sunum katmanı ise simetrik şifreleme ve oturum anahtarını kullanarak haberleşmenin geri kalanını şifreler. Bu iki modelde TLS ve SSL, bölütleri şifreli verileri iletmekte olan taşıma katmanı adına çalışmaktadır.

2.4.2. TLS El Sıkışma Protokolü

El sıkışma protokolü, uygulamayı kullanan istemci ve sunucu tarafından değiştirilen bir dizi mesajdan oluşur. Bu ilk aşama sırasında, uç noktalar birbirini karşılıklı olarak doğrular ve güvenlik parametrelerini müzakere eder. TLS' de el sıkışmanın birden fazla yolu vardır. Bir veya iki yönlü kimlik doğrulaması veya içeren belirli bir uygulama sertifika doğrulama vardır. Bu isteğe bağlı el sıkışma işlevselliği mesajlar bu bölümün kapsamı dışındadır ve bu nedenle sadece temel el sıkışma mesajları Şekil 3.14' de aşağıda belirtilmiştir. Bu el sıkışma iletileri TLS özelliğinde tanımlanan adımlarda gruplandırılmış ve bitişik olarak gönderilen karşılıklı mesajların gruplarında ise çift yönlü gösterilmiştir [40].



Şekil 2.13. TLS protokol katmanları

2.4.3. Şifreleme Tipleri

2.4.3.1. Asimetrik Şifreleme

Asimetrik şifreleme ve şifre çözme işlemi farklı anahtarlar ile yapılır. Bu anahtar çiftini oluşturan anahtarlara açık ve özel anahtar adı verilir. Bu şifreleme yönteminde özel anahtar gizli tutulmalıdır fakat açık anahtar gerekli kişilere verilebilir ve başka kişilerle paylaşılabilir. Bu özelliğinden dolayı asimetrik şifreleme, açık anahtarlı şifreleme adıyla da anılmaktadır [41,42].

Asimetrik şifreleme algoritmasıyla iletişime kullananlar:

- Şifreleme içeri aynı algoritmayla üretilir

- Uyum açısından taraflar gerçekleme yapabilirler
- İhtiyaç durumunda şifreleme anahtarına ulaşabilirler

2.4.3.2. Simetrik Şifreleme

Simetrik şifreleme ve şifre çözme işlemi aynı anahtar ile yapılır. Simetrik şifrelemede bu anahtar gizli tutulmalıdır. Bu yüzden, bu tip sistemlere gizli anahtarlı şifreleme sistemi adı da verilmektedir [43,44].

Bu sistemde paket gönderimi yapan taraflar:

- Şifreleme içeri aynı algoritmayla üretilir
- Uyum açısından taraflar gerçekleme yapabilirler
- Benzer anahtar tipini kullanırlar

Simetrik şifreleme artı tarafları aşağıdaki gibidir:

- Algoritmalar hızlıdır
- Algoritmaların aygıtlarda gerçeklemesi kolaydır
- "Gizlilik" güvenlik hizmetini yerine getirir

Simetrik şifreleme zayıf tarafları aşağıdaki gibidir:

- Ölçekleme konusunda iyi bir yöntem değildir
- Zorunlu anahtar dağıtımı zordur
- Kimlik doğrulama ve Bütünlük güvenlik kriterleri gerçeklenmesi zordur

Şifreleme sistemlerinin karşılaştırması

Asimetrik ve simetrik şifreleme sistemlerinin özellikleri Çizelge 2.11'de verilmektedir.

Çizelge 2.11. TLS protokol katmanları

Konu	Simetrik Şifreleme	Asimetrik Şifreleme
Gizlilik	+	+
Bütünlük	-	+
Kimlik Doğrulama	-	+
Performans	Daha hızlı	Daha yavaş
Güvenlik	Anahtar boyutuna bağlı	Anahtar boyutuna bağlı

2.4.4. KAA ve Şifreleme Kriterleri

Güvenli şifreleme tipi iki çeşittir: Simetrik Şifreleme, Asimetrik Şifreleme. Şifreleme, esasen iletişime geçen iki veya ikiden daha çok nesnenin bilgi alışverişini güvenli bir şekilde kurmasını, temelinde matematik problemlerinde kullanılan tekniklerin ve uygulamaların bütünüdür. İletişime geçen iki tarafın güvenlik açısından istekleri bulunur. Bu istekler iletişimde kriterlerin oluşturması için bölümlere ayrılmıştır. Paket aktarılırken bazı kriterler aşağıda verilmiştir.

Bütünlük: Paket bilgisinin içeriğinin gönderim esnasında değiştirilememesidir.

Kimlik Doğrulama: Bilgiyi gönderen kişinin kimliğinin doğruluğunun kontrol edilmesidir.

Gizlilik: Paket bilgisinin içeriğinin gizli olmasıdır.

Haberleşmenin Sürekliliği: Paket aktarımının kesintiye uğramadan yapılmasıdır.

Örneğin; 128 bitlik şifrelenmiş iki tabanında sayı formatı aşağıda verilmiştir.

128 bitlik Anahtar = 11001010101100010001101000000111011
01000100111101100111010011101

Temel Şifreleme Algoritmaları: Şifre paketi tipik olarak bir anahtar değişimi, bir kimlik doğrulama, bir toplu şifreleme ve bir MAC algoritmasından oluşur. ChipherSuite şifre paketinin üzerinden bir ağ bağlantısı güvenliğini sağlamak için gerekli algoritmaları kendi içinde içermektedir [45]. Aşağıda en çok tercih edilen sertifikaları Çizelge 2.12'de tablosu verilmiştir.

Çizelge 2.12. DTLS protokol katmanlarında kullanılan algoritmalar

Algoritma Tipleri	Şifreleme Algoritması
Anahtar Değişim Algoritmaları	RSA, DH, ECDH, ECDHE
Kimlik Doğrulama Algoritmaları	RSA, DSA, ECDSA
Toplu Şifreleme Algoritmaları	AES, 3DES, CAMELLIA
MAC algoritmaları	SHA, MD5

DTLS sertifikasında kullanılan algoritma örnek sertifikası aşağıdaki gibidir.

TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384

- ECDHE, anahtar değişim algoritmasını ifade eder.
- ECDSA kimlik doğrulama algoritmasını ifade eder.
- AES_256_CBC, toplu şifreleme algoritmasını gösterir ve SHA384 MAC algoritmasını gösterir.
- AES_256_CBC, bu şifreleme paketinin özellikle CBC (şifre bloğu zincirleme) modunda çalışan 256 bit AES [31,32] kullandığı anlamına gelir.
- Benzer şekilde, SHA384 şifreleme paketinin Güvenli Karma Algoritmasının (SHA) belirli bir sürümünü kullandığı anlamına gelir.

Anahtar Değişimi: Örnek olarak şifreleme algoritması RSA algoritması Asimetrik şifreleme yöntemini kullanan şifreleme türüdür. Kimlik denetimini sağlamak için diğer bir yöntem de IKE (İnternet Şifre Değişimi) protokolüdür [26]. IKE kimlik denetimini yapılabilmesi için özellikle sertifika belirlenmesi, kullanıcı hesabı oluşturulması, tek kullanımlık parola, sayısal sertifikalar gibi çeşitli yöntemlerle uygulanmaktadır. TLS_RSA_WITH_AES_CBC_SHA gibi olabilir. Anahtar değişim algoritması AES_CBC Gelişmiş Şifreleme Standardı (Cipher-Block Chain) modda simetrik şifreleme algoritması ve SHA güvenli algoritma, ileti bütünlüğü sağlamak için kullanılmaktadır [46]. Bir şifre paketi oluşturan algoritmalar tipleri şunlardır:

- Anahtar değişim algoritmasında anahtar kullanımı açısından tek anahtar yerine birden fazla anahtarı açık ve kapalı şekilde kullanır.
- Kimlik doğrulama algoritması- sunucu kimlik doğrulamasının ve (isteğe bağlı) istemci kimlik doğrulamanın nasıl gerçekleştirileceğini belirler.
- Toplu şifreleme algoritması- gerçek verileri şifrelemek için hangi simetrik anahtar algoritmasının kullanılacağını belirler.
- Mesaj Doğrulama Kodu (MAC) algoritması- veri bütünlüğü kontrollerini gerçekleştirmek için bağlantının kullanacağı yöntemi belirler.

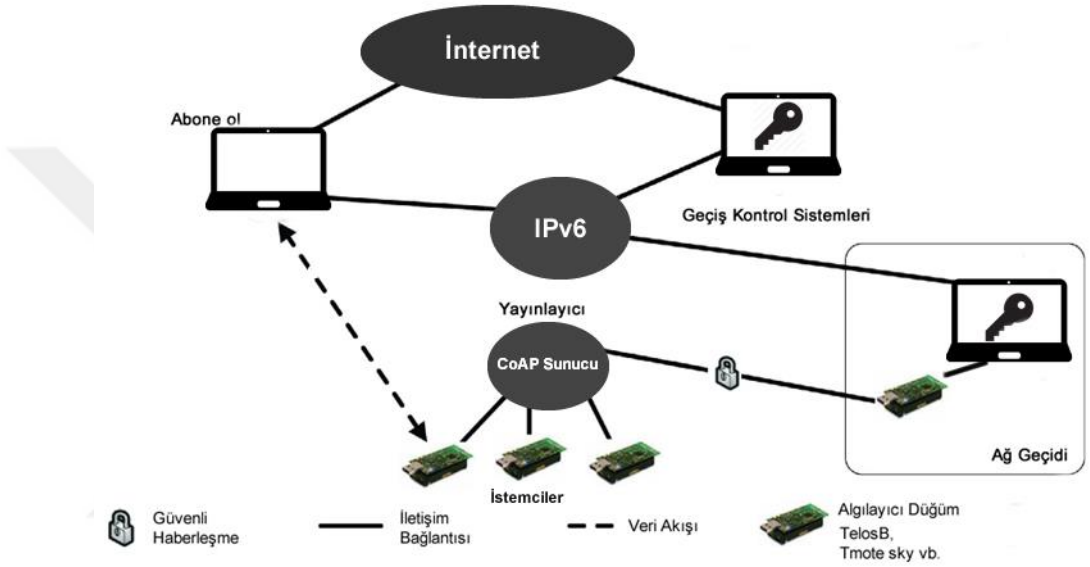
BÖLÜM 3. DTLS İLE GÜVENLİ COAP PROTOKOLÜ TASARIMI

3.1. Sistemin Çalışma Mimarisi ve Akış Diyagramı

Bu tez çalışmasında programlanabilir mikro denetleyici yapısına sahip TelosB algılayıcı düğüm ve benzetim ortamında TmoteSkye düğümü yardımıyla iki hedefi birbiri ile şifreli haberleştirecek bir şekilde yapı oluşturulmuştur. Haberleşemeyi hızlı yapmak için çalışmada CoAP haberleşme protokolü tercih edilmiştir. Çünkü CoAP kısıtlı kapasiteli aygıtlarda kullanılan en performanslı haberleşmeyi sağlayan bir IoT haberleşme protokolüdür. Ayrıca tez çalışmasında şifreleme için DTLS altyapısı kullanılarak AES algoritmasıyla şifre oluşturulmuştur. Tasarlanan sistem birçok IoT uygulamasında ortak kullanılabilecek X509 sertifikası ile beraber düğüm aygıtlarının güvenliğini sağlayacak şekilde yapılmıştır.

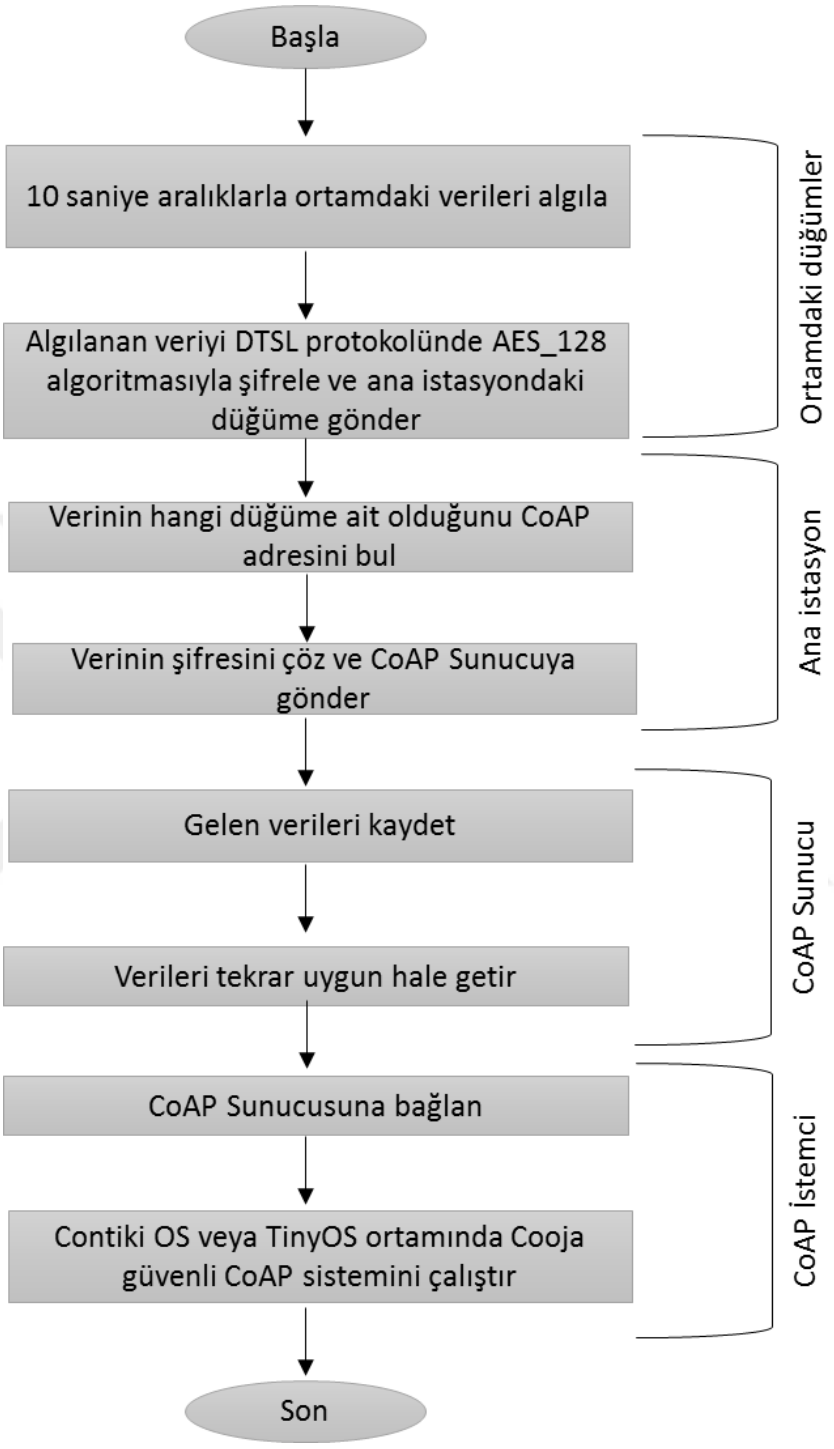
İlk adımda, KAA üzerinde iki farklı bölgede bulunan yerel ağlara bağlı programlanabilir fiziksel platformların aynı zamanda olmayacak olarak CoAPBlip uygulaması ile CoAP Sunucusu ve İstemcisi arasında veri alışverişi gerçekleştirebilmesi sağlanmıştır. İkinci adımda, bu platformun üzerinden akacak paket trafiğini IP'lerine göre ayıran ve şifreleme işlemini yapan yazılım fonksiyonları oluşturularak, platform üzerine Contiki-OS' da Cooja benzetim ortamında NesC program arayüzü yardımı ile işlenmiştir. Üçüncü adımda ise platformun bulunduğu yerel ağda bulunan ana istasyonda verileri yüklemek üzere bir kontrol benzetim oluşturulmuştur. Bu benzetimler sayesinde cihaza çeşitli uzunluktaki paketler gönderilerek, cihazdan performans verileri ve paket trafiği Wireshark ağ izleme aracı ile takip edilmiştir. KAA üzerinde bulunan her iki ağda tanımlı arabulucu düğüm, bilgisayar ve düğüm aygıtlarının vasıtasıyla güvenli bir veri transferi sağlanmıştır.

Şekil 3.1’ de KAA ağlar CoAP İstemci ve CoAP Sunucu arasında çift taraflı güvenli veri iletimi gösterilmiştir. Tasarlanan sistemde kısıtlı kapasiteli düğümler asenkron haberleştiği için ters yönde tekrar istek gönderebilmiştir. Bu çalışma kablosuz algılayıcı aygıtlarının noktadan noktaya birbiriyle iletişimde bulunan IoT uygulamalarında güvenlik sorunlarına çözüm olması için önerilmiştir.



Şekil 3.1. Geliştirilen çözüm platformunun mimari yapısı

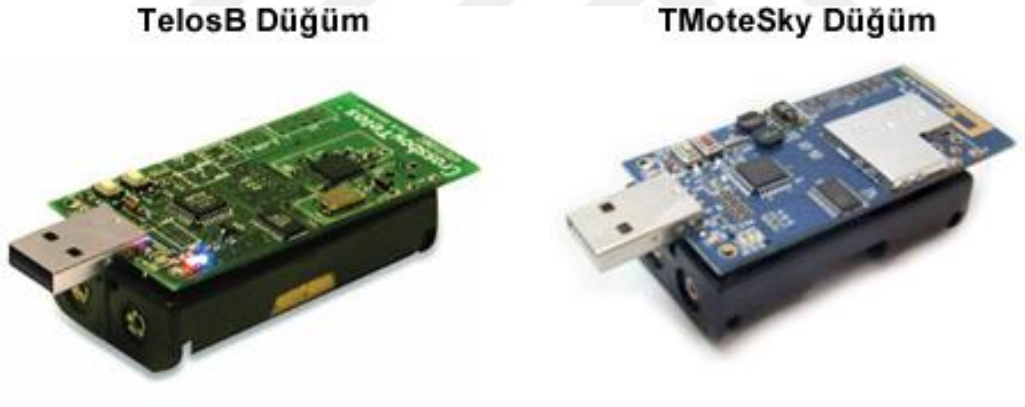
Son adımda KAA yerel ağında bulunan TelosB ve TmoteSky aygıtlarının görevi, gelen isteklerin hangi IP’den olduğunu ve Port bilgisine bakılarak gönderilen paketin şifreli ya da şifreyi çözebilecek IP’ye yönlendirilmiştir. Düğüm aygıtının kendi içerisindeki yazılım yardımıyla gerçekleştirdiği bu yönlendirme modeli Şekil 3.2’de verilen akış diyagramındaki aşağıdadır. Tasarlanan bu diyagramda düğüm devamlı olarak 10 sn aralıklarla ortamdaki verileri algılar ve kendi IP’si üzerindeki paket trafiğini dinleyerek, gelen paketin kaynak adresine göre hareket eder. Eğer bilinmeyen bir düğümden adresinden paket geldiyse bunu KAA ağında bulunan sertifika yayımlayıcı ait kullanıcı arayüzüne iletir. Cihaz arayüzündeki yazılım bunu sertifikasını kontrol eder. Eğer KAA ağda bulunan tanımlı düğümden paket geldiyse, ana istasyondaki düğüm uzak adrese yönlendirileceğini ve gönderilen paketin şifreleneceğini bilir. Bu olayında tersinde uzak mesafeden gelen paket geldiğinde ise düğüm aygıtı uzak mesafeden geldiğini tespit ederek gerekli yapılacak işlemleri başlatmaktadır.



Şekil 3.2. Uygulanan algılayıcı güvenlik sistemine ait akış diyagramı

3.1.1. Kullanılan Yazılım ve Donanımlar

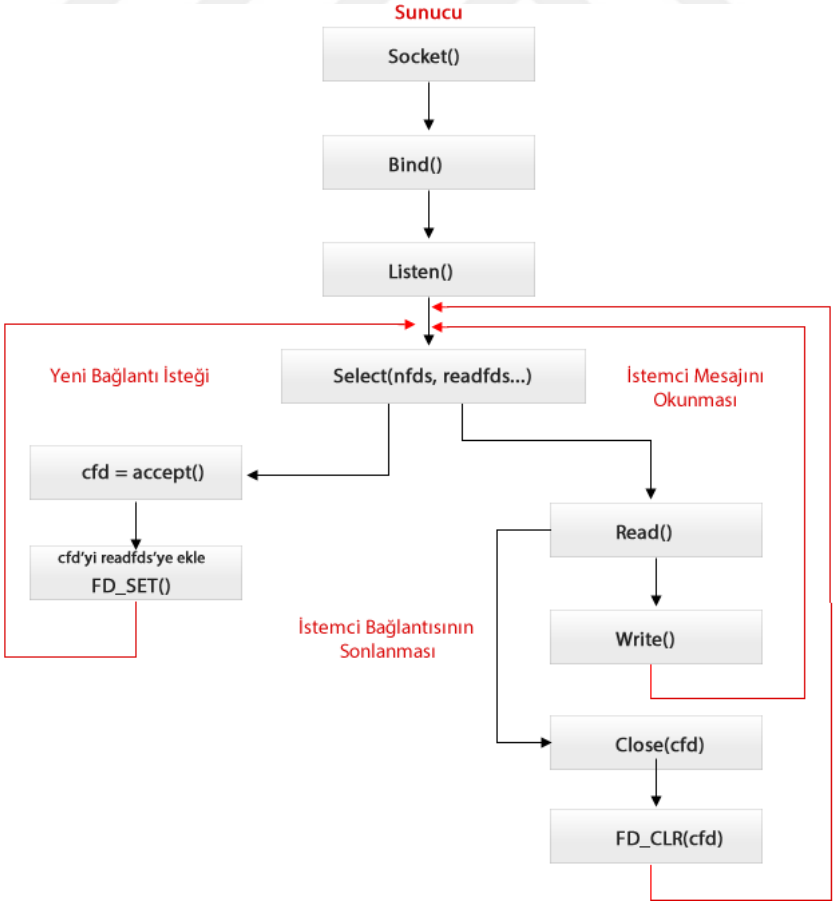
Bu çalışma kapsamında kablosuz algılayıcı ağların bulunduğu çeşitli düğüm aygıtlarını Crossbow firması tarafından üretilmektedir. Çalışmada kullanılan algılayıcı düğümler TMoteSky, TelosB kullanılmıştır. Bunun nedeni KAA alanındaki akademik çalışmalarda en çok kullanılan algılayıcı olmasıdır. Düğümleri USB üzerinden COM portu kullanarak geliştirilen yazılım yüklenmiştir. Ayrıca TelosB düğüm cihazının maliyeti yüksek olduğu için benzetim ortamında Şekil 3.4' deki TMoteSky düğümü kullanılmıştır. TelosB düğümü üzerinde MSP430 mikrodenetleyiciye kullanıyor ve özelliklerini incelediğimizde 16 KB veri hafızası içeren 48 KB kod ve 8 MHz'de çalışan bir düğümdür. Kablosuz iletişimi IEEE 802.15.4 uyumlu Chipcon CC2420 alıcı/verici tüm devresini kullanarak gerçekleştirilebilmektedir. Kullanılan KAA donanım aygıtlarının görüntüsü Şekil 3.3' de verilmiştir.



Şekil 3.3. Kullanılan algılayıcı düğümler

3.2. Geliştirilen CoAP Protokolü İstemci / Sunucu Modeli

Bu çalışmada kullanılan CoAP sunucusu arka tarafta UDP protokolü gibi davranmaktadır. CoAP Sunucusu ortamdan bağımsız olarak CoAP istemcilerinden gelebilecek mesajları dinleyebilmektedir ve IPv4 ve IPv6 destekleyecek şekilde tasarlanmıştır. CoAP paketlerini gönderen düşük kapasiteli aygıtlar (6LowPAN) ağında UDP altyapısını kullanan istemci düğümleridir. CoAP Sunucu, ağdaki paketleri izleyebilmesi için ilk adımda soket bağlantısı kurduktan sonra onunla iletişime geçer ve izlemeye başlar. Okuma sistem çağrısı sayesinde birçok istemciden gelen paketler dinlenebilmektedir. Çok sayıda istemci olmasına karşın UDP sunucu çoklu paket dinleyebilmekte ve bu sayede paket kaybının önüne geçilmektedir. Okuma sistem çağrısı yaptıktan sonra geliştirilen sistem tek bir işlem üzerinden birden fazla istemciye servis sunabilmektedir. Geliştirilen CoAP sunucu sistem çağrılarının çalışması Şekil 3.4'de gösterilmiştir.



Şekil 3.4. Gerçekleştirilen CoAP Sunucu sistem çağrısı

Web uygulamalarında kablosuz ağ bulunan düşük kapasiteli cihazlara bağlayabilen bir CoAP sunucusu tasarımı sunulmaktadır. Tasarım olarak CoAP cihazı ve Web uygulaması ile iletişim kurmak için CoAP sunucusu tarafından kullanılan iletişim altyapı analizinin yapılabilmektedir. Bu çalışma anlamda düşük kapasite cihazlarda kısa ve uzun süren paket alışverişlerinde başa çıkabilecek protokollerin kullanılması olarak düşünülmektedir. Uzun ömürlü iletişimlerin neden olduğu veri akışı, yükü ve iş yükünü azaltabilecek şekilde tasarlanmıştır. CoAP Sunucu uygulama bileşenleri NesC dili ile ayrıntılı tanımlandıktan Çizelge 3.1' da belirtildiği şekilde oluşturulmuştur.
Uygulama, aşağıdaki verileri parametreler olarak gerektirir:

- **Rid** - yanıtın işlem kimliği;
- **Uri** - sunucu yanıtını temsil eden durum kodu;
- **Coap_icerik** - CoAP yanıtının yükünü içeren arabelleğin konumu;
- **Metod** - istenen yöntemi temsil eden kod;
- **İcerik** - CoAP'ın paket içeriğini sakladığı parametre

Çizelge 3.1. CoAP Sunucu uygulama bileşenleri arayüzü

Kaynak Kodu	
interface CoAPServer { event void request (COAP_RID rid, COAP _ABSURI * uri, COAP _METOD metod, COAP _ICERIK * icerik, bool TOACK);	command hata_t cevap (COAP _RID_t rid , COAP _DURUM durum, COAP _ICERIK * icerik); }

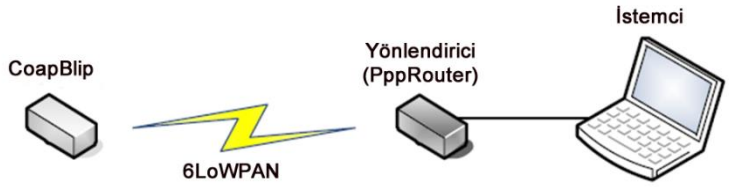
Web uygulamaları ile kalıcı bağlantılar oluşturmak için WebSocket protokolü CoAP sunucusu içinde bulunmaktadır. Gözlem protokolü aynı amaçla, ancak CoAP cihazı ile iletişim kurmak için kullanılır. CoAP sunucusu ayrıca, geleneksel HTTP uzun arşivleme tekniğini kullanan web uygulamalarını desteklemektedir. CoAP vekil sunucusu sinin tasarımı bir önbellek belleği ve bir kaynak havuzu (RD) içermektedir. RD, CoAP cihazı tarafından sunulan kaynakların tanımlamasını sağlamak için kullanılmaktadır. Bir düğüm atlama, bir paketin son varış adresine ulaşmak için geçmesi gereken farklı ağ

sayısını anlamına gelmektedir. Tekli ve çoklu iletim (hop) ağı arasındaki ana fark, bir paketin nihai hedefe ulaşmak için atladığı düğüm sayısıdır.

3.2.1. KAA TelosB Düğümü ile Paket Aktarım Uygulaması

Tek İletim Ağı

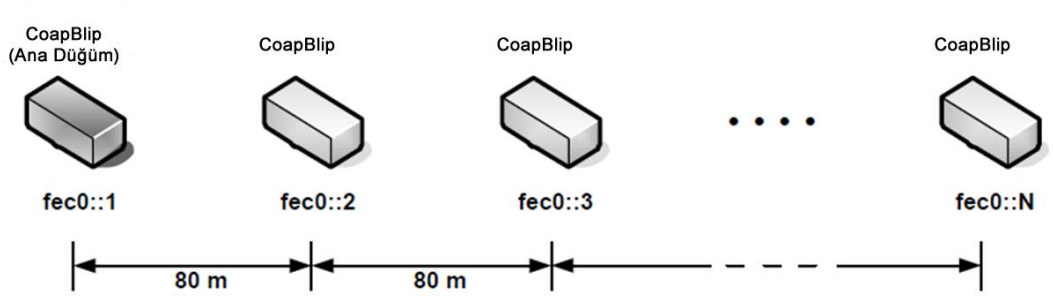
Tek iletim ağlarında paket iletiminde bir paket kaynaktan çıktığında, hedef adresine ulaşmadan önce tek bir paket aktarımını ana istasyona gönderme işlemini teker teker yapmaktadır. Şekil 3.5' de gösterildiği gibi düğümlerin kendi aralarındaki kapsama alanı dışında olan düğümlerle haberleşebilmesi için diğer düğümler aracılığıyla paketlerin bir düğümden başka düğüme haberleşmesini sağlamıştır. Ancak KAA'larda bu tip haberleşme herkese açık bir ortamda yapıldığı için veri güvenliğinin düşürülmesi gerekmektedir.



Şekil 3.5. CoAP Sunucu uygulama bileşenleri arayüzü

Çift İletim ağı

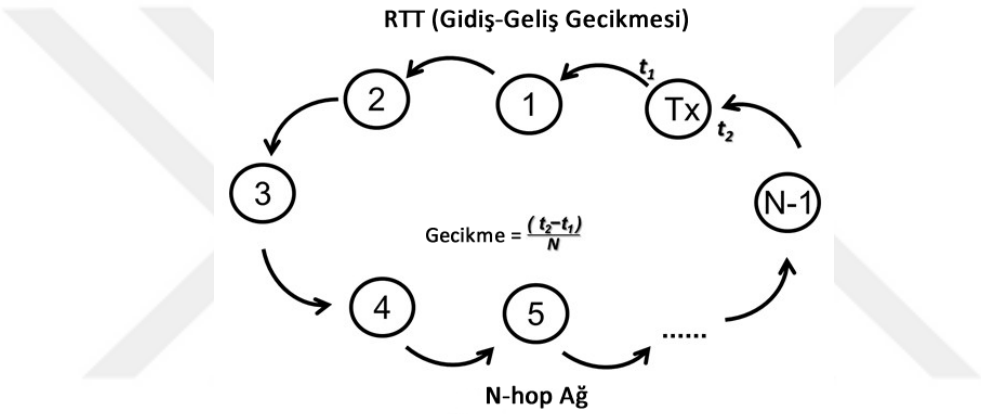
Çoklu iletim ağlarında bir paket Şekil 3.6'daki gösterildiği gibi hedef adresine ulaşmak için iki veya daha fazla düğümü dolaşmalıdır. Bir paket Farklı bir ağ üzerinden bir atlama yaparken yönlendiriciler, ağ köprüleri, anahtarlar, gibi çeşitli cihazlar aracılığıyla gidebilmektedir. Burada önemli olan kısım ana istasyona gelene kadar paket kayıplarının en aza indirilerek gönderilmesidir.



Şekil 3.6. CoAP Sunucu uygulama bileşenleri arayüzü

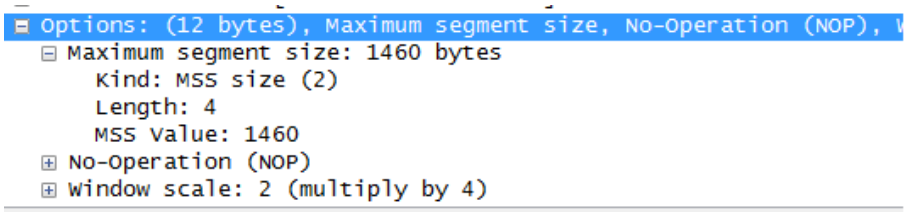
Çift-İletim ağında paket gecikme ölçümü

KAA düğümleri arasındaki RTT (Gidiş-Geliş Gecikmesi) ölçüm hesaplanması Şekil 3.7' de paketler yol boyunca iletilmek zorundadır. N-iletim (hop) ağ için gecikme elde etmek kolaydır. Tıkanıklık kontrolü haberleşen düğümler için değil, iletim ortamı için yapılmaktadır ve bu kontrolü gönderici tarafında yapılmalıdır. Gönderici taraf ağı iletim ortamını ne kadar kapasiteye ihtiyacı olduğunu öğrenebilmesi için tıkanıklık kontrolü yapmalıdır. CoAP uygulama katmanından önce UDP protokolü ile haberleştiği her bir bağlantı için yeniden iletim zamanlayıcısı kullanmak zorundadır. RTT hesaplaması yaparken Şekil 3.6'daki gibi formül kullanılmaktadır.



Şekil 3.7. N-iletim(hop) KAA ağında RTT hesaplanması

Wireshark' tan ağı izlediğimizde aşağıdaki şekilde seçenekler (Options) sekmesinde belirtilen kısımlardan düğümlerin konumlarıyla ilgili olan fazladan belirtilen özellik Maksimum Bölüm Boyutu (MSS) kısmıdır. Burada CoAP istemci ve Sunucu birbirlerine MSS boyutlarını bildirdikleri görülmektedir. Aşağıda deneyde CoAPBlip uygulamasında TelosB aygıtlarından veriler toplanarak benzetim gösterilmiştir. Düğümlerin paket boyutu artarsa iletim gecikmesi MSS' in büyüdüğü gözlemlenmiştir.



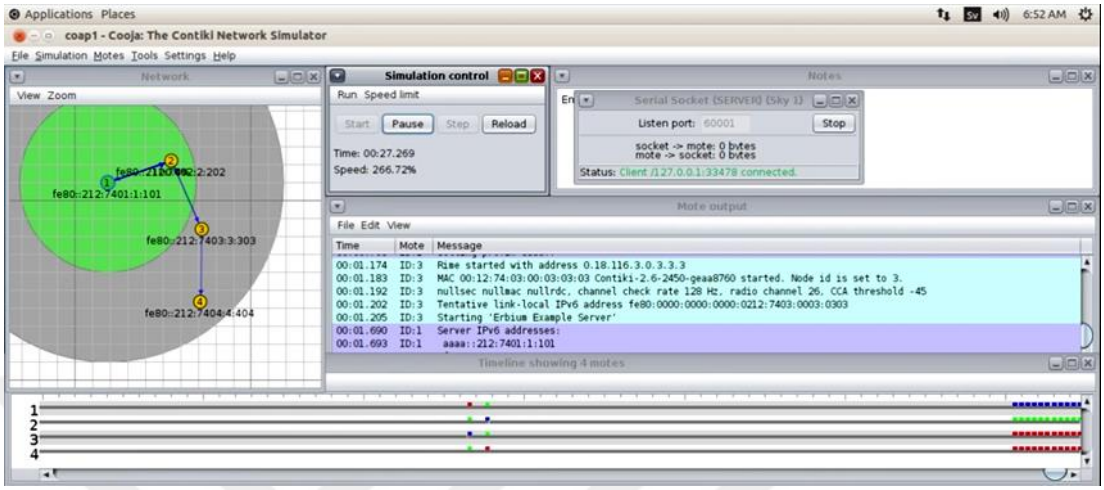
Şekil 3.8. Wireshark paket dinleme programı ile RTT hesaplanması

Bu çalışmada CoAPBlip uygulaması ile CoAP Sunucusu ile İstemcileri arasında CoAP protokolü kullanarak çoklu iletim uygulanmıştır. NesC programlama dili ile yazılan yazılımı TelosB cihazı üzerine yüklenmiştir. Ana istasyondaki düğüm alıcı olarak KAA ortamındaki düğümlerden sıcaklık bilgisi almaktadır. Paket transferini hexadecimal formatta seri porttan bilgisayara aktarılmıştır. Gerçekleştirilen bu uygulamada konsol üzerinde çalışmakta ve yazılımı sadece TinyOS işletim sisteminde 2.1 versiyonda desteklemektedir. CoAP protokolü haberleşmesi için Java 1.6. sürümü kullanılmıştır.

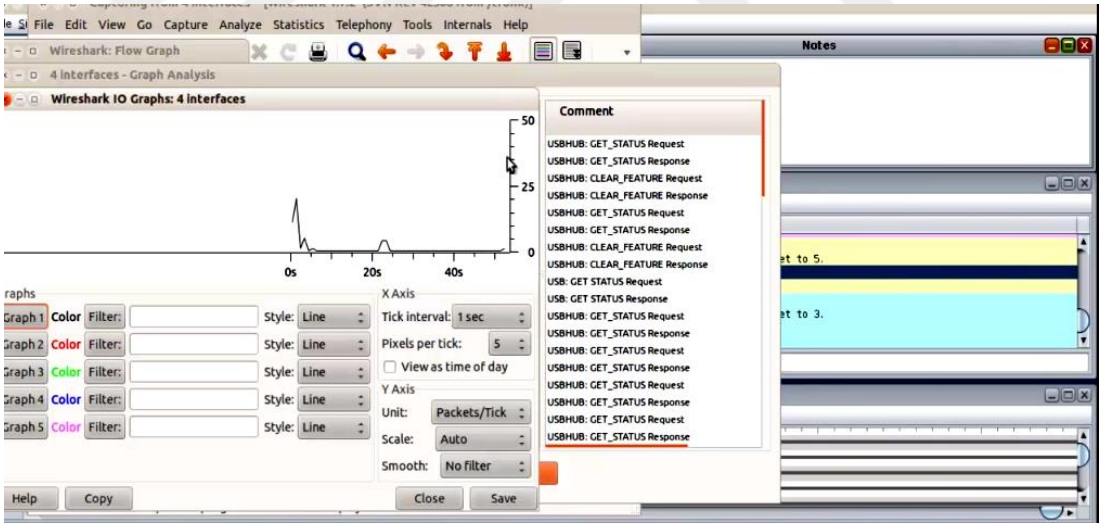
Şekil 3.9. Wireshark paket dinleme programı ile RTT hesaplanması

Ayrıca bu deneyi uyguladıktan sonra düğümlerin mesafe kapalı alanda 30m iken boş alanda 120m kadar radyo frekans alanı oluştuğu test edilmiştir. Buna göre paket kayıpları düğümlerin arasında mesafe artıkça %11 daha paket kaybı meydana geldiği görülmüştür. Paket kaybını artıran bir diğer husus düğüm sayısı artmasıyla ana istasyondaki alıcının kapasitesi artması gerekmektedir. Contiki-OS içindeki Cooja benzetim programı ile çoklu iletim deneyi yapılmıştır. Bu benzetimde Şekil 3.9'da bir

CoAP Sunucusu ve üç adet İstemci düğümünden oluşmaktadır. Ayrıca paket aktarım süreleri Şekil 3.10' deki gibi gösterilmiştir.



Şekil 3.10. Cooja benzetim ortamında TmoteSky düğümü ile çoklu iletim(multicast) yapılması



Şekil 3.11. CoAP protokolü paket aktarım görsel grafiği

3.2.1. KAA Aygıtında Kullanılan AES Sertifika Algoritması

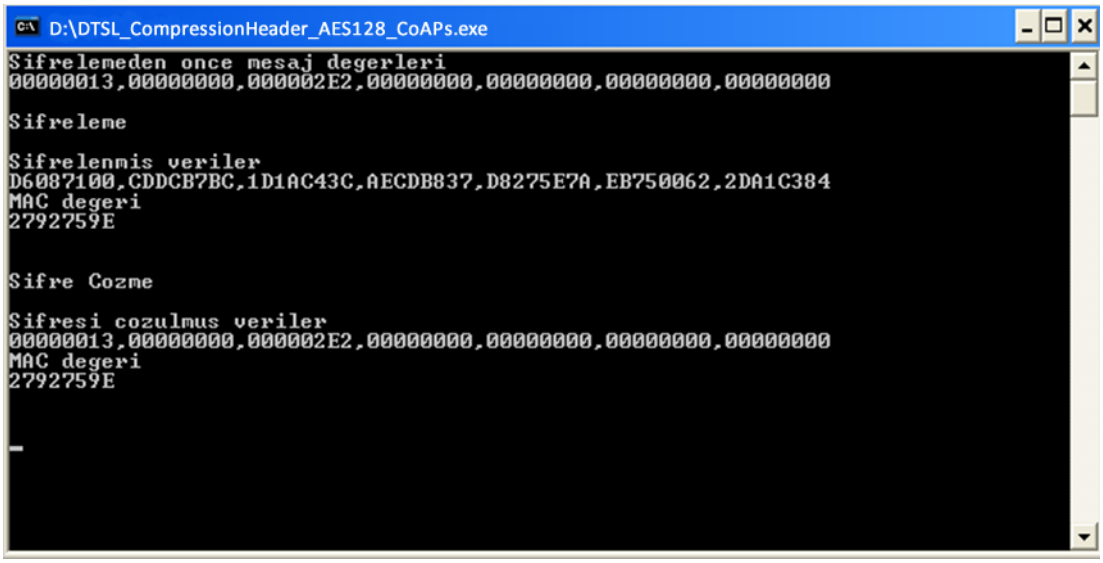
Bu çalışmada AES 128 asimetrik şifreleme algoritması olduğundan paket verilerini şifrelemek için ideal bir algoritmadır. Bu yüzde en çok tercih edilen SSL/TLS sertifikalarında AES algoritması kullanılmıştır. Bu çalışmada Şekil 3.12' de paket önce EncryptStringFromBytes_AES metoduyla şifrelenir. Daha sonra şifrelenmiş veriyi çözen DescriptyStringFromBytes_AES adında metot oluşturulmuştur. Böylece CoAP haberleşirken paketler şifrelenmiş bir biçimde aktarılıp daha sonra çözülmesi Şekil 3.13'de gösterilmiştir.

```
// Paket verisi bir bayt dizisine şifreleme
byte[] encrypted = EncryptStringToBytes_Aes(original, key);

// Şifrelenmiş paketi çözümleme
string roundtrip = DecryptStringFromBytes_Aes(encrypted, key);

//Orijinal verileri ve şifresi çözülmüş verileri görüntüleme
Console.WriteLine("Orjinal Paket:   {0}", original);
Console.WriteLine("Şifrelenmiş veriler (b64-encode): {0}", Convert.ToBase64String(encrypted));
Console.WriteLine("Şifre Çözme: {0}", roundtrip);
```

Şekil 3.12. AES şifreleme algoritması konsole uygulama kodu



Şekil 3.13. AES algoritması ile şifrelenmiş X509 sertifikasının paket mesajı

3.3. DTLS Protokolünde Oturum Başlık Bilgisinin Sıkıştırılması

Birbiriyle iletişimde bulunan CoAP İstemcileri algıladığı verileri ana istasyondaki düğüme aktarırlar. Contiki-OS oluşturulan CoAP sıkıştırma uygulamasını tasarladığımızda dosya içeriğinde otomatik bir şekilde oluşan başlık dosyası, Makefile konfigürasyon dosyası ve modül dosyasını içermektedir. Geliştirilen bu yazılım NesC programlama dili ile kodlanıp düğümlere yüklenmiştir. Makefile ile kullanılacak dosya belirlenir. Başlık dosyasında ise ana istasyonuna hangi düğümün veri gönderdiği, verinin tipi ve veri tanımlanmıştır.

```
COMPENENT=Environment
Include$(MAKERULES)
```

```
typedef struct OrtamMesaji{
        uint16_t kaynak;
        uint16_t tip;
        uint64_t veri;
}
```

Toplam paket boyutu Çizelge 3.2'de gösterildiği gibi 82 bit sıkıştırılmış mesaj paketi başlığı, 16 bit kaynak düğümün adresi, 16 bit algılanan verinin tipi, 64 bit veri ve 16 bit CRC hata ayıklama biti olmak üzere 128 bitten oluşmaktadır.

Çizelge 3.2. Tasarlanan modelin paket formatı

Sıkıştırılmış mesaj paketi başlığı	Kaynak	Tip	Veri	CRC
82 bit	16 bit	16 bit	48 bit	8 bit

3.4. Benzetim Yöntemi İle Elde Edilen Değerler

Bu çalışmada Cooja benzetim ortamında TMoteSky düğümünden alınan değerlere göre yapılmıştır. Kullanılan algıyıcı düğümlerin bellek ihtiyacı ve şifreleme süreleri kullanılan düğüm tipine göre farklılık göstermektedir. Tez çalışması kapsamında TMoteSky ve Telosb gibi düğümler için bu benzetim sonuçları farklılık göstermesi beklenmektedir. Bunun nedeni aygıtların işlem kapasitesi ve donanım kısıtlarının farklı olmasından kaynaklıdır. Cooja benzetim ortamında 2.4 GHz da çalışan TmoteSky düğümü en tutarlı düğüm olduğu için tercih edilmiştir.

- I. Senaryo: CoAP ile paket gönderimi yapan 10, 30, 50 ve 100 düğüm sayısında istemcinin bellek kullanım değerleri Çizelge 3.3' de verilmiştir.
- II. Senaryo: Normal DTLS CoAP ile paket gönderimi yapan 10, 30, 50 ve 100 düğüm sayısında bellek kullanım değerleri Çizelge 3.4'de verilmiştir.
- III. Senaryo: Sıkıştırılmış DTLS başlığıyla haberleşen CoAP düğümleri 10, 30, 50 ve 100 düğüm sayısında bellek kullanım değerleri Çizelge 3.5'de verilmiştir.

Çizelge 3.3. CoAP protokolü ile haberleşen düğümlerin bellek kullanım değerleri

Gerçekleştirilen Durum	Düğüm Sayısı			
RAM [Bayt] Değerleri				
	10	30	50	100
Çalıştırma 1	32030	112389	130758	132171
Çalıştırma 2	33432	107347	123574	123548
Çalıştırma 3	33739	105478	123199	123081
Çalıştırma 4	29375	106103	122791	123752
Çalıştırma 5	32132	107111	123081	124584
Çalıştırma 6	33045	105640	122412	123244
Çalıştırma 7	32726	105276	122120	123632
Çalıştırma 8	33477	105238	121328	122043
Çalıştırma 9	29305	104905	120678	121976
Çalıştırma 10	31715	104951	121135	121632
Ortalama	32098	106444	123107	123966
Standart Sapma	1518	2138	2707	2871

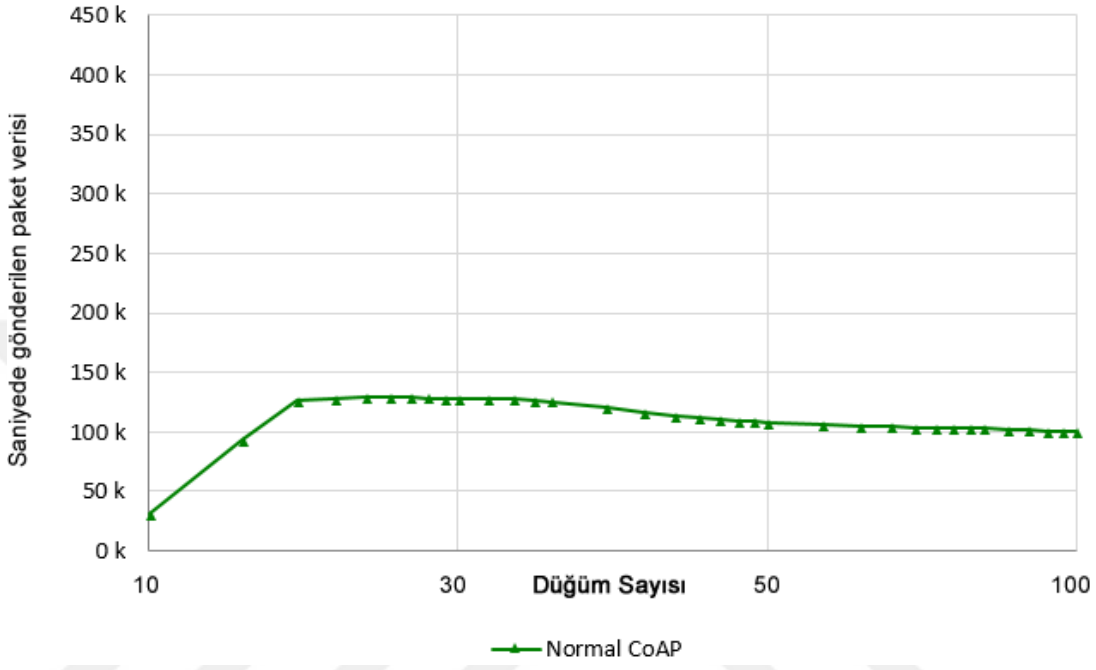
Çizelge 3.4. CoAPs yöntemiyle haberleşen düğümlerin bellek kullanım değerleri

Gerçekleştirilen Durum	Düğüm Sayısı			
RAM [Bayt] Değerleri				
	10	30	50	100
Çalıştırma 1	27097	97487	427706	426860
Çalıştırma 2	28983	91530	427986	434226
Çalıştırma 3	27653	91964	427403	433421
Çalıştırma 4	27515	91213	420365	427948
Çalıştırma 5	28018	91897	426837,5	432535
Çalıştırma 6	28111	91368	427380,4	433490
Çalıştırma 7	27239	91565	428791	433227
Çalıştırma 8	27363	91498	428570	434031
Çalıştırma 9	29344	91655	427292	433459
Çalıştırma 10	27368	91435	427220	433860
Ortalama	27870	92162	426956	432306
Standart Sapma	719	1788	2272	2502

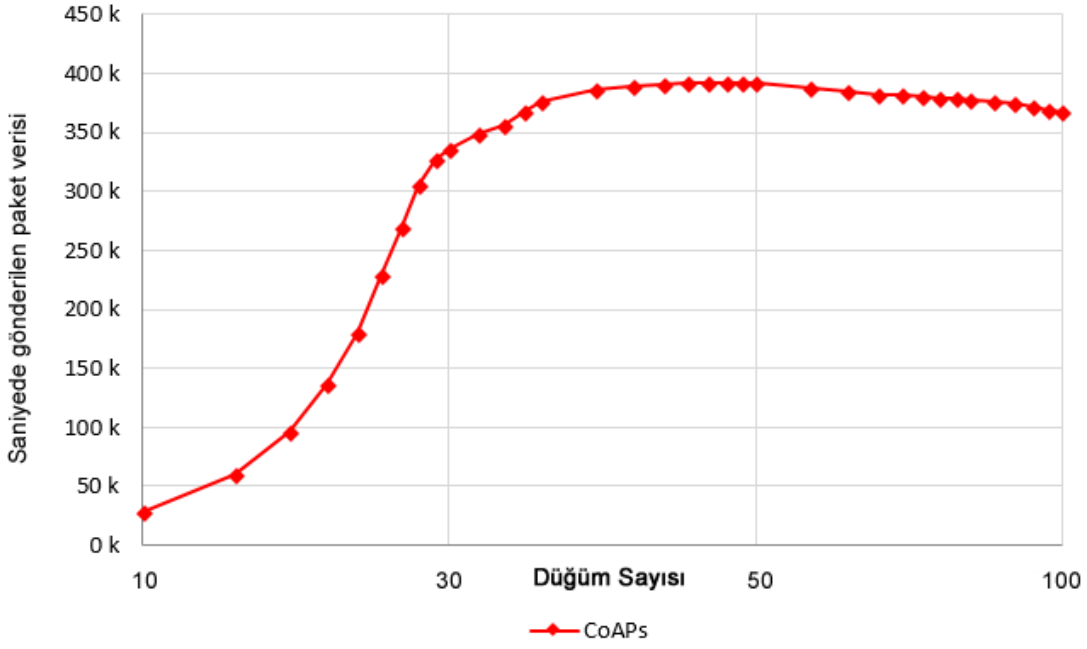
Çizelge 3.5. CoAP protokolü ile haberleşen düğümlerin bellek kullanım değerleri

Gerçekleştirilen Durum	Düğüm Sayısı			
RAM [Bayt] Değerleri				
	10	30	50	100
Çalıştırma 1	28242	58836	383900	427706
Çalıştırma 2	27585	59607	384726	427986
Çalıştırma 3	27804	60026	384016	427403
Çalıştırma 4	27208	59844	379882	420365
Çalıştırma 5	28243	59278	384426	426837
Çalıştırma 6	28312	60753	383979	427380
Çalıştırma 7	28076	58938	385728	428791
Çalıştırma 8	28053	59374	385466	428570
Çalıştırma 9	28309	62430	384088	427292
Çalıştırma 10	28128	63358	384790	427220
Ortalama	27996	60245	384100	426956
Standart Sapma	343	1440	1528	2272

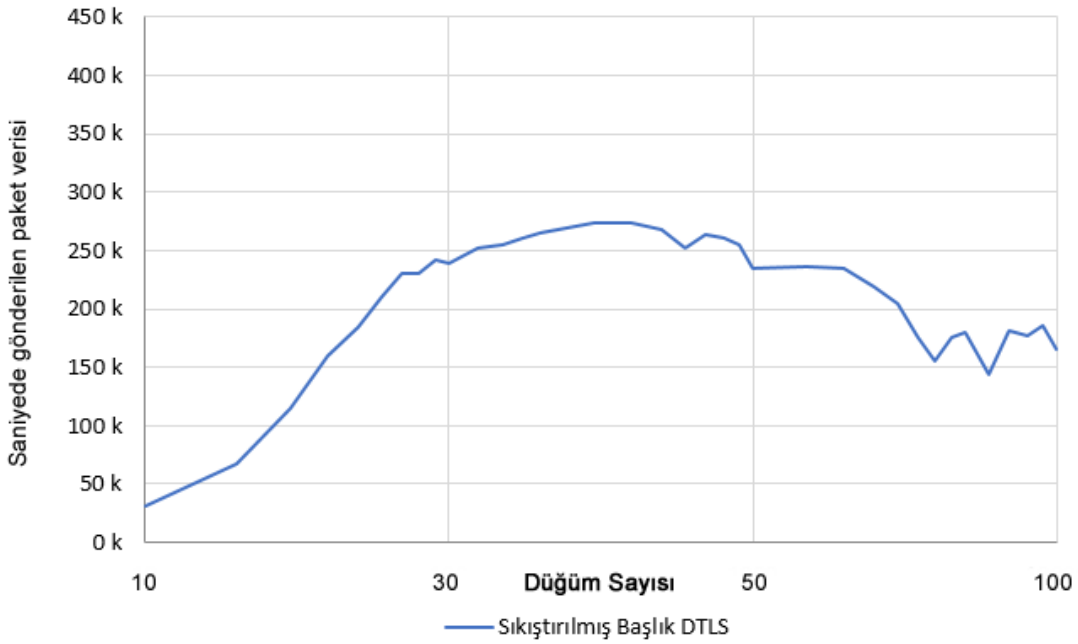
Şekil 3.14'deki grafikten CoAPs yöntemi Şekil 3.14'deki normal CoAP göre bellek kullanım oranında artış tespit edilmiştir. Bu yöntemde şifreleme yapıldığı için bellek kullanım miktarı arttığı görülmüştür.



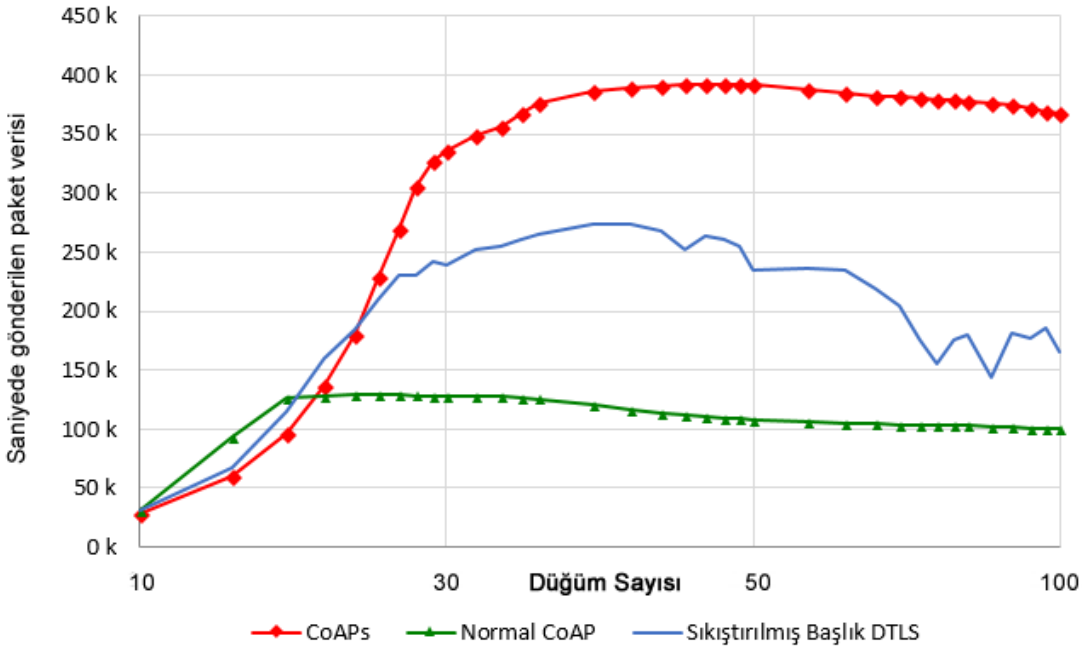
Şekil 3.14. CoAP ile test sonuçlarının grafiksel gösterimi



Şekil 3.15. CoAPs ile test sonuçlarının grafiksel gösterimi



Şekil 3.16. Sıkıştırılmış başlık bilgisi ile CoAPs test sonuçlarının grafiksel gösterimi

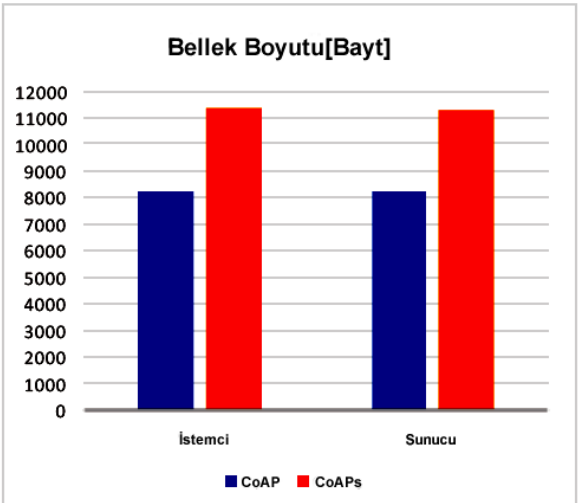


Şekil 3.17. Senaryo sonuçlarının grafiksel karşılaştırılması

Benzetim sonuçlarının değerlendirilmesini Şekil 3.17'deki grafiğe baktığımızda tüm senaryoların bellek kullanım değerleri gösterilmiştir. CoAPs güvenlik yöntemini kullanmak normal CoAP protokolüne göre bellek kullanımı büyük oranda artış olacağını göstermiştir. Bu yüzden bellek kullanımı düşürmek ve güvenlik-hızlı haberleşme arz dengesini yakalamak için sıkıştırılmış DTLS başlık yöntemi uygulanmıştır. Grafiğe baktığımızda düğüm sayısı artıkça önerilen yöntemle beraber bellek kullanımı oranını başarıyla düştüğü görülmektedir. Ayrıca bu çalışmada, mevcut algılayıcı düğümlerinin çoğunun gerekli kodu doldurmak için yeterli ROM ile birlikte gelmemesi nedeniyle ROM'a kıyasla RAM' e daha fazla ihtiyaç vardır. Çizelge 3.6' da düşük ROM tabanlı algılayıcı düğümü TmoteSky, sadece 48 KB toplam ROM'a sahipti ve bu proje için maksimum kod boyutu 20 KB civarındaydı. Bu nedenle Contiki-OS' daki dahil olmak üzere toplam kod boyutu yaklaşık 45 KB olmuştur.

Çizelge 3.6. TelosB düğümlerin ROM ve RAM ihtiyaçları

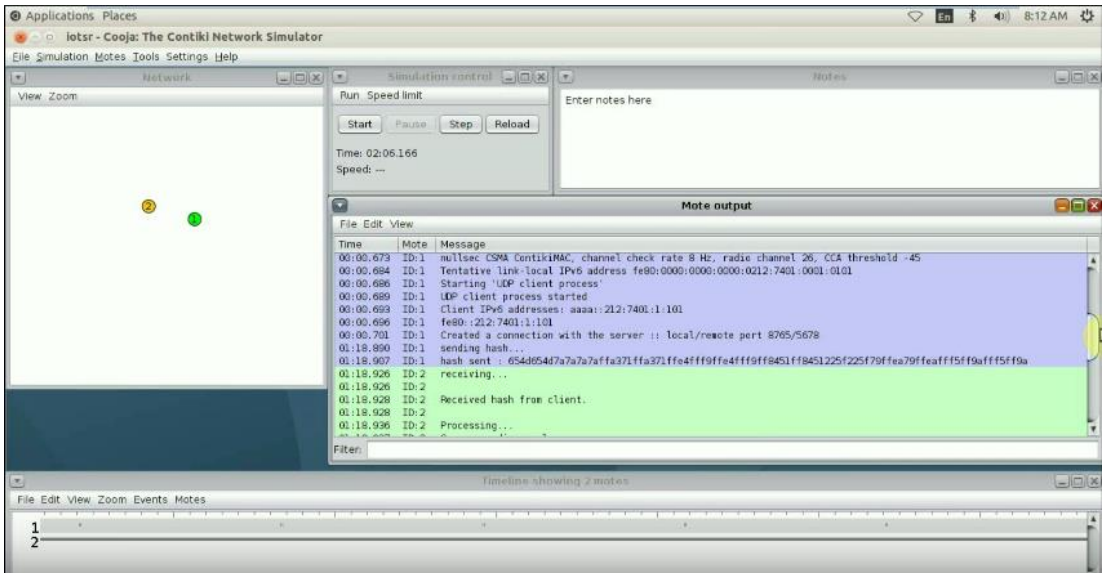
Tek Düğüm Aygıtı İçin	ROM [Bayt]	RAM [Bayt]
DTLS Kripto (SHA-256, CCM, AES)	6592	2870
DTLS	10662	992
Contiki OS	32145	4983
CoAP	82	586
DTLS Sıkıştırma	2826	258
Toplam	60899	9676



Şekil 3.18. CoAP ve CoAPs arasındaki bellek performans karşılaştırılması

BÖLÜM 4. SONUÇLAR

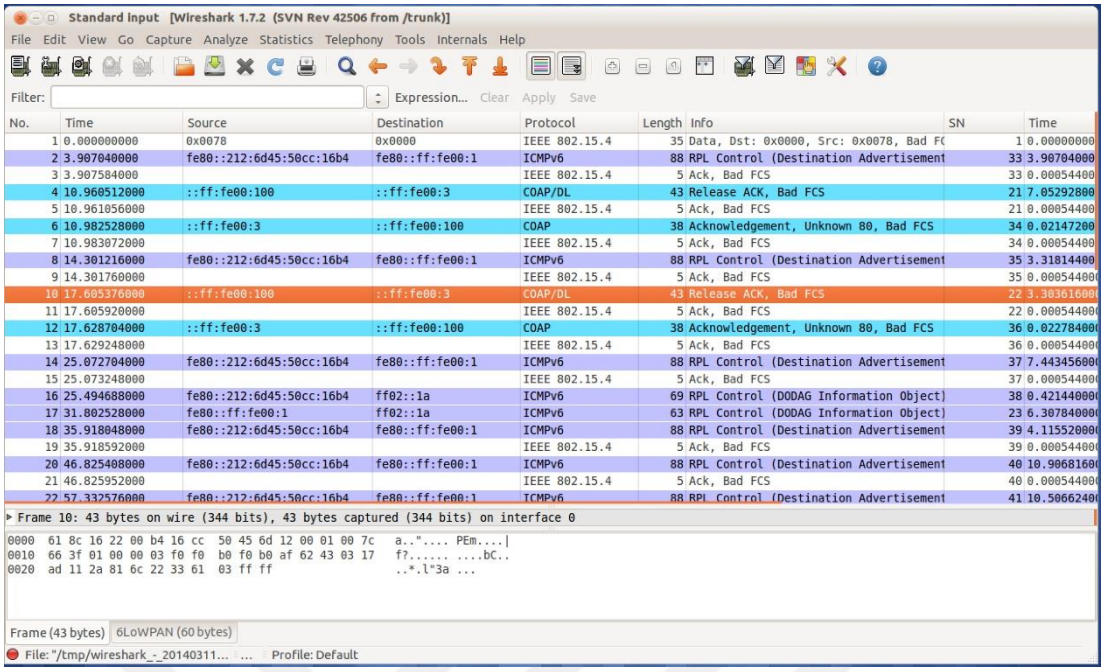
Tez çalışmasında günümüzde IoT' de kullanılan haberleşme protokolleri hakkında bilgi verilerek güvenli CoAP (CoAPs) haberleşme protokolü tasarlanmıştır. Güvenli CoAP, gerçek IoT ortamında özellikle kaynak kısıtlı cihazların temel ihtiyacıdır. Bu yüzden DTLS standart protokolü ile CoAP tümleştirme yapılarak güvenli paket gönderimi sağlanmıştır. Sistemin normal CoAP ile CoAPs arasındaki performans farkı incelenmiştir. Çalışmanın birinci bölümünde IoT haberleşme protokolü güvenliği ilgili bilgi verilmiştir. İkinci bölümünde IoT güvenliğinin sağlanması, IoT ekosistemi ve IoT'de kullanılan haberleşme protokolleri çalışması incelenmiştir. Üçüncü bölümde CoAP ve DTLS protokolünün yapısı, katmanları ve çalışma mimarisi anlatılmıştır. Dördüncü bölümde CoAP ile DTLS protokolü tümleştirme yapılarak güvenli CoAP tasarımı gerçekleştirilmiştir. Sistem bir istemci ve bir sunucudan oluşan düğümler ile kablosuz algılayıcı ağı Şekil 4.1' deki gibi Cooja benzetim ortamında gösterilmiştir. Ayrıca gerçekleştirilen tasarımda DTLS protokolün sıkıştırılmış başlık bilgisi sunulmuştur. Sistemin benzetim uygulamasında çalışma senaryoları sonucunda elde edilen analiz ve grafikler sunulmuştur.



Şekil 4.1. CoAPs ile Cooja benzetim ortamında TmoteSky düğümünü haberleşmesi

Elde edilen sonuçlar ve bulgular şunlardır;

- KAA'da dolaşan verilerin şifrelenmesinde kullanılan TLS sertifika algoritmaları çalışma hızları farklı gösterilmiştir. 50 tane CoAP istemcisi ile CoAP sunucusu arasında bir gidiş-dönüş (RTT) oluşturuldu. CoAP sunucusuna POST talebi geldiğinde ilk önce güvenli bir bağlantı kurması için %3 daha geç cevap verdiği belirlenmiştir. Bu da çok aynı anda 10 fazla istek olduğunda sunucu tarafında isteklerin gerçekleşip gerçekleşmediğini göstermiştir.
- Geliştirilen protokolde hem güvenlik artırılmış hem de KAA için önem teşkil eden bellek kullanımı Şekil 4.2' deki gibi kriterlerde diğer göre daha iyi sonuçlar elde edilmiştir. Gecikme kriterinde ise CoAP' a oranla ihmal edilebilir bir düzeyde artış göstermiştir.
- Düğümlerin kapsama alanları düştükçe paket kayıplarından dolayı oturum açarken zaman aşımına uğradığı belirlenmiştir.
- CoAPs protokolü CoAP protokolüne göre paket aktarımındaki bellek ihtiyacı arttığı belirlenmiştir. Bununda ana sebebi güvenli haberleşme kriterlerini artırmakla haberleşme maliyetleri yükseltmiştir. Bu nedenle dengeyi yakalamak adına başlık sıkıştırma yöntemi veya farklı yöntemler günümüzde uygulanabileceğini göstermiştir.
- Cooja benzetim ortamında düğüm sayısı arttıkça DTLS protokolündeki kimlik doğrulanma süresi arttığı gözlenmiştir.
- Geliştirilen bu çalışma kablosuz algılayıcı ağların güvenliği özellikle askeri ve sağlık alanlarda verilerin güvenli bir şekilde gönderilmesi görülmüştür.
- Wireshark paket izleme programı ile düşük kapasiteli cihazların paket verilerini DTLS ve CoAP protokollerinin performans değerlerini ölçebildiği belirlenmiştir.



Şekil 4.1. WireShark paket izleme programı ile kablosuz ağ düğümünün izlenmesi

KAYNAKÇA

[1] M. Rahman, B. Carbunar, U. Topkara, Secure Management of Low Power Fitness Trackers, IEEE Transactions on Mobile Computing, vol. 15, pp. 447 – 459, February 2016.

[2] K. Chris, S. Naveen, W. David, TinySec: A Link Layer Security Architecture for Wireless Sensor Networks, SenSys'04, Baltimore, Maryland, USA, November 3–5, 2004.

[3] H. Rikard, Lightweight Message Authentication for the Internet of Things", KTH Royal Institute of Technology, Stockholm, Sweden, 2014.

[4] J. King, A Distributed Security Scheme to Secure Data Communication between Class-0 IoT Devices and the Internet, Master Thesis, Luleå University of Technology Department of Computer Science, pp. 33-46, Sweden, 2015.

[5] M. Suresh, P. Saravana, Sundararajan, Data Security in IoT Environment, 2-7, TVP, 2005.

[6] D. A.Camarero., Performance analysis of IPv4/IPv6 protocols over the third generation mobile network, Sweden, 2014.

[7] D. J. Lloret, I. Bosch, S. Sendra, A. Serrano, A wireless sensor network for vineyard monitoring that uses image processing, Sensors, vol. 11(6), pp. 6165–6196, July 2011.

[8] Anna Ha´c: Wireless Sensor Network Designs, ISBN 0-470-86736-1, 2003

[9] A. Luoto, K. Systä, Iot application deployment using request-response pattern with MQTT, 2016.

[10] OASIS, "MQTT Version 3.1.1.," 29 October 2014. [Online]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html, February 2016.

[11] Information technology-message queuing telemetry transport (mqtt) v3.1.1. Standard, International Organization for Standardization, Geneva, CH, April 2016.

[12] M.T. Özsu, P. Valduriez, Principles of Distributed Database Systems; Springer: Berlin, Germany, 2011.

[13] I.E. Agiriano, I. Calvo, F. Pérez, O.G. Albeniz, "Mapping Different Communication Traffic over DDS in Industrial Environments", Information Systems and Technologies (CISTI),6th: 1-6, 2011

[14] M. Richards. "Understanding the Differences Between AMQP and JMS." In: NFJS Magazine 3, URL: http://www.wmrichards.com/amqp.pdf, pp. (26, 39, 40, 42, 43), May 2011.

[15] Information technology-advanced message queuing protocol (amqp) v1.0 specification. Standard, International Organization for Standardization, Geneva, CH, October 2014.

[16] Z. Shelby, K. Hartke, C. Bormann, The Constrained Application Protocol (CoAP), Internet Request for Comments, vol. RFC 7252 (Proposed Standard), Janurary 2014, [Online], Available: http://www.rfc-editor.org/rfc/rfc7252.txt

[17] CoRE Working Group. Constrained Application Protocol (CoAP), Internet-Draft draft-ietf-core-coap-11, 2012.

[18] Z. Shelby, K. Hartke, C. Bormann, The constrained application protocol (coap), 2014.

[19] A. Yegin, Z. Shelby, CoAP Security Options, *Internet Draft*, vol. Draft-yegin-coap-securityoptions-00, Oct. 2011 [Online].

[20] H. Yu, J. He, T. Zhang, P. Xiao, Y. Zhang. Enabling end-to-end secure communication between wireless sensor networks and the internet. World Wide Web, pp. 1–26, 2012.

[21] I.F., Su, W., Sankarasubramaniam, Y., Çayırcı, E., A survey on sensor networks, IEEE Communications Magazine, 40(8), 102-114, 2002.

[22] S. Li, L. lan, Wireless Ad hoc network technologies [J], ZTE communication technology, 2002:09-12.

[23] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In Local Computer Networks, Proceedings 2006 31st IEEE Conference, pp. 641–648, IEEE, 2006.

[24] S. Raza, D. Trabalza, T. Voigt. 6LoWPAN Compressed DTLS for CoAP. In: Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference, pp. 287–289 (cit. on pp. 4, 65) 2012.

[25] J. E. Aranguren, G. Modeling of the data transportation network of a multi-hop data-content-sharing home network, IEEE Trans. Consum. Electron, pp. 61, 31–38, 2015.

[26] J. E. Aranguren, G.; Mazzara, M. Analysis of the data transportation multi-hop network for an intelligent environment. J. Ambient Intell. Smart Environ, pp. 8,205–218, 2016.

[27] A. Liu, P. Ning. TinyECC: A configurable library for elliptic curve cryptography in wireless sensor networks. In 7th International Conference on Information Processing in Sensor Networks (IPSN'08), Washington, DC, USA, 2008.

[28] S. Raza, D. Trabalza, T. Voigt. Poster Abstract: 6LoWPAN Compressed DTLS for CoAP. In proceedings of 8th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'12), 16-18 May 2012, Hangzhou, China.

[29] T. Kothmayr, C. Schmitt, W. Hu, M. Brunig, and G. Carle, A DTLS based end-to-end security architecture for the internet of things with twoway authentication, in Proc. IEEE 37th Conf. Local Comput. Netw. Workshops, pp. 956–963, October 2012.

[30] K. Hartke, H. Tschofenig, A DTLS Profile for the Internet of Things draft-hartke-dice-profile-00, April 2013, http://tools.ietf.org/html/draft-hartke-dice-profile-00

[31] S. Keoh, S. Kumar, O. Garcia-Morchon, E. Dijk, A. Rahman, "DTLS-based Multicast Security for Low-Power and Lossy Networks (LLNs) draft-keoh-dice multicast-security-05, IETF work in progress, 2014.

[32] S. Raza, D. Trabalza, T. Voigt. 6LoWPAN Compressed DTLS for CoAP. In Proceedings of the 8th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS 2012), Hangzhou, China, May 2012.

[33] History of the SSL/TLS Protocol Suite, http://cromwell-intl.com/cybersecurity/ssl-tls.html, accessed: 2016-08-16.

[34] Jain, R. AES Key Expansion, 22-24, 2011.

[35] B. A. Hakhamaneshi, Hardware Implemantation of the Advanced Encryption Standart (AES) Algorithm Using System Verilog, Master Thesis, California State University, Sacramento, USA, pp. 35-41, 2009.

[36] İnternet: TÜBİTAK UEKAE, Açık Anahtar Altyapısı Eğitim Kitabı, http://www.kamusm.gov.tr/dosyalar/kitaplar/aaa/ 2012.

[37] tinyos2.x, http://www.tinyos.net/tinyos2.x/doc. [Ziyaret Tarihi: Nisan 2018].

[38] TinyNode, http://www.tinynode.com/. [Ziyaret Tarihi: Nisan 2018].

[39] Q. W. Li, Q. Sun, L. Zhu, H. Liu, Y. RestThing: A Restful Web Service Infrastructure for Mash-Up Physical and Web Resources. In Proceedings of the 9th International Conference Embedded and Ubiquitous Computing (EUC) 2011, Melbourne, Australia, pp. 197–204, October 2011.

[40] M. Luk, G. Mezzour, A. Perrig, V. Gligor, Minisec: a secure sensor network communication architecture, Proceedings of the 6th international conference on Information processing in sensor networks, pp. 479–488, New York, USA, 2007.

[41] I.E. Bagci, M. R. Pourmirza, S. Raza, U. Roedig, T Voigt. Codo: Confidential data storage for wireless sensor networks. In 8th IEEE International Workshop on Wireless and Sensor Networks Security (WSNS), Las Vegas, Nevada, USA, October 2012.

[42] P. Gutmann. Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS). Internet Requests for Comments 7366, http://tools.ietf.org/html/rfc7366. 2014.

[43] S. Gueron. AES-GCM Software Performance on the Current High End CPUs as a Performance Baseline for CAESAR Competition. Presented at DIAC 2013, 11-13 August 2013, Chicago, USA, http://2013.diac.cr.yp.to/slides/gueron.pdf.

[44] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson. Twofish: a 128-bit Block Cipher. 1st AES Conference, California, USA, Aug 1998. http://csrc.nist.gov/encryption/aes/.

[45] J. Kelsey, Kohno, T. Schneier, B. Amplified boomerang attacks against reduced-round mars and serpent. In Fast Software En- cryption, pp. 13–23, Springer, 2001.

[46] W. Wu, L. Zhang, LBlock: a lightweight block cipher. In Applied Cryptography and Network Security pp. 327–344, Springer, 2011.