İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

AĞ İLETİŞİMLERİNDE TEMEL YENİLİKÇİ ÇÖZÜMLERİN STANDARTLAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Muhammed Salih KALKAN

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

AĞUSTOS 2023

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

AĞ İLETİŞİMLERİNDE TEMEL YENİLİKÇİ ÇÖZÜMLERİN STANDARTLAŞTIRILMASI

YÜKSEK LİSANS TEZİ

Muhammed Salih KALKAN
(504191579)

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

Tez Danışmanı: Dr. Öğr. Üyesi Gökhan SEÇİNTİ

AĞUSTOS 2023

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

STANDARDIZATION OF BASIC INNOVATIVE SOLUTIONS IN NETWORK COMMUNICATIONS

M.Sc. THESIS

Muhammed Salih KALKAN
(504191579)

Department of Computer Engineering

Computer Engineering Programme

Thesis Advisor: Dr. Öğr. Üyesi Gökhan SEÇİNTİ

AUGUST 2023

İTÜ, Lisansüstü Eğitim Enstitüsü'nün 504191579 numaralı Yüksek Lisans Öğrencisi Muhammed Salih KALKAN, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı "AĞ İLETİŞİMLERİNDE TEMEL YENİLİKÇİ ÇÖZÜMLERİN STANDARTLAŞTIRILMASI" başlıklı tezini aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Tez Danışmanı :	**Dr. Öğr. Üyesi Gökhan SEÇİNTİ** İstanbul Teknik Üniversitesi	.....
Jüri Üyeleri :	**Dr. Öğr. Üyesi Mehmet Tahir SANDIKKAYA** İstanbul Teknik Üniversitesi	.....
	Dr. Levent ÇARKACIOĞLU Aselsan	.....
		.....

Teslim Tarihi : **26 Mayıs 2023**
Savunma Tarihi : **30 Ağustos 2023**

Eşim Aybüke'ye,

ÖNSÖZ

Tez çalışmamda bana danışmanlık eden, fikirlerimi dikkatle dinleyip beni yönlendiren, bilgisiyle ve tecrübesiyle bana yol gösteren, bana her zaman ilgi ve alakayla yaklaşan, bana yardımcı olabilmek için elinden gelen her şeyi yapan Dr. Öğr. Üyesi Gökhan SEÇİNTİ hocama sonsuz teşekkürlerimi sunuyorum.

Yüksek lisans öğrenimim boyunca, tüm süreçlerde bana desteğini esirgemeyen, her türlü zorluğu ve sıkıntıyı benimle birlikte hissedip bana moral veren, bana sonsuz sevgisini gösterip bana değerli olduğumu hissettiren, hayatın her alanında beni motive edip başarılarımın arkasındaki gizli kahraman olan sevgili eşim Aybüke BOY KALKAN'a; Kendilerini gururlandırmaktan daima onur duyduğum, ailelerinin bir parçası olmaktan dolayı kendimi çok şanslı hissettiğim, ne olursa olsun benim arkamda olan canım annem Tuba KALKAN'a, canım babam Mehmet Ali KALKAN'a, canım kardeşim Ahmet Taha KALKAN'a binlerce kez teşekkür ederim.

Yüksek lisans eğitimim için bana sağladığı imkanlardan dolayı, bir personeli olduğum ASELSAN A.Ş'ye çok teşekkür ederim. Son olarak, yüksek lisans eğitimim boyunca beni motive eden ve yol gösteren Dr. Levent ÇARKACIOĞLU'na ayrıca çok teşekkür ederim.

Ağustos 2023

Muhammed Salih KALKAN
(Bilgisayar Mühendisi)

İÇİNDEKİLER

KISALTMALAR

OSI	**:** Open Systems Interconnection
AMQP	**:** Advanced Message Queuing Protocol
MQTT	**:** Message Queuing Telemetry Transport
TCP	**:** Transmission Control Protocol
UDP	**:** User Data-gram Protocol
IP	**:** Internet Protocol
QUIC	**:** Quick UDP Internet Connections
MCP	**:** Messaging Control Protocol
ASCII	**:** American Standard Code for Information Interchange
GB	**:** Gigabyte
MOM	**:** Message-oriented middle-ware
IoT	**:** Internet of Things
MAC	**:** Media Access Control
ms	**:** Milliseconds
KB	**:** Kilobyte
GB	**:** Gigabyte
Gbps	**:** Gigabit Per Seconds
DES	**:** Data Encryption Standard

SEMBOLLER

f	**:** Özellik
t	**:** Zaman
O()	**:** Big O Notasyonu

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

AĞ İLETİŞİMLERİNDE TEMEL YENİLİKÇİ ÇÖZÜMLERİN STANDARTLAŞTIRILMASI

ÖZET

Ağ iletişimlerindeki problemler oldukça eskiye dayanır. Bu problemleri çözmek için birçok çalışma yapılmıştır. Bu çalışmalar, günümüzde OSI model olarak adlandırdığımız, katmanlı bir iletişim yapısını ortaya çıkarmıştır. Bu katmanlardan birisi uygulama katmanıdır. Mesajlaşma ile ilgili problemler, bu katmana aittir. Dolayısıyla, mesajlaşma ile ilgili özellikler bu katmanda kullanılır. Bazı mesajlaşma özelliklerini standartlaştırmak için, bazı uygulama katmanı protokoller oluşturulmuştur. AMQP, MQTT vb. protokoller, uygulama katmanı protokollerine örnektir. Bu araştırmada da, temel yenilikçi çözümler uygulama katmanında değerlendirilir.

Uygulamalar, mesajlaşma ile ilgili sorunları farklı şekillerde çözmektedir. Bazı özellikler uygulama koduyla, bazıları kütüphanelerle ve bazıları da protokollerle standardize edilerek sağlanır. Uygulama koduna eklenen mesajlaşma özelliklerinin her uygulama için tekrar tekrar yazılması gerekmektedir. Her uygulama için gerekli mesajlaşma özelliklerinin kodlarının tekrar tekrar yazılması, iş gücü kaybına, hata olasılığına, kodun her seferinde artan karmaşıklığına neden olur. Mesajlaşma sorunlarını kütüphane kodları ile çözmek, bu kütüphanenin diğer tüm uç noktalarla paylaşılmasını gerekli kılar. Bu nedenle mesajlaşma özelliklerinin bir protokol ile standardize edilmesi gerekmektedir. Bu çalışmada, yerel ağlarda ve IoT'de kullanılmak üzere temel yenilikçi özellikleri standartlaştırarak iş gücü kazancı sağlanması, uygulama kodunun karmaşıklığının azaltılması, çözümlerin her uç nokta için ortaklanması amaçlanmıştır. Bir protokol standardı oluşturmak için, protokollere ait özelliklerin arkaplan bilgisine ihtiyaç vardır. Bu yüzden öncelikle, ikili-metin protokoller, iletişim modelleri, merkezi-merkeziyetsiz yaklaşımlar gibi arkaplan bilgileri incelenmiştir.

İkili protokoller, verileri ikili olarak ileten protokollerdir. Metin protokolleri, verileri Unicode veya ASCII olarak ileten protokollerdir. İkili protokoller, verilerin daha küçük boyutlarda iletilmesini sağladığı için performans açısından daha iyidir. Metin protokolleri, verileri daha büyük boyutlarda iletmesine karşın ikili protokollere kıyasla kolayca hata ayıklanabilir ve veriler insan tarafından okunabilirdir. Hem yüksek performans özelliği, hem verinin okunabilir olma özelliğine sahip olmak için, izleyici uç noktanın, ikili verilerin metin karşılıklarını bilmesi gerekir. Ayrıca ikili protokoller için bayt sırası (endianness) önemliyken, metin protokolleri için bayt sırası önemli değildir. Cihazın endianness tipi little-endian veya big-endian olabilir. İkili protokollerde, farklı endianness'e sahip iki cihaz iletişim kurduğunda, verilerin serileştirilmesinden önce ve verinin serisini çözümleme işleminden önce verilerin bayt adreslemesi tersine çevrilmelidir. Bu problemlerin çözümleri, uygulama katmanında

standartlaştırılırsa, geliştiricilerin bu problemleri tekrar tekrar çözmeye çalışmasına gerek kalmaz.

Sunucu-istemci modeli, birden fazla istemci uç noktasının tek bir sunucu uç noktasından hizmet talep ettiği bir modeldir. Yayınla-abone ol modeli, yayıncı ve abone uç noktalarının merkezi bir mesaj yönelimli ara yazılım aracılığıyla mesaj iletimlerini sağlayan bir modeldir. Uç noktalar, konulara abone olur veya mesajları yayınlar. Mesaj aracısı, yayınlanan mesajları, mesaja abone olan uç noktalara iletir. Mesaj aracısı, gevşek bağlantı ve esneklik sağlar. Uç noktalar, birbirlerinin varlığından bağımsız olarak mesajlaşmaya devam eder. Transformatörler ve filtreler, mesaj aracısı üzerinde çalışabilir. Gevşek bağlantı aynı zamanda bir dezavantajdır. Yayıncı uç noktaları, abone uç noktalarının iletişim kurup kurmadığından emin olamaz. Yayıncılar ve aboneler arttıkça, mesaj aracısını aşırı yükleyebilir. Mesaj aracısı, merkezi olduğundan darboğaza neden olabilir. Bu, yatay ölçeklenebilirliği sınırlar. İletileri doğrudan hedef uç noktalara iletmek yerine önce mesaj aracısına iletmek gecikmeyi artırır. Mesaj aracısı ile gelen bu problemlerden kurtulmak için, merkezi olmayan yayınla-abone ol modeline ihtiyaç vardır.

Mesajlaşan uç noktalar için en büyük sorunlardan biri, uç noktalardan birinde mesaj yapılarının güncel olmaması veya yanlış implement edilmiş olmasıdır. Mevcut mesajlaşma protokolleri için, bir bağlantıdaki uç noktaların mesaj yapılarının uyumluluğunu kontrol etmeye yönelik standart bir yaklaşım yoktur.

Bir iletişimde giden ve gelen mesajları izlemek kritik olabilir. Mesaj gönderme noktadan noktaya ise, üçüncü bir izleme uzak uç noktası iletişime dahil edilemez. IP paket başlığındaki hedef IP adresi, noktadan noktaya iletişim için tek bir cihaza ait olmalıdır. Bu problem, uygulama katmanında üçüncü uzak noktalara yönlendirme yapılarak çözülebilir.

Birçok uygulama katmanı protokolü, taşıma katmanındaki bir protokole bağlıdır. Bu da gelecek kullanımları kısıtlayabilir. Örneğin, QUIC protokolü, TCP'nin yerini aldığını varsayalım. Artık TCP implementasyonlarının ortadan kalktığını varsayalım. Bu durumda, düzinelerce TCP tabanlı protokolün yeni bir sürümle QUIC tabanlı olması gerekecektir. Bu yüzden alt protokollerden soyutlanmak, gelecek kullanımlar için önemlidir.

Birden çok protokol kullanmak için birden çok iletişim arabirimi oluşturulmalıdır. Ancak bir protokol, çoklu alt katman protokol ile kullanılabilir olma özelliğine sahip ise, tek bir iletişim arabirimi yeterli olacaktır.

Bu çalışmada, mevcut protokollerin, bu sorunları ne kadar çözdüğüne dair veriler toplandı. Bu sorunları çözen özellikler ile mevcut protokolleri kullanarak bir tablo oluşturuldu. Diğer uygulama katmanı protokollerinin tüm bu özellikleri desteklemediği görülmektedir. Bu nedenle, bu özellikleri sağlayan yeni bir protokole ihtiyaç vardır. Bu protokolün adı mesajlaşma kontrol protokolüdür (MCP). MCP'nin hedeflediği kullanım alanı daha çok yerel ağ iletişimleridir. MCP, daha çok yerel ağ iletişimleri, asenkron iletişimler, non-stateless iletişimler ve gömülü sistemlerde kullanılabilecek özelliklere yoğunlaşmıştır.

MCP'nin alt katman protokollerinden bağımsız olması için ve çoklu alt protokollerle kullanılabilmesi için MCP'nin iki bileşeni vardır: MCP Adaptörü ve iletişim arayüzü.

MCP Adaptörü, MCP'nin ön koşullarını sağlamak için gereklidir. Alt protokollerin işlevlerini kullanmak için iletişim arayüzü gereklidir. Böylece MCP alt protokollerden bağımsız hale gelir ve birden fazla alt protokol ile kullanılabilir.

MCP'de iki mesaj sınıfı vardır: MCP Standart Mesajı, MCP Uygulama Mesajı. MCP, MCP standart mesajları olarak adlandırılan, uygulama kodundan bağımsız yerleşik mesajlara sahiptir. 5 tür standart mesaj vardır: El Sıkışma Mesajı, Kalp Atışı Mesajı, Rol Başvuru Mesajı, Abone Olma Mesajı, Abonelikten Çıkma Mesajı. İstemciler, kullanıcı tanımlı mesajların yapılarını el sıkışma istek mesajı ile JSON formatında gönderir. Böylece uç noktaların mesaj uyumlulukları kontrol edilir. Sunucu, endianness tipini el sıkışma yanıt mesajı ile gönderir. İstemci, sunucunun endianness tipini öğrenir. İstemci ve sunucunun endianness türleri farklıysa, istemci verilerin bayt sıralamasını otomatik olarak değiştirir. Bağlantının canlı olup olmadığını tespit etmek için periyodik olarak kalp atışı mesajı gönderilir. Bir istemci, bir mesaja abone olmak için ya da bir mesajın aboneliğinden çıkmak için Abone Olma Mesajı ve Abonelikten Çıkma Mesajını kullanır.

MCP uygulama mesajları, uygulama kodunda tanımlanan mesajlardır. Dört tür uygulama mesajı vardır: İstek-Yanıt Mesajı, Olay Mesajı, Başlangıç Mesajı, Rapor Mesajı. İstek-yanıt mesajları için, yalnızca ilgili istek mesajı alındığında ilgili yanıt mesajı oluşturularak iletişim sağlanır. Olay mesajları, bir olayın tetiklenmesi ile iletilir. Olay mesajları tüm bağlı abone istemcilerine gönderilir. Başlangıç mesajı, aslında bağlantı kurulduğunda tetiklenen bir olay mesajıdır. Rapor mesajı, aslında zamana göre tetiklenen bir olay mesajıdır.

Yetkilendirme için rol tabanlı erişim kontrol yöntemi kullanılır. İstemcilerin MCP bağlantısında rolleri vardır. İstemcilerin rolleri, mesajlaşma arayüzündeki mesajların erişilebilirliğini belirler. Sunucu, her mesaj için hangi istemci rollerinin erişebileceğini belirler. Rollerin istemcilere atanmasını ise, admin rolündeki istemci gerçekleştirir. Noktadan noktaya iletişimde mesajları izlemek isteyen istemcilerin rolü, izleme rolüdür. İzleyici rolü, iletilerin erişilebilirliğinden bağımsızdır. Noktadan noktaya iletişimdeki tüm mesajlar monitör istemcisine iletilir. İzleme istemcisi, iletişime katılmak için bir bağlantı isteği gönderir. Monitör, bağlantı kurma aşamasında el sıkışma mesajı ile mesaj yapılarını alır ve iletişimdeki ikili verilerin metin karşılıklarını öğrenir. Böylece veriler ikili olarak iletilse de, metin olarak görüntülenebilir.

Uygulama katmanında oluşturulan MCP protokolü, mesajlaşma problemlerini protokol kodunda çözerek problemlerin çözümünü standardize eder. Diğer uygulama katmanı protokolleri, MCP'nin çözdüğü tüm sorunları çözemez. Bu nedenle, MCP fark yaratır. MCP kullanılırsa, bu çalışmada belirtilen çözümlerin uygulama kodunda olmasına gerek kalmaz. Böylece uygulama kodunun karmaşıklığı azaltılmakta ve mesajlaşma özelliklerinde oluşabilecek hatalar ortadan kaldırılmaktadır. MCP sadece mesajlaşma için birçok özellik sunmakla kalmaz, aynı zamanda performansa da önem verir. Performans için, MCP dinamik başlık boyutunu kullanır ve MCP ikili protokoldür. MCP, temel mesajlaşma problemlerine odaklandığı ve performansı önemsediği için yerel ağların yanında IoT'ye de uygulanabilir. Gelecekte IoT alanında MCP'nin kullanılabilmesi için analizler yapılabilir. Sonuç olarak, MCP yenilikçi temel mesajlaşma özellikleri sağlar, bu özellikleri standardize ederek hata olasılığını azaltır ve uygulama kodunun karmaşıklığını azaltır.

STANDARDIZATION OF BASIC INNOVATIVE SOLUTIONS IN NETWORK COMMUNICATIONS

SUMMARY

Network communication problems are very old. Many studies have been done to solve these problems. These studies have revealed a layered communication structure, which we now call the OSI model. One of these layers is the application layer. Messaging related problems belong to this layer. Therefore, messaging-related features are used in this layer. Some application layer protocols have been created to standardize some messaging features. AMQP, MQTT etc. protocols are examples of application layer protocols. Basic innovative solutions are evaluated at the application layer in this research.

The applications solve messaging-related problems in different ways. Some features are provided by application code, some by libraries, some by middle-ware, and some by standardizing with protocols. The messaging features added to the application code need to be written over and over again for each application. Repeatedly writing codes of the necessary messaging features for each application causes loss of workforce, possibility of bugs, increasing complexity of the code in every time. Solving messaging problems in library code requires that this library be need to shared with all other endpoints. Using middle-wares may not be applicable in all areas such as embedded systems. Therefore, messaging features need to be standardized with a protocol.

Binary protocols are protocols that transmit data in binary. Text protocols are protocols that transmit data as Unicode or ASCII. Binary protocols are better in terms of performance. It allows data to be transmitted in smaller sizes. Text protocols transmit data in larger sizes but text protocols are easily debugged and easily read compared to binary protocols. The monitor endpoint must know the text representations of binary data to have both high performance capability and data readability. Also, while byte order (endianness) is important for binary protocols, byte order is not important for text protocols. The endianness type of the device can be little-endian or big-endian. In binary protocols, When two devices with different endianness communicate, the byte addressing of the data must be reversed before serialization and deserialization. If the solutions to these problems are standardized at the application layer, developers don't need to try to solve these problems over and over.

The server-client model is a model where multiple client endpoints request services from a single server endpoint. The publish-subscribe model is a model that enables publisher and subscriber endpoints to send message through a central message-oriented middle-ware. Subscriber endpoints subscribe to topics and publisher endpoints publish messages. The message broker forwards the message to the subscriber endpoints that have subscribed to the message. The message broker provides loose coupling

and flexibility. Endpoints continue to messaging regardless of each other's existence. Transformers and filters can operate on the message broker. Loose coupling is also a disadvantage. Publisher endpoints cannot be sure whether subscriber endpoints are communicating. As publishers and subscribers increase, it can overload the message broker. Message broker can cause bottleneck since it is centralized. This limits horizontal scalability. Forwarding messages to the message broker first, rather than forwarding them directly to the target endpoints, increases latency. A decentralized publish-subscribe model is needed to get rid of these problems that come with the message broker.

One of the biggest problems for messaging endpoints is that the message structures are out of date or implemented incorrectly in one of the endpoints. There is no standard approach to checking the compatibility of message structures of endpoints in existing messaging protocols.

Monitoring outgoing and incoming messages in a communication can be critical. If sending messages is point-to-point, a third monitoring remote endpoint cannot be involved in the communication. The destination IP address in the IP packet header must belong to a single device for point-to-point communication. This problem can be solved by routing messages to third remote points in the application layer.

Many application layer protocols depend on a protocol at the transport layer. For example, If QUIC is to replace TCP, dozens of TCP-based protocols will need to become QUIC-based with a new version. Thus, abstracting from sub-protocols is important for future uses.

In order to use multiple protocols, multiple communication interfaces must be created. if a protocol is capable of being used with multiple sub-layer protocols, a single communication interface is sufficient.

Data were collected on how current protocols solve these problems in this study. A table was created using existing protocols with features that solve these problems. It seems that other application layer protocols do not support all these features. Therefore, there is a need for a new protocol that provides these features. The name of this new protocol is the Messaging Control Protocol (MCP).

For the MCP to be used with different sub-protocols, the MCP has two components: the MCP Adapter and the communication interface. The MCP Adapter is required to provide the prerequisites for the MCP. The communication interface is required to use the functions of the sub-protocols. Thus, the MCP becomes independent of the sub-protocols. Moreover, MCP can be used with multiple sub-protocols.

There are two classes of messages in the MCP: MCP Standard Message, MCP Application Message. MCP has built-in messages independent of application code, called MCP standard messages. There are 4 types of standard messages: Heartbeat Message, Handshake Message, Subscribe Message, Subscribe Message. Heartbeat message is periodically sent to detect that the connection is keep alive and the quality of connection. Clients sends the structures of the user-defined messages as JSON format with the handshake request message. Thus, the message compatibility of the endpoints is checked. The server sends its endianness type with the handshake response message. The client learns the endianness type of the server. If the endianness types of the client

and the server are different, the client reverse the data according to its endianness type. A client sends a subscribe message to subscribe to a message and sends a unsubscribe message to unsubscribe to a message.

MCP application messages are messages defined in the application code. There are four types of MCP application messages: Request-Response Message, Event Message, Initial Message, Report Message. Request-response messages are transmitted with a point-to-point communication model. Event messages are transmitted with a point-to-multipoint communication model. The transmission of event messages is triggered by the server-side application code. Event messages are sent to all connected subscriber clients. The initial message is actually a event message that is triggered by establishing connection. The report message is actually a event message that is triggered by time. A message can have more than one message type. In other words, a message can be all of message types: a response message, an event message, an initial message and a report message.

The clients have roles in the MCP connection. The roles of the clients determine the accessibility of messages in the messaging interface. The server determines which client roles can access for each message. Accessibility of a message can be given to multiple client roles. The default role of each client is common role. The default accessibility of each message is common role. An unlimited number of client roles can be generated in a messaging interface. These roles are not created by the protocol. These roles differ from the standard roles, which are the common role and monitor role, as they are user-defined. Message-based authorization is provided with these roles. The role of clients that want to monitor messages in a point-to-point communication is the monitor role. The Monitor role is independent of the accessibility of the messages. All messages in point-to-point communication are forwarded to the monitor client. Monitor client sends a connection request to join the communication. The monitor learns the message structures with the handshake message during the client connection establishment phase. Thus, it learns the text representations of the binary data in communication. Therefore, although data is transmitted in binary, the data is displayed as text.

The MCP protocol, created at the application layer, standardizes the solution of the problems by solving messaging problems at the protocol code. Other application layer protocols cannot solve all the problems that MCP does. Therefore, MCP makes the difference. If MCP is used, it is expected that the application code will have almost no code for the features required for messaging. Thus, the complexity of the application code is reduced and the bugs that may arise in the messaging features are eliminated. MCP not only offers many features for messaging, but also cares about performance. For performance, MCP uses dynamic header size and MCP is binary protocol. MCP is suitable for use in many usage areas because it focuses on basic messaging problems and it cares about performance. However, MCP alone may not be sufficient in some areas. MCP-based upper layer protocols may be needed. In the future, MCP can be developed and brought to a level that can be sufficient on its own in all areas. As a result, MCP provides innovative basic messaging features, standardizing these features, reduces the possibility of bugs, and reduces the complexity of application code in messaging applications.

1. GİRİŞ

Protokoller, veri iletimi için geliştirilen özellikleri ve algoritmaları standardize eden sözleşmelerdir. Ağ iletişimlerindeki tüm veri iletimleri, bu protokollere göre gerçekleştirilir. Bu sözleşmeler, tüm uç noktaların aynı dili kullanması gibidir. Tüm uç noktaların aynı dilde konuşabilmesi için, veri iletimi özelliklerinin protokoller ile standartlaştırılması, Madsen(1986)'in [1] açıkladığı gibi bir gerekliliktir.

Mesajlaşma özelliklerinin standartlaştırılması, OSI modelindeki uygulama katmanında geliştirilen protokollerle gerçekleştirilir. Mesajlaşma özelliklerinin standardize edilmesi, uygulama kodunun karmaşıklığını azaltır, hata olasılığını düşürür, bu özelliklerin tekrar tekrar implement edilmesini önler [2]. Uygulama katmanı protokollerinin kernel seviyesi protokol yerine kullanıcı seviyesi protokol olması, taşınabilirlik ve esneklik bakımından daha avantajlıdır [3]. Protokollerde standardize edilen özellikleri ve algoritmaları daha iyi anlamak için, veri iletimi problemlerinin arkaplan bilgisine ihtiyaç vardır.

1.1 Arkaplan Bilgisi

Protokollerin veri iletimi iki türe ayrılabilir: mesaj yönelimli veya akış yönelimli. Mesaj yönelimli veri iletimi, mesaj sınırlarını korumayı garanti eder. Bir uç noktadaki tek bir gönderme, diğer uç noktadaki tek bir alıma eşdeğerdir. Akış yönelimli veri iletimi, mesaj sınırlarını korumayı garanti etmez. Dolayısıyla, akış odaklı bir protokol üzerinden mesajlaşma yapılıyorsa, uç noktaların mesaj sınırlarının yeniden bulunması gerekir [4].

Mesaj odaklı veri iletiminin avantajı, mesaj sınırlarını yeniden bulmaya gerek olmamasıdır. Bu nedenle, işlemciye ek bir yük getirmez. Akış odaklı veri iletiminin avantajı, ağın daha verimli kullanılmasıdır. Örneğin, küçük verilerin ağ üzerinden ayrı paketler halinde gönderilmesi gerekmez. Hepsi tek bir pakette birleştirilebilir. Benzer şekilde, büyük veriler daha küçük paketlere bölünerek gönderilebilir. Böylece yeniden

iletim gerektiğinde, tam büyük verinin yeniden iletimi yerine küçük verilerin yeniden iletimi gerçekleştirilir.

Mesajlaşan uygulamalar için veri iletiminin, ağ katmanında akış yönelimli olması ancak uygulama katmanında mesaj yönelimli olması ideal çözümdür. Ağ katmanındaki akış yönelimli verilerin uygulama katmanında mesaj yönelimli olabilmesi için alıcı uç noktanın, verileri ayrıştırarak mesaj sınırlarını bulması gerekir. Verileri ayrıştırmak için iki popüler yöntem vardır: uzunluk verileriyle ayrıştırma ve sabit verilerle ayrıştırma [4]. Birinci yöntemde gönderilen mesaja mesajın uzunluk değeri eklenir. Alıcı uç nokta, uzunluk değerine göre mesaj sınırlarını bulabilir. İkinci yöntemde gönderilen mesajın başına sabit mesaj başlığı eklenir veya mesajın sonuna sabit mesaj sonlandırıcı eklenir. Böylece, alıcı uç nokta mesaj sınırlarını bulabilir.

İkili protokoller, verileri ikili olarak ileten protokollerdir. Metin protokolleri, verileri Unicode veya ASCII olarak ileten protokollerdir. İkili protokoller performans açısından daha iyidir [5]. Verilerin daha küçük boyutlarda iletilmesini sağlar. Metin protokolleri, verileri daha büyük boyutlarda iletir, ancak metin protokolleri, ikili protokollere kıyasla daha kolay hata ayıklanabilirdir ve okunabilirdir [6]. Ayrıca ikili protokoller için bayt sırası (endianness) önemliyken, metin protokolleri için bayt sırası önemli değildir.

Sunucu-istemci modeli, birden fazla istemci uç noktasının tek bir sunucu uç noktasından hizmet talep ettiği bir modeldir [7]. İstek mesajları yalnızca istemciler tarafından, yanıt mesajları ise yalnızca sunucular tarafından oluşturulabilir. İstemciler sunucuya bir istek mesajı gönderir ve sunucu bir yanıt mesajı döndürür. Kullanım alanlarına örnek olarak e-posta, World Wide Web vb. verilebilir.

Mesaj kuyruğu modeli, üretici ve tüketici uç noktalarının merkezi bir mesaj odaklı ara yazılım (MOM) aracılığıyla mesaj göndermesini sağlayan bir modeldir [8]. Mesaj aracısı, mesaj yönelimli ara yazılımın yapı taşıdır. Üretici tarafından üretilen mesaj, mesaj aracısında bir kuyruğa alınır. Mesaj kuyruğunu dinleyen bir tüketici, mesajı kendi uç noktasına alır, mesajı değerlendirir ve mesaja cevap verir. Mesaj kuyruğu modeli, cloud computing gibi alanlarda kullanılır.

Publish-subscribe modeli, merkezi bir mesaj yönelimli ara yazılım aracılığıyla, yayın ve abone uç noktalarının mesaja gönderilmesini sağlayan bir modeldir [9,10]. Abone uç noktaları, mesaj aracısı aracılığıyla ilgilenilen konulara abone olur. Yayıncı uç noktaları, mesajları yayımlar ve bunları mesaj aracısına gönderir. Mesaj aracısı, mesajı, mesaja abone olan abone uç noktalarına iletir. Publish-subscribe modelinde, mesaj kuyruğu modelinden farklı olarak, mesaj birden fazla uç noktaya iletilir. Kullanım alanlarına örnek olarak IoT, log yönetimi vb. verilebilir.

Mesaj aracısı kullanmanın avantajları ve dezavantajları vardır [11]. Mesaj aracısı, gevşek bağlantı ve esneklik sağlar. Uç noktalar birbirinden habersizdir. Uç noktalar, birbirlerinin varlığından bağımsız olarak mesajlaşmaya devam eder. Uç noktalar farklı protokoller kullansalar da, mesaj aracısı aracılığıyla birbirlerine mesaj gönderebilirler. Yayıncı ve abone uç noktalarındaki mesaj yapıları uyumsuz olsa bile, dönüştürücüler mesaj aracısı üzerinde çalışabilir.

Mesaj aracısının gevşek bağlantı sağlaması, her zaman avantaj değildir [12]. Yayıncı uç noktaları iletilerin teslim durumundan haberdar değildir. Yayıncı uç noktaları, abone uç noktalarının iletişim kurup kurmadığından emin olamaz. Bunu garanti etmek için daha sıkı bağlantı gereklidir. Yayıncılar ve aboneler arttıkça, mesaj aracısını aşırı yükleyebilir. Mesaj aracısı, ağda darboğaza neden olabilir. Bu, yatay ölçeklenebilirliği azaltır.

İletileri doğrudan hedef uç noktalara iletmek yerine önce mesaj aracısına iletmek gecikmeyi artırır [13]. İleti yapılarındaki uyumsuzluğu gidermek için ileti aracısına dönüştürücüler koymak düşük performansa neden olabilir. Binlerce uç noktalı bir örnek için, aynı işlemci üzerinde binlerce dönüştürücü çalıştırmak, mesaj aracısı sunucusunda düşük performansa neden olabilir. Dönüştürücülerin uç noktaların kendi tarafında çalışması daha verimli olacaktır.

1.2 Tezin Amacı

Bu çalışmanın amacı, yerel ağlarda, non-stateless ve asenkron iletişimlerin gömülü sistemlerde kullanılmasının sonucunda ortaya çıkan mesajlaşma problemlerinin çözümlerini bir standart kullanarak uygulamaktır. Bu çalışmada, uygulama

katmanı problemlerine odaklanılmıştır. Uygulama kodundaki çözümlerin, bir uygulama katmanı protokolüne aktarılması hedeflenmektedir. Böylece bu çözümlerin uygulamalara tekrar tekrar eklenmesi engellenir. Bunun sonucunda uygulama kodunun karmaşıklığı azaltılır, iş gücü kazancı sağlanır, hata olasılığı düşürülür, diğer uç noktaların aynı çözümleri uygulaması garanti edilir.

2. SORUN BİLDİRİMİ

Uç noktalarda karşılaşılan bazı mesajlaşma sorunları uygulama kodunda çözülmüştür. Fakat bazı problemler, ancak diğer uzak uç noktalarda da kodlama yapılarak çözülebilir. Diğer tüm uç noktalar için çözümlerin doğru şekilde uygulandığından emin olunamaz. Çözümler standartlaştırılmadığından, uygulama kodundaki her uç noktada sorunların doğru şekilde çözülüp çözülemediği bilinemez [1].

2.1 Mesaj Sınırları Problemi

Akış yöntemiyle mesajlaşan tüm uç noktalar, mesaj sınırlarını korumak için mesajlara ayrıştırma verileri eklemelidir [4]. Akış veri iletişimi her oluşturulduğunda, ayrıştırma kodunun alıcı uç noktaya tekrar tekrar yazılması gerekir. Her iletişim farklı ayrıştırma yöntemleri kullanabileceğinden, ayrıştırma kodları da her seferinde farklılık gösterecektir. Ayrıştırma kodunu tekrar tekrar yazmak, her seferinde hata olasılığına yol açar. Ayrıştırma kodunun tekrar tekrar yazılması iş gücü kaybına neden olur.

2.2 Manipüle Edilebilir Ayrıştırma Kodu Problemi

Akış verilerini ayrıştıran iki uç noktadan biri anlamsız veriler gönderdiğinde, diğer uç noktanın ayrıştırma kodunu bozabilir. Örneğin, akış verilerini mesajın uzunluğuyla ayrıştıran bir mesajlaşma arabirimi olduğunu varsayalım. Bir uç noktanın diğer uç noktaya gönderdiği anlamsız verilerde mesaj uzunluğu değeri 100 GB'a eşit olsun. Bu durumda, ayrıştırma uç noktası, 100 GB veri alınana kadar bloke edilecektir. Bu veriler bellekte tutulmaya çalışılacağından, verileri alan bilgisayarın performansı önemli ölçüde düşecektir.

2.3 İstek-Yanıt Mesajlarının İlişki Problemi

Asenkron iletişimlerde, istek ve yanıt mesajlarının arasındaki ilişki belirsizdir [14]. Mesajlaşan iki uç noktadan biri, aynı istek mesajından farklı içerikte iki tane gönderdiğinde, gönderen uç nokta, gelen iki yanıt mesajına hangi istek mesajının karşılık geldiğini anlayamaz. Farklı içeriğe sahip aynı istek mesajından ikisi, sıralama korunarak alıcı son noktaya ulaşsa bile, alıcı uç nokta gelen ikinci mesajı daha hızlı değerlendirebilir. İkinci mesajın yanıt mesajını birinci mesajın yanıt mesajından daha erken gönderebilir. Örneğin, içeriği 5 olan A mesajı ile içeriği 10 olan A mesajının arka arkaya gönderildiğini varsayalım. Gelen iki yanıt mesajının içeriğinden biri başarılı, diğeri başarısız olsun. İstek mesajını gönderen uç nokta, 5 içerikli A mesajının mı yoksa 10 içerikli A mesajının mı başarısız olduğunu anlayamaz.

2.4 Uyumsuz Mesaj Yapıları Problemi

Mesajlaşan uç noktalar için en büyük sorunlardan biri, uç noktalardan birinde mesaj yapılarının güncel olmaması veya yanlış implement edilmiş olmasıdır. Mevcut mesajlaşma protokolleri için, bir bağlantıdaki uç noktaların mesaj yapılarının uyumluluğunu kontrol etmeye yönelik standart bir yaklaşım yoktur. Uygulama katmanında bir sorun olduğunda, yazılımın mesaj yapılarının uyumsuz olduğunu anlaması zordur. Bu sorun genellikle bir geliştirici tarafından hata ayıklama ile tespit edilebilir. Bu da iş gücü kaybına neden olur.

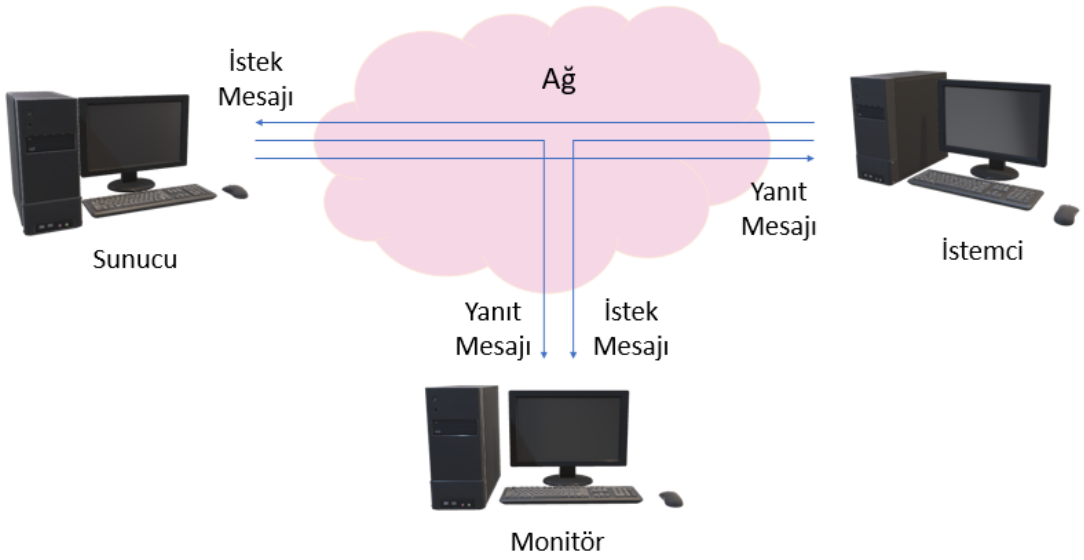
2.5 Sunucu Tarafında İstemci İsteğine Göre Filtreleme Problemi

Bir sunucu, sunucuya bağlanan tüm istemcilere aynı mesajlaşma arabirimini sunacaktır. Bazı uygulamalarda sunucu, sunucuya bağlanan istemcilere periyodik olarak raporlama mesajları veya yayın mesajları gönderir. Ancak, bazı istemciler bu mesajları almak istemeyebilir. İstemciler kendi taraflarında bu mesajları filtreleseler de bu mesajlar arka planda ağ üzerinde gönderilecek ve ağa gereksiz yük getirecektir. Bu nedenle sunucu tarafında istemci kontrollü mesaj filtreleme yapılmalıdır. Ancak

sunucu, bağlı istemcinin hangi mesajları kullanmak istediğini, bağlı istemcinin hangi mesajları filtrelemek istediğini anlayamaz. Bu nedenle, istemci-sunucu modelinde yayınla-abone ol modelinin fonksiyonlarına ihtiyaç duyulabilir.

2.6 Noktadan Noktaya İletişimlerde Mesajların İzlenmesi Problemi

Bir iletişimde giden ve gelen mesajları izlemek kritik olabilir. Şekil 2.1'deki gibi üçüncü bir uzak uç noktadan mesajların izlenmesi istenebilir. Mesaj gönderme noktadan noktaya ise, üçüncü bir izleme uzak uç noktası iletişime dahil edilemez. IP paket başlığındaki hedef IP adresi, noktadan noktaya iletişim için tek bir cihaza ait olmalıdır. Noktadan noktaya iletişim için hedef IP adresine çoklu yayın IP adresi girilemez. Bu nedenle, taşıma katmanındaki bir segment yalnızca hedef cihaza iletilir. Segment, izleme cihazına yönlendirilemez. Bu sorunu çözmek için, mesajları üçüncü bir uç noktaya iletmek, uygulama katmanında çözülebilir. Ancak bu özellik standardize edilmezse her uygulama için tekrar tekrar eklenmesi gerekir.



Şekil 2.1 : Noktadan Noktaya İletişimlerde Mesajları İzleme.

2.7 Mesaj Erişim Yetkileri Problemi

Sunucu, istemciler için bir mesajlaşma arabirimi sağlar. Sunucu, her istemcinin her mesajı kullanabilmesini istemeyebilir. Başka bir deyişle, mesaj erişimlerinin yetkilendirilmesi istenebilir. Ancak sunucu, henüz istemcileri tanımıyor olabilir.

Bu nedenle, kod geliştirme aşamasında sunucu tarafında, istemci kimliğine göre yetkilendirme ve filtreleme yapılamayabilir. Ayrıca sunucu admin olmadığı için yetkileri belirlemede karar mercii olmayabilir. Bir başka deyişle, istemcilerin kimlikleri bilinse bile, o kimliğe sahip istemcinin erişim yetkisinin ne olması gerektiği, sunucu tarafından değil, admin yetkisine sahip başka bir istemci tarafından biliniyor olabilir. Bu yüzden kimliklendirme olmadan yetkilendirme yapmaya ihtiyaç vardır. Rol tabanlı erişim kontrolü [15,16] yöntemi kullanılarak, mesaj erişim yetkileri kimlikler ile değil, roller ile ilişkilendirilebilir. Bu yetkilendirme yaklaşımı protokol seviyesinde standart hale getirilebilir.

Yetkilendirme yönteminin protokol seviyesinde standartlaştırılması beklenirken, kimliklendirmenin protokol seviyesinde standartlaştırılmaya çalışılmamasının sebebi, kimliklendirmelerin her uygulamaya göre özelleşmesi ve çeşitli dış bileşenlerin kullanılmasıdır. Bu yüzden, kimliklendirmenin mesajlaşma protokolü seviyesinde standardize edilmesi doğru bir yaklaşım değildir. Kimliklendirme daha üst seviye bir problemdir. Ancak mesaj yetkilendirme yöntemi, mesajlaşma özelliklerini standardize eden bir protokol ile standart hale getirilebilir.

2.8 Kimlik Bilgisi Olmadan İstemcilere Rol Atama Problemi

Eğer rol tabanlı erişim kontrolü yöntemi kullanılırsa, bu yöntemin adımlarından biri, istemcilere rol atama adımıdır. Sunucu, hiçbir istemciyi tanımadığı için kendi başına rol ataması yapamaz. Rol atamaları için istemcileri tanıyan, kimliklerini bilen, erişim yetkilerinin ne olduğunu bilen, admin rolündeki bir istemciye danışılması gerekir.

2.9 Güncel Olmayan Veri Problemi

Sunucu-istemci modelinde, uç noktalar noktadan noktaya yöntemini kullanarak iletişim kurar. Bu nedenle, diğer istemci uç noktalarında veriler eski olabilir. Sunucuya bağlı istemcilerden biri sunucudaki aygıtın durumunu değiştirecek bir istek mesajı gönderdiğinde, yanıt iletisinin yalnızca bir istemciye gönderildiğini varsayalım. Bu durumda, yalnızca bir istemci, sunucunun üzerinde çalıştığı aygıtın mevcut durumunu

bilir. Bu nedenle, istemci-sunucu modelinde yayınla-abone ol modelinin faydalarına ihtiyaç duyulabilir.

2.10 Endianness Problemi

İkili veri iletimlerinde cihazların endianness tipi önemlidir. Cihazın endianness tipi little-endian veya big-endian olabilir. Verilerin seri hale getirilmesi ve seriden çıkarılması endianness'e göre işlenir. Farklı endianness'e sahip iki cihaz iletişim kurduğunda, seri hale getirme ve serisini kaldırma işleminden önce verilerin bayt adreslemesi tersine çevrilmelidir. Bu işlemler bir iletişim protokolü ile standartlaştırılmadığından uygulama kodunda bu işlemlerin tekrar tekrar yazılması gerekir.

2.11 İkili Protokollerde İletilen Verilerin İnsan Tarafından Okunamaması Problemi

Performans önemli olduğunda, metin protokolleri yerine ikili protokoller kullanılmalıdır. Ancak ikili protokollerde, veri iletimi metin olarak yapılmadığından, iletilen veriler insan tarafından okunabilir değildir. Bu da hata ayıklamayı zorlaştırır. Veriler ikili olarak iletilseler de, metin olarak görüntülenebilecekleri bir yönteme ihtiyaç vardır.

2.12 Bağlantı Kopma Tespiti Problemi

Örneğin bir TCP bağlantısı kurulduktan sonra bir kablo kopması olursa uç noktalar bağlantının koptuğunu tespit edemeyeceklerdir. Uç noktalar bir TCP FIN mesajı almadığından, uç noktalar bağlantının canlı olduğunu varsayar.

2.13 Sabit Boyutlu Başlıklarda Gereksiz Büyük Başlık Problemi

Gereksiz veri yükü, ağ üzerinden iletişimde performansı düşürecektir. Protokole özellikler eklendikçe, mesaj başlığına yeni alanlar eklemek gerekebilir. Mesaj başlığındaki alanların boyutu çok büyük olmasa da her mesaja eklendikleri için ağa

ciddi bir yük bindirir. Başlıktaki bazı alanlar her mesaj için kullanılmasa da, alanların her başlıkta olması gereksiz ek yük getirir. Ek olarak, alanların sabit boyutu olması da ek yükü artırabilir.

2.14 Alt Katman Protokollerine Bağımlılık Problemi

Birçok uygulama katmanı protokolü, taşıma katmanındaki bir protokole bağlıdır. Örneğin, TCP tabanlı AMQP, eğer QUIC ile kullanılmak istenirse, yeni bir AMQP versiyonuna ihtiyaç vardır. QUIC, TCP'nin yerini alacaksa, düzinelerce TCP tabanlı protokolün yeni bir sürümle QUIC tabanlı olması gerekecektir. Gelecekte QUIC yerine daha performanslı bir veri aktarım protokolü geliştirildiğinde QUIC tabanlı tüm protokollerin yeniden güncellenmesi gerekecektir.

2.15 Tek Alt Katman Protokolü Kullanabilme Problemi

Uygulama katmanı protokolleri, tek bir alt katman protokolü ile tasarlanmıştır. Örneğin, bir uygulama katmanı protokolü TCP tabanlı ya da UDP tabanlı ya da QUIC tabanlı vb. olabilir. Ancak iletişim uygulamaları bazı veriler için UDP ve bazı veriler için TCP kullanmak isteyebilir. Birden çok protokol kullanmak için birden çok iletişim arabirimi oluşturulmalıdır.

3. İLGİLİ ÇALIŞMALAR

Çözümlerin standardize edilmemesi, hangi problemlere sebep olduğu Bölüm 2'de anlatılmıştır. Bu nedenle çözümlerin bir protokol ile standardize edilmesi gerekmektedir. Bu bölümde, Bölüm 2'de anlatılan problemleri çözebilen diğer uygulama katmanı protokoller incelenecektir. Diğer uygulama katmanı protokollerinin, problemleri ne kadar çözebildiği yorumlanacaktır. Öncelikle Bölüm 2'de anlatılan problemleri çözebilen özellikler belirlenecektir. Daha sonra bu özelliklerin hangi protokoller tarafından sağlanıp sağlanmadığı çizelgelerle gösterilecektir.

Kullanılacak protokolün yüksek performanslı olması için ikili protokol ve başlık boyutunun dinamik olması gerekir. Ayrıca, İkili protokoller kullanan uç noktaların endianness türlerindeki farklılıkların çözülmesi gerekir. Metin protokolünün avantajları başka yollarla elde edilmelidir. Bağlantı aşamasında, mesaj yapıları uç noktalarla metin olarak paylaşılır. Böylece tüm uç noktalar, ikili verilerin metin eşdeğerini bilebilir. Veri iletimi, insan tarafından okunabilir hale gelir. Böylece hata ayıklaması da kolay olur. Asenkron iletişim için istek-yanıt mesajları arasında korelasyon olmalıdır. Hem istemci-sunucu modelinin hem de yayınla-abone ol modelinin faydalarının hibrit bir modelde birleştirilmesi gerekir.

Noktadan noktaya iletişimin kolay izlenmesi ve hata ayıklaması için mesajların üçüncü bir uzak noktaya iletilmesi gerekir. Mesajların teslim edilip edilemeyeceğini tespit etmek için bağlantı kopukluklarının tespit edilmesi gerekir. Mesaj uyumluluğu problemlerinin tespiti standardize edilmelidir. Mesaj aracısı ile gelen ağdaki darboğaz riski, çakılma riski ve sınırlı ölçeklenebilirlik gibi dezavantajları ortadan kaldırmak için merkeziyetsiz bir çözüm üretilmelidir. Aracısız çözümün, mesaj aracısının faydalarını başka şekillerde sağlayarak konsolide edilmesi gerekir. Tüm bu ihtiyaçlar aşağıdaki gereksinimlere sağlanabilir:

- f 1: Mesaj Odaklı Veri İletimi
- f 2: İkili (Binary) Protokol
- f 3: İstek-Yanıt Mesajları için Korelasyon
- f 4: İstemci-Sunucu Modelinin Desteklenmesi
- f 5: Publish-Subscribe Modelinin Desteklenmesi
- f 6: Mesaj Erişimlerinin Yetkilendirilmesi
- f 7: İki Uzak Uç Nokta Arasındaki Noktadan Noktaya Mesajları İzleme
- f 8: İletilen Verilerin İnsan Tarafından Okunabilir Şekilde Görüntülenmesi
- f 9: Uç Noktaların Endianness Tipine Göre Verilerin Serileştirilmesi
- f 10: Bağlantı Kopukluğu Tespiti
- f 11: Dinamik Başlık Boyutu
- f 12: Mesaj Yapılarının Uyumluluğu Kontrolü
- f 13: Merkeziyetsiz İletişim Modeli
- f 14: Alt Katman Protokollerinden Bağımsızlık
- f 15: Çoklu Alt Katman Protokolü Kullanabilme

3.1 Diğer Uygulama Katmanı Protokollerinin Çözebildiği Problemler

Gereksinimlerin, incelenen protokollerde özellik olarak mevcut olup olmadıklarını tespit etmek amacıyla, protokollerin spesifikasyon dosyaları referans alınarak Çizelge 3.1 oluşturulmuştur [17]–[24]. Her gereksinim için bir özellik numarası atanarak çizelge boyutlarının verimli kullanılması amaçlanmıştır. Çizelgede özellikleri temsilen f sembolü kullanılmıştır.

Çizelge 3.1 : Protokollerin Özelliklerinin Kıyaslanması.

	TCP	SCTP	HTTP/1.1	HTTP/2.0	HTTP/3.0	WebSocket	TCPROS	UDPROS	QUIC	SOAP	AMQP	MQTT	CoAP	STOMP	DDS
f 1		X	X	X	X	X	X	X	X	X	X	X	X	X	X
f 2	X	X		X	X	X			X		X	X	X		X
f 3											X	X	X		
f 4	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
f 5							X	X			X	X		X	X
f 6			X	X	X	X			X	X	X	X	X	X	X
f 7															
f 8			X				X	X		X				X	
f 9															
f 10						X		X			X	X		X	
f 11	X	X				X		X	X		X	X	X	X	
f 12															
f 13	X	X	X	X	X	X	X	X	X	X			X		X
f 14	-	-								X					
f 15	-	-													

4. YÖNTEM

Var olan uygulama katmanı protokollerinden hiçbirinin, Bölüm 2'de açıklanan problemlerin hepsini çözemediği Bölüm 3'te gösterilmiştir. Dolayısıyla tüm problemleri çözebilen yeni bir uygulama katmanı protokolüne ihtiyaç vardır. Bu yüzden Bölüm 2'de açıklanan problemlerin hepsini çözebilen MCP (Messaging Control Protocol) adlı yeni bir protokol oluşturulmuştur.

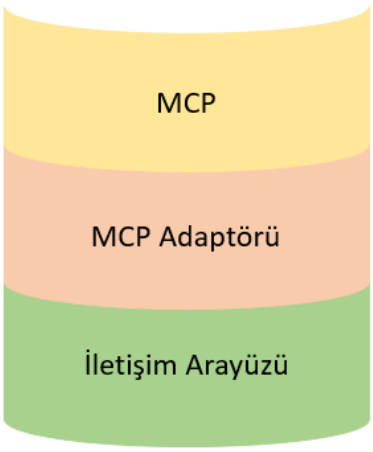
MCP ikili protokoldür, bağlantı yönelimlidir ve veri iletimi mesaj yönelimlidir. MCP, bir uygulama katmanı protokolüdür. MCP transport katmanına ait veri iletimi ile ilgilenmez. Bu nedenle, MCP, veri iletimi ile ilgilenen alt katman protokollerinden bağımsızdır. Bu sayede MCP tüm veri aktarım protokolleri ile birlikte kullanılabilir. MCP'nin kullanılabilmesi için veri iletim katmanında bazı ön koşulların sağlanması gerekmektedir. Bu ön koşullar şunlardır:

- Bilinen Veri Başlangıç Noktası
- Bozulamaz Veri (Checksum Control)
- Mesaj Odaklı Veri İletimi (Akış Odaklı Olmayan)
- Sınırsız Mesaj Boyutu (Teorik Olarak)

4.1 MCP'nin Alt Katman Protokollerinden Soyutlanması

MCP'nin farklı alt protokollerle kullanılabilmesi için MCP'nin iki alt bileşeni vardır: Şekil 4.1'deki gibi MCP Adaptörü ve iletişim arayüzü. MCP Adaptörü, MCP'nin ön koşullarını sağlamak için bir wire protokol olarak tasarlanmıştır. Alt protokollerin işlevlerini kullanmak için iletişim arabirimi tasarlanmıştır. Farklı alt protokollere göre, MCP adaptörü ve iletişim arayüzü farklılaşır. Kısım 2.14'deki problem, MCP Adaptör bileşeni ve iletişim arayüzü bileşeni ile çözülmüştür. QUIC, TCP ve UDP

protokolleri için standart yapılandırmalar mevcuttur. Ek olarak, diğer alt protokol için MCP adaptörü ve iletişim arabirimi yapılandırılabilir.

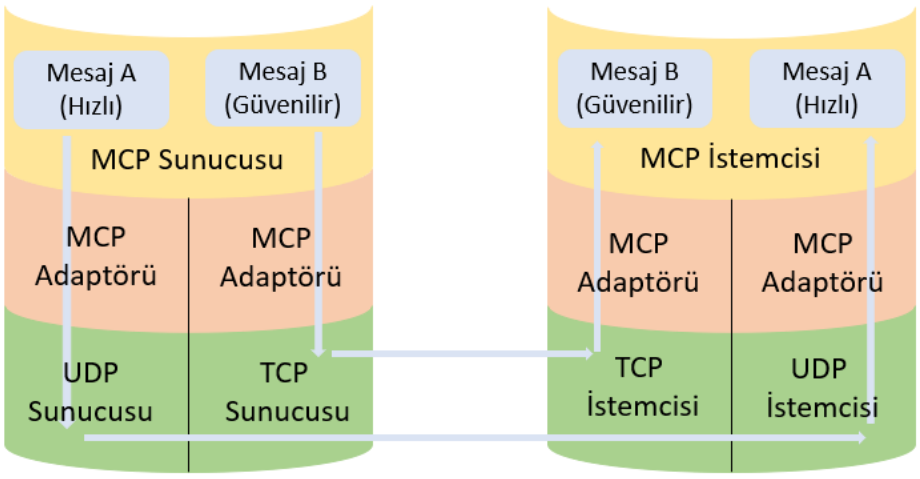


Şekil 4.1 : MCP Bileşenleri.

MCP, birden çok alt protokol kullanmayı destekler. Hangi mesajın hangi alt protokol ile iletileceğini, mesaj için konfigüre edilmiş iletim özelliği belirler. Her protokol için bir iletim özelliği oluşturulur ve numaralandırılır. Veri iletim özellikleri ve ilişkili oldukları protokoller aşağıdaki gibidir:

- Güvenilir (1) ⇒ TCP
- Güvenceli Güvenilir (2) ⇒ QUIC
- Hızlı (3) ⇒ UDP
- Güvenceli Hızlı (4) ⇒ DTLS
- ...
- *Veri İletim Özelliği (#) ⇒ MCP Adaptörü ve İletişim Arayüzü Konfigüre Edilmiş Protokol*

Şekil 4.2'de görüldüğü gibi, mesajlar için belirlenen iletim özelliğine göre mesajlar ilgili protokoller ile iletilebilir. Böylece, Kısım 2.15'teki sorun çözülmüş olur.



Şekil 4.2 : Çoklu Alt Katman Protokolü Kullanılarak MCP Veri İletimi Örneği.

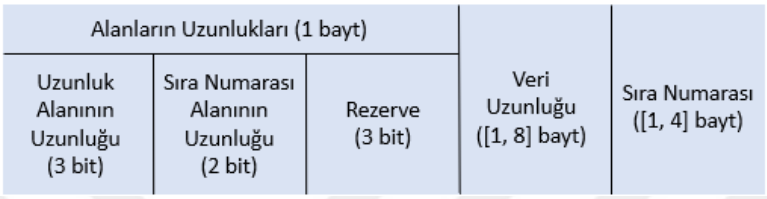
4.2 MCP Adaptörü

MCP adaptör bileşeni, MCP için ön koşulları sağlama amacıyla oluşturulmuş bir wire protokolüdür. MCP için "Bilinen Veri Başlangıç Noktası" ve "Bozulamaz Veri" önkoşulları, data link katmanında ve transport katmanında zaten karşılanmıştır. Ancak MCP ile kullanılmak istenen bir veri iletim protokolü, "Mesaj Odaklı Veri İletimi" ve "Sınırsız Mesaj Boyutu" ön koşullarını karşılamıyor olabilir. Bu ön koşullar sağlanmasa bile, MCP adaptörü sayesinde bu veri iletimi protokolü MCP protokolü ile kullanılabilir. MCP adaptörü, kullanılan veri aktarım protokolünün özelliklerine göre otomatik olarak etkinleşir ve yapılandırılır. Adaptörün, veri aktarım protokolünün aşağıdaki özelliklerini bilmesi gerekir:

- Mesaj Odaklı veya Akış Odaklı Veri İletimi
- Sınırlı veya Sınırsız Mesaj Boyutu
- Güvenilir (Yeniden İletimli) veya Güvenilir Olmayan (Paket Kaybı)
- Sıralı veya Sırasız Veri İletimi

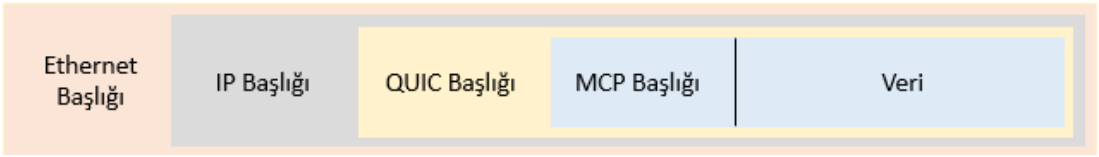
Geliştirici, alt katman protokolünün özelliklerini MCP'ye bildirdikten sonra, bu özelliklere göre MCP adaptörü otomatik olarak yaratılabilir yada yaratılmasına gerek olmaz. MCP adaptörü başlığındaki olası alanlar Şekil 4.3'teki gibidir:

- Alanların Uzunluklar
  - Uzunluk Alanının Uzunluğu
  - Sıra Numarası Alanının Uzunluğu
  - Rezerve Alan
- Veri Uzunluğu
- Sıra Numarası



Şekil 4.3 : MCP Adaptör Başlığı Yapısı.

Veri iletim protokolü mesaj yönelimliyse ve sınırsız mesaj boyutunu destekliyorsa adaptöre gerek yoktur. Dolayısıyla bu durumda adaptör başlığı oluşturulmaz. Örneğin, QUIC hem mesaj odaklıdır hem de mesaj sınırı yoktur. Bu yüzden MCP adaptörüne ihtiyaç duymaz. QUIC tabanlı MCP çerçevesinin OSI modeline göre konumu Şekil 4.4'deki gibidir.



Şekil 4.4 : QUIC tabanlı MCP Çerçevesi.

Veri iletim protokolü akış yönelimliyse, Uzunluk Alanının Uzunluğu ve Veri Uzunluğu alanları MCP adaptörü başlığına eklenir. Örneğin, TCP akış yönelimlidir. Bu nedenle mesaj sınırlarının yeniden bulunması gerekir. MCP adaptörü, uzunluk verileriyle mesaj sınırlarının bulunmasını sağlar. Böylece veri, mesaj sınırları korunarak uygulama koduna beslenir ve Kısım 2.1'deki problem çözülür. Ayrıca, ayrıştırma kodu standartlaştırıldığı için ayrıştırma kodu değiştirilemez. Kısım 2.2'deki sorun da çözülür. TCP tabanlı MCP çerçevesinin OSI modeline göre konumu Şekil 4.5'teki gibidir.

Şekil 4.5 : TCP Tabanlı MCP Çerçevesi.

Veri iletim protokolü sınırsız mesaj boyutunu desteklemiyorsa ve veri iletim protokolü sıralı veri iletimini destekliyorsa, Uzunluk Alanı Uzunluğu ve Veri Uzunluğu alanları MCP adaptörü başlığına eklenir. Veri iletim protokolü sınırsız mesaj boyutunu desteklemiyorsa ve veri iletim protokolü sıralı veri iletimini desteklemiyorsa Uzunluk Uzunluğu Alanı, Sıra Numarası Uzunluğu Alanı, Veri Uzunluğu ve Sıra Numarası alanları MCP adaptör başlığına eklenir.

Örneğin UDP, sınırlı mesaj boyutunda veri gönderir. UDP ile büyük mesajlar göndermek zaten risklidir. Ayrıca, sıralı veri iletimi garanti edilmez. Bu nedenle, büyük veriler küçük paketlere bölünerek gönderilmelidir. Alıcı uç nokta, bu paketleri sırayla birleştirir. MCP adaptörü bunu uzunluk verileri ve sıra numarası ile yapar. UDP tabanlı MCP çerçevesinin OSI modeline göre konumu Şekil 4.6'daki gibidir.

Şekil 4.6 : UDP tabanlı MCP çerçevesi.

Veri uzunluğu alanının uzunluğu, veri uzunluğu alanının boyutunu bayt olarak tanımlar. Veri uzunluğu alanının uzunluğu 3 bittir. Veri uzunluğu alanının uzunluğu 0 olamaz. Dolayısıyla 0 değeri, maksimum 8 bayt olan değeri ifade eder. Veri uzunluğu alanının uzunluğu, veri uzunluğu alanının boyutunu 1 bayt olan minimum değere kadar indirebilir.

Sıra numarası alanının uzunluğu değeri, sıra numarası alanının boyutunu bayt olarak tanımlar. Sıra numarası alanının uzunluğu 2 bittir. Sıra numarası alanının uzunluğu 0 olamaz. Dolayısıyla 0 değeri maksimum 4 bayt olan değeri ifade eder. Sıra numarası

alanının uzunluğu, sıra numarası alanının boyutunu 1 bayt olan minimum değere kadar indirebilir.

4.3 İletişim Arayüzü

MCP, iletişim arayüzü ile veri iletim protokollerinden bağımsız hale gelir. Üç tür iletişim arayüzü vardır:

- Sunucu Arayüzü
- İstemci Arayüzü
- Bağlı Uç Nokta Arayüzü

MCP, bu üç arabirimi uygulayan herhangi bir iletişim standardı ile kullanılabilir. UDP, TCP ve QUIC protokolleri için bu üç arabirimin implementasyonu MCP'de standarttır. MCP başka bir protokolle kullanılacaksa, bu üç arabirimin geliştirici tarafından implement edilmesi gerekir. Örneğin, bu üç arayüz SCTP için implement edilebilir.

4.3.1 Sunucu arayüzü

- int Open() : Bu işlev bir sunucu başlatır. Kullanılan soket port numarası döndürülür.
- void Close() : Bu işlev sunucuyu kapatır.
- ConnectedEndpoint Accept() : Bu işlev, bağlanmaya çalışan istemcileri kabul eder. Mevcut iş parçacığı, bir istemci kabul edilene kadar bloke edilir. Kabul edilen istemciler, section 4.3.3'te açıklanan Bağlı Uç Nokta Arayüzü olarak döndürülür.

4.3.2 İstemci arayüzü

- void Connect(string address, int port) : Bu işlev, bir IP adres ve bağlantı noktası numarasıyla belirtilen bir sunucuya bağlanır.
- void Disconnect() : Bu işlev, sunucuyla bağlantıyı keser.

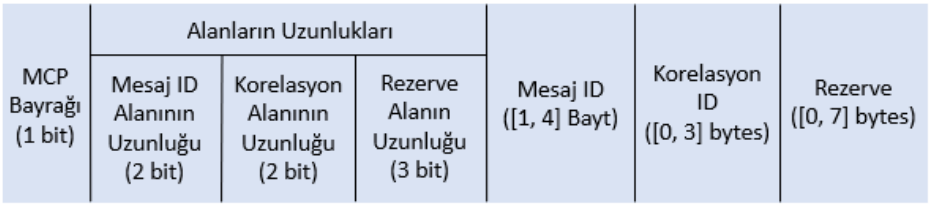
- void Send(byte[] data) : Bu işlev verileri sunucuya gönderir.
- int Receive(byte[] buffer) : Bu işlev, sunucudan gelen verileri belleğe alır. Alınan veri boyutunu döndürür. Mevcut iş parçacığı, veri alınana kadar bloke edilir.

4.3.3 Bağlı uç nokta arayüzü

- void Disconnect() Bu işlev, uzak uç noktadan bağlantıyı keser.
- void Send(byte[] data) : Bu işlev, verileri uzak uç noktaya gönderir.
- int Receive(byte[] buffer) : Bu işlev, uzak uç noktadan gelen verileri belleğe alır. Alınan veri boyutunu döndürür. Mevcut thread, veri alınana kadar bloke edilir.

4.4 MCP Başlığı

MCP protokolü, sorunları çözmek için bazı verilere ihtiyaç duyar. Bu verileri saklamak için bir MCP başlığı oluşturulmuştur. Şekil 4.7'de görüldüğü gibi, MCP başlığında bir bayrak, bir mesaj kimliği alanı, bir korelasyon kimliği alanı ve bu alanların uzunluklarını gösteren uzunluk alanları bulunur.



Şekil 4.7 : MCP Başlık Yapısı.

4.4.1 MCP bayrağı

MCP'de iki mesaj sınıfı vardır:

- MCP Standart Mesajı
- MCP Uygulama Mesajı

MCP, sorunları çözmek için uygulama kodundan bağımsız yerleşik mesajlara sahiptir. Kısım 4.5'te açıklanan MCP standart mesajlar, MCP'nin yerleşik mesajlarıdır. MCP

üzerinden iletilen uygulama mesajları, Kısım 4.6'da açıklanan MCP uygulama mesajları olarak adlandırılır. MCP standart mesajları için MCP bayrağı 0'a ayarlanır. MCP uygulama mesajları için MCP bayrağı 1'e ayarlanır. MCP bayrak alanının boyutu 1 bittir.

4.4.2 Alanların uzunlukları

Sabit boyutlu mesaj başlığı alanlarına sahip olmak, mesaj başlığının büyük olmasına neden olur. Örneğin, mesaj kimliği alanı 4 bayt ile sabitlenirse, toplam mesaj kimliği sayısı 255'ten az olduğunda kullanılmayan 3 bayt, mesaj başlığında gereksiz yer kaplayacaktır. Bu nedenle, mesaj başlığındaki alanların uzunlukları dinamik olarak boyutlandırılmıştır. Mesaj başlık alanlarının boyutu dinamik olduğundan, bu alanların boyutunu tanımlayan alanlara ihtiyaç vardır:

- Mesaj Kimliği Alanının Uzunluğu
- Korelasyon Kimliği Alanının Uzunluğu
- Ayrılmış Alanın Uzunluğu

Mesaj kimliği alanının uzunluğu, mesaj kimliği alanının boyutunu bayt olarak tanımlar. Mesaj kimliği alanının uzunluğu 2 bittir. Mesaj kimliği alanının uzunluğu 0 olamaz. Dolayısıyla 0 değeri, maksimum 4 bayt olan değeri ifade eder. Mesaj kimliği alanının uzunluğu, mesaj kimliği alanının boyutunu 1 bayt olan minimum değere kadar indirebilir.

Korelasyon kimliği alanının uzunluğu değeri, korelasyon kimliği alanının boyutunu bayt olarak tanımlar. Korelasyon kimliği alanının uzunluğu 2 bittir. Korelasyon kimliği alanının uzunluğu minimum değer olan 0 olabilir. Maksimum değer 3, 3 bayt anlamına gelir. Korelasyon Kimliği, yalnızca Kısım 4.6.1'de açıklanan istek-yanıt mesajları için kullanılır. Bu nedenle, diğer mesaj türleri için korelasyon kimliği alanının uzunluğu 0'dır.

Ayrılmış alanın uzunluğunun değeri, ayrılmış alanın boyutunu bayt olarak tanımlar. Ayrılmış alanın uzunluğunun boyutu 3 bittir. Ayrılmış alanın uzunluğu minimum değer olan 0 olabilir. Maksimum değer 7, 7 bayt anlamına gelir.

Alan uzunluklarının toplam boyutu 7 bittir. Bu uzunluk alanları ile mesaj başlığı 15 bayttan 2 bayta düşürülebilir. İleti başlığında gereksiz yere büyük alanlar yoktur. Ağ üzerinde daha az veri yükü sağlar. Böylece Kısım 2.13'deki sorun çözülmüş olur.

4.4.3 Mesaj kimliği

Kullanıcı tanımlı tüm mesajların kimlikleri, sunucu tarafında mesaj yapılarının MCP'ye tanımlanması aşamasında MCP tarafından otomatik olarak belirlenir. Belirlenen mesaj kimlikleri,Kısım 4.5.1'de açıklanan el sıkışma mesajı ile istemcilere iletilir. Mesaj kimliği, MCP'nin kendi başlığında taşınır.

4.4.4 Korelasyon kimliği

Eş zamansız iletişimlerde, Kısım 4.6.1'de açıklanan istek mesajları ile yanıt mesajları arasındaki ilişkiyi temsil eden bir kimlik numarası gereklidir. Bu nedenle, korelasyon kimliği alanı mesaj başlığına eklenir. Birden fazla istek mesajına karşılık gelen birden fazla yanıt mesajı, Korelasyon Kimliği ile ilişkilendirilebilir. Böylece Kısım 2.3'te anlatılan problem çözülmüş olur. Korelasyon Kimliği alanı sadece istek-yanıt mesajları için kullanılır.

4.4.5 Rezerve

Ayrılmış alan, gelecekte MCP'nin yeni sürümleri tasarlanırken MCP başlığında alanlara ihtiyaç duyulması durumunda kullanılabilir.

4.5 MCP Standart Mesajları

MCP başlığındaki MCP bayrağı, standart mesajlar için 0 olarak ayarlanmıştır. Protokole ait yerleşik mesajlara MCP standart mesajları denir. 5 tür standart mesaj vardır:

- El Sıkışma Mesajı
- Kalp Atışı Mesajı
- Rol Başvuru Mesajı

- Abone Olma Mesajı
- Abonelikten Çıkma Mesajı

MCP standart mesajları, TCP protokolü kullanılarak ayrılmış bir "well-known" port numarası [25] üzerinden iletilir. MCP için ayrılmış port numarası 678'dir. Veriler ikili olarak iletilir ve veriler daima little-endian olarak iletilir. Bunun sebebi henüz uç noktaların endianness tipi bilinmiyor olabilir.

4.5.1 El sıkışma mesajı

Bu mesaj bir istek-yanıt mesajıdır. El sıkışma istek mesajının kimlik numarası 0, el sıkışma yanıt mesajının kimlik numarası 1'dir. El sıkışma mesajı bir istek-yanıt mesajı olmasına rağmen korelasyon kimliği alanı kullanılmaz. Çünkü bir iletişimde aynı anda sadece bir el sıkışma istek mesajı olabilir. El sıkışma isteğinin yapısı Şekil 4.8'deki gibidir. El sıkışma istek mesajı, bağlantı anahtarı, rol anahtarı ve kullanıcı tanımlı mesajların yapılarını içerir. Bu alanların kullanımı ile ilgili detaylar Kısım 4.7.1'de anlatılır.

MCP Bayrağı (1 bit)	Alanların Uzunlukları			Mesaj ID (1 bayt)	Korelasyon ID ve Rezerve (0 bayt)	Bağlantı Anahtarı (8 bayt)	Rol Anahtarı (8 bayt)	Kullanıcı Tanımlı Mesaj Yapıları (dinamik boyutlu)
	Mesaj ID Alanının Uzunluğu (2 bit)	Korelasyon Alanının Uzunluğu (2 bit)	Rezerve Alanın Uzunluğu (3 bit)					
0	1	0	0	0	-			

Şekil 4.8 : MCP El Sıkışma İstek Mesajı.

El sıkışma yanıt mesajının başlığı Şekil 4.9'da gösterildiği gibidir. El sıkışma yanıt mesajının gövdesi de Şekil 4.10'da gösterildiği gibidir. El sıkışma yanıt mesajı, endianness bilgisi, bağlantı kabul sonucu, bağlantı hata mesajı, bağlantı anahtarı, mesaj kimlik numaraları ve alt katman protokollerine ait port numaralarını içerir. Bu alanların kullanımı ile ilgili detaylar Kısım 4.7.1'de anlatılır.

Bu mesajın amacı, bağlantı kurulma aşamasında mesajların iletilmesi için gerekli olacak verilerin uç noktalara aktarılması ve ön koşulların sağlanmasıdır. Bağlantı kurma aşamasında, istemci tarafından sunucuya bir el sıkışma istek mesajı gönderilir

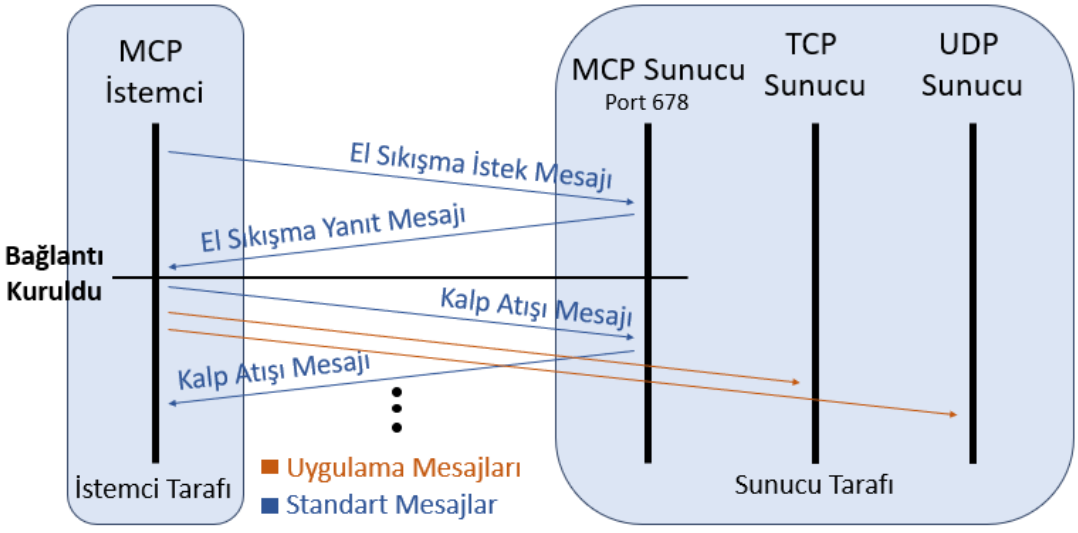
MCP Bayrağı (1 bit)	Alanların Uzunlukları			Mesaj ID (1 bayt)	Korelasyon ID (0 bayt)	Rezerve (0 bayt)	Mesaj Gövdesi
	Mesaj ID Alanının Uzunluğu (2 bit)	Korelasyon Alanının Uzunluğu (2 bit)	Rezerve Alanın Uzunluğu (3 bit)				
0	1	0	0	1	-	-	

Şekil 4.9 : MCP El Sıkışma Yanıt Mesajı Başlığı.

Endianness Bilgisi (1 bayt)	Sonuç (1 bayt)	Hata Mesajı Uzunluğu (2 bayt)	Hata Mesajı (dinamik)	Bağlantı Anahtarı (8 bayt)	Mesaj ID ve Mesaj İsimleri İkilileri Alanının Uzunluğu (3 bayt)	Port Numaraları Sayısı (1 bayt)	Mesaj ID ve Mesaj İsimleri İkilileri (JSON Formatında) (dinamik)	Port Numaraları (Özellik-Port İkilisi) (Özellik 2 bayt) (Port 2 bayt) (dinamik)

Şekil 4.10 : MCP El Sıkışma Yanıt Mesajı Gövdesi.

ve sunucu yanıt olarak el sıkışma yanıt mesajını istemciye gönderir. Sunucu, el sıkışma istek mesajını aldıktan sonra, bağlantı kabul edilse de edilmese de el sıkışma yanıt mesajını gönderir. Bağlantı kabul edildiyse sonuç alanının değeri 0'dır ve Şekil 4.11'de gösterildiği gibi bağlantı kurulur. Eğer bağlantı kabul edilmediyse sonuç alanının değeri 1'dir ve hata mesajı bağlantının neden kurulamadığını açıklar.



Şekil 4.11 : MCP Bağlantı Kurulumu.

4.5.2 Kalp atışı mesajı

Bu mesaj bir istek-yanıt mesajıdır. Kalp atışı istek mesajının kimlik numarası ve kalp atışı yanıt mesajının kimlik numarası 2'dir. Kalp atışı mesajı bir istek-yanıt mesajı olmasına rağmen korelasyon ID alanı kullanılmaz. Çünkü bir iletişimde aynı anda sadece bir kalp atışı istek mesajı olabilir. İletinin gövdesi boştur. Kalp atışı istek mesajı ve kalp atışı yanıt mesajlarının yapıları aynıdır ve yapıları Şekil 4.12'deki gibidir.

MCP Bayrağı (1 bit)	Alanların Uzunlukları			Mesaj ID (1 bayt)	Korelasyon ID (0 bayt)	Rezerve (0 bayt)
	Mesaj ID Alanının Uzunluğu (2 bit)	Korelasyon Alanının Uzunluğu (2 bit)	Rezerve Alanın Uzunluğu (3 bit)			
0	1	0	0	2	-	-

Şekil 4.12 : MCP Kalp Atışı Mesajı.

Bir bağlantı kurulduktan sonra, bağlantının canlı tutulduğunu tespit etmek için istemci tarafından sunucuya periyodik olarak kalp atışı istek mesajı gönderilir. Sunucu ayrıca istemciye Şekil 4.11'de gösterildiği gibi bir kalp atışı yanıt mesajıyla yanıt verir. Kalp atışı istek mesajı, her kalp atışı yanıt mesajı alındıktan bir saniye sonra gönderilir. İstemci, kalp atışı yanıt mesajını 5 saniye boyunca almazsa, istemci bağlantının kesildiğini algılar. Benzer şekilde, sunucu 6 saniye boyunca kalp atışı istek mesajını almazsa, sunucu bağlantının kesildiğini algılar. Kalp atışı mesajının zaman aşımı süresi ve frekansı kullanıcı tarafından değiştirilebilir. Kalp atışı mesajı ile Kısım 2.12'deki sorun giderildi.

4.5.3 Rol başvuru mesajı

Bu mesaj bir istek-yanıt mesajıdır. Rol başvuru istek mesajının kimlik numarası 3, rol başvuru yanıt mesajının kimlik numarası 4'tür. Rol başvuru mesajı bir istek-yanıt mesajı olmasına rağmen korelasyon kimliği alanı kullanılmaz. Çünkü bir iletişimde, bir istemciye ait aynı anda sadece bir rol başvuru istek mesajı olabilir. Rol Başvuru istek mesajının yapısı Şekil 4.13'teki gibidir. Rol başvuru istek mesajı, istemcinin IP numarasını ve başvurulan rolün kimlik numarasını içerir.

MCP Bayrağı (1 bit)	Alanların Uzunlukları			Mesaj ID (1 bayt)	Korelasyon ID ve Rezerve (0 bayt)	İstemci IP Adresi (16 bayt)	Rol ID (2 bayt)
	Mesaj ID Alanının Uzunluğu (2 bit)	Korelasyon Alanının Uzunluğu (2 bit)	Rezerve Alanın Uzunluğu (3 bit)				
0	1	0	0	3	-		

Şekil 4.13 : MCP Rol Başvuru İstek Mesajı.

Rol Başvuru yanıt mesajının yapısı Şekil 4.14'deki gibidir. Rol başvuru yanıt mesajı, sonucu ve başvurulan rolün anahtarını içerir.

MCP Bayrağı (1 bit)	Alanların Uzunlukları			Mesaj ID (1 bayt)	Korelasyon ID ve Rezerve (0 bayt)	Sonuç (1 bayt)	Rol Anahtarı (8 bayt)
	Mesaj ID Alanının Uzunluğu (2 bit)	Korelasyon Alanının Uzunluğu (2 bit)	Rezerve Alanın Uzunluğu (3 bit)				
0	1	0	0	4	-		

Şekil 4.14 : MCP Rol Başvuru Yanıt Mesajı.

Bir MCP bağlantısında, istemcilerin rolleri vardır. İstemci rolleri ve rol atama süreci Kısım 4.8'de açıklanmıştır. Kısım 4.8.2'de anlatılan rol atama sürecine göre, rol başvuru istek mesajı ve rol başvuru yanıt mesajının içeriğinin nasıl doldurulduğu bu paragrafta anlatılmıştır. Rol başvuru mesajı, istemcilere rol ataması yapılabilmesi için oluşturulmuş bir standart mesajdır. Bir istemci, sunucu ile bağlantı kurduktan sonra, hangi role başvurmak istiyorsa, o rolün numarasını rol başvuru istek mesajına yerleştirerek sunucuya gönderir. Sunucu, mesajın içeriğine istemcinin IP adresini ekleyerek, mesajı Kısım 4.8.1'de açıklanan admin rolündeki istemciye iletir. Admin rolündeki istemci, eğer rol atamasına onay verirse, sonuç alanı 0 olan bir rol başvuru yanıt mesajını sunucuya iletir. Sunucu, rolü talep eden istemcinin rol atamasını yapar. Ek olarak sunucu, ilgili rolün anahtarını rol başvuru yanıt mesajına ekler ve rol başvuru yanıt mesajını rolü talep eden istemciye iletir. Eğer rol atamasına onay verilmezse veya zaman aşımı oluşursa, rol başvuru yanıt mesajındaki sonuç alanının değeri 1 olur. Bu durumda sunucu rol ataması yapmaz. Rol başvuru yanıt mesajını istemciye ileterek, başvurunun reddedildiğini istemciye bildirir.

4.5.4 Abone olma mesajı

Bu mesaj bir istek-yanıt mesajıdır. Abone olma istek mesajının kimlik numarası 5, abone olma yanıt mesajının kimlik numarası 6'dır. Eğer aynı anda birden fazla abone olma mesajı istek mesajı gönderilirse, korelasyon kimlik numarası otomatik olarak devreye girer. Abone olma istek mesajının yapısı Şekil 4.15'teki gibidir. Abone olma istek mesajı, abone olmak istenilen mesajın yapısını ve ismini kapsayan bir JSON objesi içerir.

MCP Bayrağı (1 bit)	Alanların Uzunlukları			Mesaj ID (1 bayt)	Korelasyon ID (1 bayt)	Rezerve (0 bayt)	JSON Formatında Mesaj (dinamik)
	Mesaj ID Alanının Uzunluğu (2 bit)	Korelasyon Alanının Uzunluğu (2 bit)	Rezerve Alanın Uzunluğu (3 bit)				
0	1	1	0	5	1	-	

Şekil 4.15 : MCP Abone Olma İstek Mesajı.

Abone olma yanıt mesajının yapısı Şekil 4.16'daki gibidir. Abone olma yanıt mesajı, abone olunan mesaj kimlik numarasını içerir.

MCP Bayrağı (1 bit)	Alanların Uzunlukları			Mesaj ID (1 bayt)	Korelasyon ID (1 bayt)	Rezerve (0 bayt)	Mesaj ID (4 bayt)
	Mesaj ID Alanının Uzunluğu (2 bit)	Korelasyon Alanının Uzunluğu (2 bit)	Rezerve Alanın Uzunluğu (3 bit)				
0	1	1	0	6	1	-	

Şekil 4.16 : MCP Abone Olma Yanıt Mesajı.

Bağlı bir istemci bir mesaja abone olmak istediğinde, sunucuya abone olma isteği mesajını gönderir. Kısım 4.9'da mesaja abone olma işlevi açıklanmıştır. Abone olma istek mesajı ile abone olunan mesajın adı ve yapısı JSON formatında sunucuya iletilir. Sunucu, mesaj yapısının uyumluluğunu kontrol eder ve ardından sunucu, istemciye yanıt olarak bir abone olma yanıt iletisi gönderir. Abone olma yanıt mesajı ile abone olunan mesajın ID'si istemciye iletilir. Eğer mesaj yapıları uyumlu değilse, mesaj kimliği -1 değeri ile istemciye iletilir.

4.5.5 Abonelikten çıkma mesajı

Bu mesaj bir istek-yanıt mesajıdır. Abonelikten çıkma istek mesajının kimlik numarası 7, abonelikten çıkma yanıt mesajının kimlik numarası 8'dir. Eğer aynı anda birden fazla abonelikten çıkma mesajı istek mesajı gönderilirse, korelasyon kimlik numarası otomatik olarak devreye girer. Abonelikten çıkma istek mesajının yapısı Şekil 4.17'deki gibidir. Abonelikten çıkma istek mesajı, Abonelikten çıkmak istenilen mesajın kimlik numarasını içerir.

MCP Bayrağı (1 bit)	Alanların Uzunlukları			Mesaj ID (1 bayt)	Korelasyon ID (1 bayt)	Rezerve (0 bayt)	Mesaj ID (4 bayt)
	Mesaj ID Alanının Uzunluğu (2 bit)	Korelasyon Alanının Uzunluğu (2 bit)	Rezerve Alanın Uzunluğu (3 bit)				
0	1	1	0	7	1	-	

Şekil 4.17 : MCP Abonelikten Çıkma İstek Mesajı.

Abonelikten çıkma yanıt mesajının yapısı Şekil 4.18'deki gibi gövdesi boştur.

MCP Bayrağı (1 bit)	Alanların Uzunlukları			Mesaj ID (1 bayt)	Korelasyon ID (1 bayt)	Rezerve (0 bayt)
	Mesaj ID Alanının Uzunluğu (2 bit)	Korelasyon Alanının Uzunluğu (2 bit)	Rezerve Alanın Uzunluğu (3 bit)			
0	1	1	0	8	1	-

Şekil 4.18 : MCP Abonelikten Çıkma Yanıt Mesajı.

Bağlı bir istemci bir mesajın aboneliğini iptal etmek istediğinde, sunucuya bir mesaj kimliği içeren abonelikten çıkma istek mesajını gönderir. Sunucu, istemciye ilgili mesajın aboneliğinden çıkıldığını bildirmek için abonelikten çıkma yanıt mesajını istemciye gönderir.

4.6 MCP Uygulama Mesajları

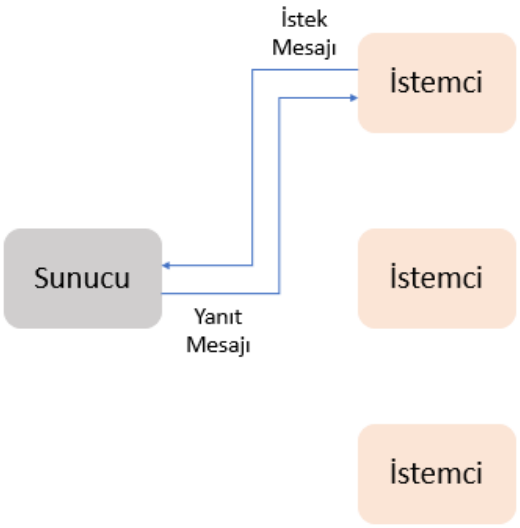
MCP uygulama mesajları, uygulama kodunda tanımlanan mesajlardır. MCP'de bazı mesajlar noktadan noktaya bir modelle iletilirken, bazı mesajlar noktadan çok noktaya

bir modelle iletilir. Mesaj türleri, mesajların nasıl iletileceğini belirleyen modeli belirler. Dört tür MCP uygulama mesajı vardır:

- İstek-Yanıt Mesajı
- Olay Mesajı
- Başlangıç Mesajı
- Rapor Mesajı

4.6.1 İstek-yanıt mesajı

İstek-yanıt mesajları, Şekil 4.19'da gösterildiği gibi noktadan noktaya bir modelle iletilir. Bir istemci yalnızca rolü için uygun olan istek mesajlarını gönderebilir. İstemci rolleri Kısım 4.8.1'de açıklanmıştır. Sunucu, yanıt mesajını yalnızca ilgili istek mesajını aldığında gönderebilir.

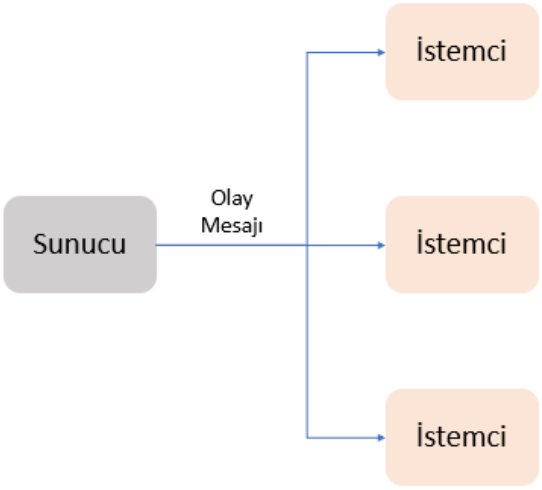


Şekil 4.19 : İstek-Yanıt Mesajı.

4.6.2 Olay mesajı

Olay mesajları, Şekil 4.20'de gösterildiği gibi noktadan çok noktaya bir modelle iletilir. Olay mesajlarının iletimi, sunucu tarafı uygulama kodu tarafından tetiklenir. Olay mesajları, rollerinin olay mesajına erişimi olmayan ve olay mesajına abone

olmayan istemciler dışındaki tüm bağlı abone istemcilerine gönderilir. İstemci rolleri Kısım 4.8.1'de açıklanmıştır. Böylece veriler ilgili tüm istemciler ile paylaşılır.



Şekil 4.20 : Olay Mesajı.

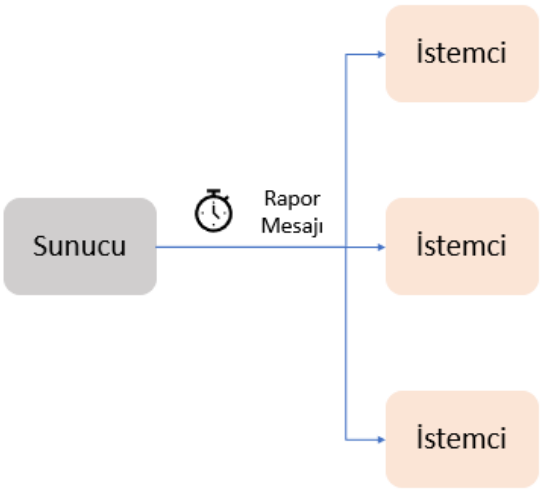
4.6.3 Başlangıç mesajı

Başlangıç mesajları, noktadan çok noktaya bir modelle iletilir. Başlangıç mesajı aslında uygulama kodu ile tetiklenmek yerine bağlantı kurulduğunda tetiklenen bir olay mesajıdır. Başlangıç mesajları, rollerinin başlangıç mesajına erişimi olmayan ve başlangıç mesajına abone olmayan istemciler dışında tüm bağlı abone istemcilere gönderilir.

4.6.4 Rapor mesajı

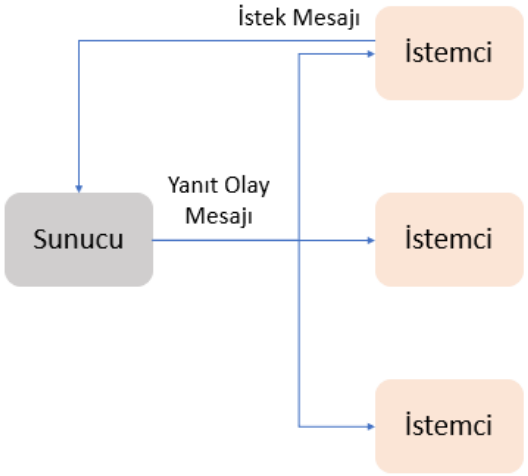
Rapor mesajları, Şekil 4.21'de gösterildiği gibi noktadan çok noktaya bir modelle iletilir. Rapor mesajı aslında uygulama kodu tarafından tetiklenmek yerine zamana göre tetiklenen bir olay mesajıdır. Rapor mesajları, rollerinin rapor mesajına erişimi olmayan ve rapor mesajına abone olmayan istemciler dışındaki tüm bağlı abone istemcilere gönderilir.

Bir mesajın birden fazla mesaj türü olabilir. Başka bir deyişle, bir mesaj aynı anda bir yanıt mesajı, bir olay mesajı, bir başlangıç mesajı ve bir rapor mesajı olabilir. Örneğin, bir yanıt olayı mesajı, Şekil 4.22'deki gibidir. Yanıt olay mesajı sayesinde, non-stateless iletişimlerde, bir istemci sunucunun durumunu



Şekil 4.21 : Rapor Mesajı.

değiştirdiğinde, sunucunun durum verisi diğer istemcilerle senkronlanır. Böylece Kısım 2.9'daki sorun giderilmiş olur.



Şekil 4.22 : Yanıt Olay Mesajı.

4.7 MCP Bağlantısı

MCP bağlantıları, hem sunucu-istemci modelini hem de yayınla-abone ol modelini destekleyen hibrit bağlantı modelidir. Yani tek bir bağlantıda her iki model de kullanılabilir. MCP, klasik yayınla-abone ol modellerinden farklıdır. MCP, bu iki modeli bağlantı tabanlı bir yaklaşım yerine mesaj tabanlı bir yaklaşım ile kullanmaya çalışır. Sunucu-istemci modeli için Kısım 4.6.1'de açıklanan istek-yanıt mesajları kullanılır. Yayınla-abone ol modeli için Kısım 4.6.2'de açıklanan olay mesajları

kullanılır. Bu nedenle MCP, birçok protokolden farklı olarak bir mesaj aracısına ihtiyaç duymaz.

4.7.1 Bağlantı kurulma aşaması

Bir MCP bağlantısının kurulabilmesi için Kısım 4.5.1'de anlatılan el sıkışma mesajlarının uç noktalar arasında iletilmesi gereklidir. Sunucu, istemciden el sıkışma istek mesajını aldıktan sonra, el sıkışma istek mesajındaki kullanıcı tanımlı mesaj yapıları ile kendi tarafında tanımlı mesaj yapılarını karşılaştırır. Sunucu, sonucu istemciye el sıkışma yanıt mesajıyla gönderir. Mesaj yapıları uyumlu ise Şekil 4.11'de gösterildiği gibi bağlantı kurulur. Böylece Kısım 2.4'teki sorun çözülmüş olur. Mesaj yapıları uyumlu değilse, istek yanıt mesajındaki hata mesajı alanına uyumsuzluk hakkında detay bilgisi eklenerek istek yanıt mesajı gönderilir.

İstemcilerin, mesajlaşma arayüzündeki tüm mesajların yapılarını tanımlaması ve bir el sıkışma istek mesajı ile sunucuya göndermesi gerekmez. İstemciler sadece kullanmak istedikleri mesajların yapılarını tanımlarlar ve bunları bir el sıkışma istek mesajı ile sunucuya gönderirler. Bir anlamda, müşteriler kullanmak istedikleri mesajlara bir el sıkışma mesajı ile abone olurlar. Bu yöntem aynı zamanda el sıkışma istek mesajının boyutunun gereksiz yere büyük olmasını da engeller.

Mesaj kimlikleri, Kısım 4.11.5'te anlatıldığı gibi sunucu tarafında protokol seviyesinde otomatik oluşturulur. Bu yüzden, istemci mesaj kimliklerini sunucu iletmediği sürece bilemez. El sıkışma istek mesajı ile iletilen mesajların kimlik numaraları, sunucu tarafından el sıkışma yanıt mesajı ile istemciye bildirilir. El sıkışma yanıt mesajında mesaj kimlikleri, JSON formatında mesaj isimleri ile ilişkilendirilir. Daha sonra bu el sıkışma yanıt mesajı istemciye iletilir.

Alt katman protokollerinde de otomatik bağlantılar kurulabilmesi için, istemcinin port numaralarını bilmesi gerekir. Bu yüzden port numaraları, sunucu el sıkışma yanıt mesajı ile istemciye bildirir. Kısım 4.1'de veri iletim özelliği, bir alt katman protokolü ile ilişkilendirilerek numaralandırılmıştı. Aynı veri iletim özelliği numarası kullanılarak, port numaraları ve alt katman protokolleri ilişkilendirilmiş olunur.

Bağlantı anahtarı, aynı istemci tekrar bağlantı kurduğunda, gereksiz yere tekrar mesaj yapılarının iletilmesini, tekrar mesaj yapılarının kıyaslanmasını, tekrar mesaj kimlik numaralarının iletilmesini, tekrar port numaralarının iletilmesini engellemek için kullanılır. Mesaj yapılarının iletilmesi ve kıyaslanması maliyetli olduğundan, bu işlemin bir kez yapılması hedeflenmiştir. Bağlantı kurulduktan sonra, DES yöntemi [26] ile 64 bitlik bir bağlantı anahtarı üretilir. Bu anahtar el sıkışma yanıt mesajı ile istemciye iletilir. Bu aşamadan sonra yeniden bağlantı kurulmak istendiğinde, istemci bağlantı anahtarını el sıkışma istek mesajında göndererek mesaj uyumluluğu kontrolü aşamasından kurtulur. Böylece ilk bağlantı kurma denemesine göre çok daha hızlı bir şekilde bağlantı kurulur. Sunucu, eğer uyumsuz bir bağlantı anahtarı alırsa, el sıkışma yanıt mesajının hata mesajı alanında bağlantı anahtarının hatalı olduğunu istemciye bildirir.

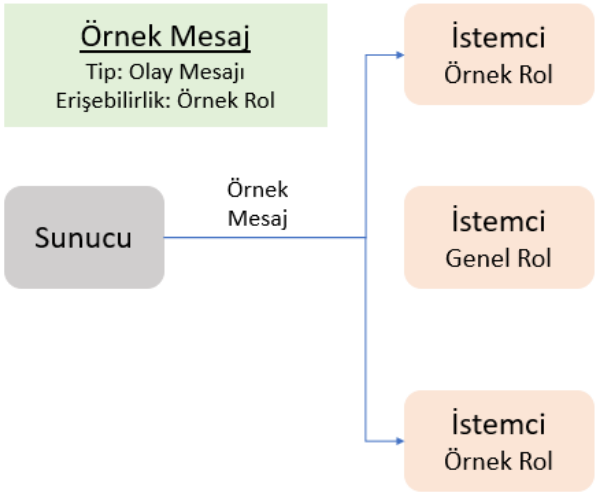
Rol anahtarı, aynı istemci tekrar bağlantı kurduğunda, aynı rolü tekrar alabilmesi için kullanılır. Böylece istemciler tekrardan Kısım 4.5.3'te açıklanan rol başvuru mesajını kullanmak zorunda kalmazlar. Roller ve rol atama süreci, Kısım 4.8'de detaylı olarak açıklanmıştır. Eğer sunucu, uyumsuz bir rol anahtarı alırsa, el sıkışma yanıt mesajının hata mesajı alanında rol anahtarının hatalı olduğunu istemciye bildirir.

El sıkışma yanıt mesajındaki endianness bilgisi alanındaki 0 değeri, sunucu cihazının little-endian mimaride bir işlemci kullandığı ve veri iletiminin de little-endian olduğu anlamını taşır. Endianness bilgisi alanındaki 1 değeri, sunucu cihazının big-endian mimaride bir işlemci kullandığı ve veri iletiminin de big-endian olduğu anlamını taşır. İstemci, sunucudan el sıkışma yanıt mesajını aldıktan sonra sunucunun endianness tipini öğrenir. İstemci ve sunucunun endianness türleri farklı olsa bile, Kısım 4.11.6 ve Kısım 4.11.7'de bu endianness tipi kullanılarak, veriler serileştirilir ve ayrıştırılır. Böylece Kısım 2.10'daki problem çözülmüş olur.

4.8 MCP Mesaj Erişim Yetkilendirmesi

Mesaj erişim yetkilendirmesi için, rol tabanlı erişim kontrol [15,16] yöntemi kullanılmıştır. Bu sayede, bir MCP bağlantısında, istemcilerin kimlikleri olmasa da rolleri vardır. İstemcilerin rolleri, mesajlaşma arayüzündeki mesajların erişim

yetkilerini belirler. Sunucu, her mesaj için hangi istemci rollerinin erişebileceğini belirler. Bir mesajın erişilebilirliği birden fazla istemci rolüne verilebilir. Böylece sunucu, Şekil 4.23 örneğinde gösterildiği gibi, istemcilerin kimliklerini bilmeden, istemci rolüne göre mesajları filtreleyebilir. Kısım 2.7'de anlatılan kimliklendirme olmadan yetkilendirme problemi çözülür.



Şekil 4.23 : Rol Tabanlı Mesaj Erişimi Yetkilendirmesi ile Olay Mesajı İletim Örneği.

4.8.1 MCP istemci rolleri

MCP istemci rollerinin sayısı sabit değildir. 3 adet standart rol olduğu gibi, kullanıcı tanımlı roller ile rol sayısı artar. MCP rolleri şunlardır:

- Genel Rol
- Admin Rol
- Monitör Rol
- *Kullanıcı Tanımlı Roller*

Genel rol dışında her istemci rolünün bir anahtarı vardır. Rol anahtarları, derleme zamanında, sunucu tarafında, DES yöntemi [26] kullanılarak, 64 bit boyutunda protokol seviyesinde üretilir. Bu rol anahtarlarının amacı, bir role atanmış istemcinin, tekrar bağlantı kurmak istediğinde, aynı rolle bağlanabilmesidir. Bağlantı kurma aşamasında istemci, rol anahtarını Kısım 4.5.1'de açıklanan el sıkışma istek mesajıyla

birlikte sunucuya iletir. Böylece bağlantı yeniden kurulmuş olsa bile, istemci daha önceden atanmış olduğu role sahip olur.

4.8.1.1 Genel rol

Her istemcinin varsayılan rolü genel roldür. Her mesajın varsayılan erişilebilirliği genel roldür. Genel rolün bir anahtarı yoktur.

4.8.1.2 Admin rol

MCP sunucusunu ayağa kaldırmak için, bir admin rolüne sahip cihazın MAC adresinin MCP sunucusunda konfigüre edilmesi zorunluluktur. Böylece bir bağlantıdaki admin rolündeki istemci bilinebilir. Kısım 4.8.2'de anlatılan tüm rol atama onayları, admin rolündeki istemci tarafından yapılır. Eğer iletişimde bir admin rolünde istemci henüz bağlanmadıysa, tüm rol başvuru istekleri reddedilir.

4.8.1.3 Monitör rol

Noktadan noktaya iletişimde mesajları izlemek isteyen istemcilerin rolü monitör rolüdür. Monitör rollerinin rol anahtarı her bağlantı için farklıdır. Sunucu, her bağlantı kurulduğunda DES yöntemi [26] ile 64 bitlik bir rol anahtarı oluşturur. Monitör rolündeki bir istemci, izlemek istediği bağlantıya rol başvuru mesajının içeriğini izleme rolü olarak doldurup gönderir. Eğer iki uç noktadan biri, bu rol başvurusunu kabul ederse, mesajları izlemek isteyen istemci, monitör rolüne sahip olarak hem sunucu hem de istemci ile bağlantı kurar.

Monitör istemcileri tarafından kurulan bağlantılar tek yönlüdür. Şekil 2.1'de gösterildiği gibi, monitör istemcileri veri gönderemez, yalnızca veri alabilirler. Monitör rolü mesajların erişilebilirliğinden bağımsızdır. Noktadan noktaya iletişimdeki tüm mesajlar monitör istemcisine iletilir. Sonuç olarak Kısım 2.6'da anlatılan sorun çözülür.

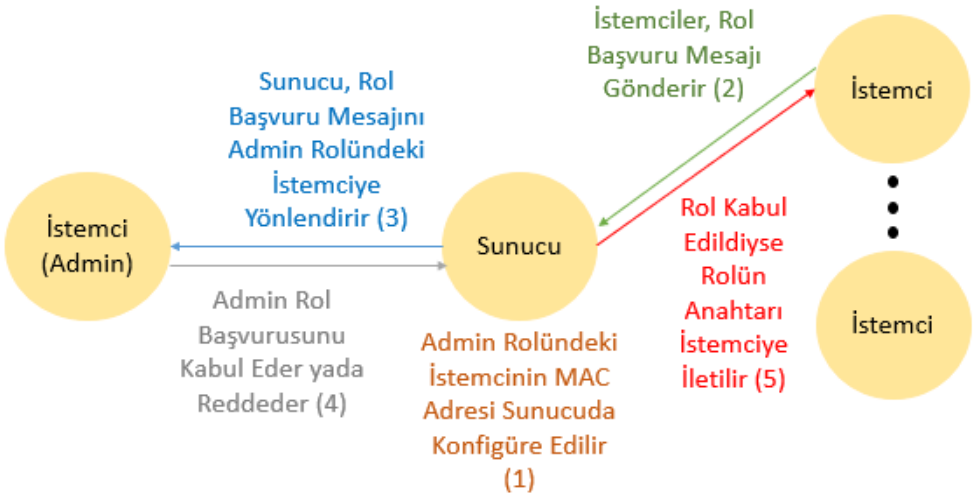
4.8.1.4 Kullanıcı tanımlı roller

Bir mesajlaşma arayüzünde sınırsız sayıda istemci rolü oluşturulabilir. Kullanıcı tanımlı roller protokol tarafından oluşturulmuş standart roller değildir. Bu roller

sayesinde, kimliklendirme olmadan mesaj erişim yetkilendirmeleri uygulamaya göre özelleştirilebilir.

4.8.2 Rol atama süreci

Rol atamaların yapılabilmesi için, bağlantıda admin rolünde bir istemcinin olması zorunluluktur. Sunucu, istemcilerin kimliklerini bilmediği için, istemcilerin kimliklerini bilen admin rolündeki istemciden rol atamaları için yardım alır. İlk olarak Şekil 4.24'de gösterildiği gibi, istemciler rol başvurularını sunucuya iletirler. Sunucu, bu istekleri admin rolündeki istemciye yönlendirir. Daha sonra admin, başvuruyu kabul edip etmediğini sunucuya bildirir. Eğer başvuru kabul edilmişse, sunucu rol anahtarını istemciye iletir. Rol anahtarı sayesinde, istemcinin sonraki bağlantı kurma isteklerinde, tekrar rol başvurusu yapmasına gerek kalmaz. Eğer rol başvurusu kabul edilmediyse, sunucu rol başvurusunun reddedildiğini istemciye bildirir. Tüm istemciler, bu yöntem sayesinde çalışma zamanında rol alabilirler. Sunucu istemcilerin kimliklerini bilmese de, rol atamaları yapılabilir. Böylece, Kısım 2.8'de anlatılan istemcilerin kimliklerini bilmeden, istemcilere rol atama problemi çözülür.



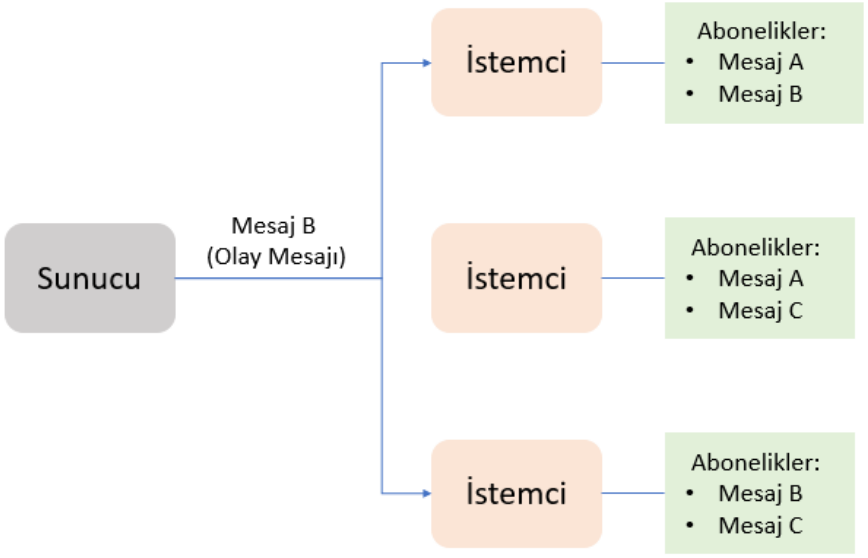
Şekil 4.24 : MCP'de Rol Atama Süreci.

4.9 Bir Mesaja Abone Olma ve Bir Mesajın Aboneliğinden Çıkma

İstemciler, el sıkışma istek mesajını kullanarak ya da abone olma istek mesajını kullanarak mesajlara abone olabilirler. Böylece sunucu, bir istemcinin hangi mesajları

kullanmak istediğini bilir. Sunucu mesaj göndermek istediğinde mesaj abonelere yayınlanır. Sunucu, bu sayede Şekil 4.25 örneğinde gösterildiği gibi mesajları abone istemcilerine göre filtreleyebilir. Bu şekilde, istemciler alınan mesajları kendi taraflarında filtrelemek zorunda kalmazlar. İstenmeyen mesajlar, ağı gereksiz yere işgal etmezler. Böylece Kısım 2.5'deki problem çözülmüş olur.

El sıkışma istek mesajında, sadece abone olunan mesajların yapıları gönderildiği için, sadece abone olunan mesajların yapıları kıyaslanır ve sadece abone olunan mesajların kimlikleri öğrenilir. Daha sonradan başka bir mesaja abone olunmak istendiğinde, abone olma istek mesajında mesaj yapısı JSON formatında iletilir. JSON formatında mesaj yapısının iletilmesinin sebebi, abone olunmak istenilen mesajın yapısının sunucudaki mesaj yapısıyla uyumluluğu henüz kontrol edilmemiş olmasıdır. Çünkü bu mesaj el sıkışma istek mesajına dahil değildi. El sıkışma istek mesajında olmadığından, abone olunmak istenilen mesajın kimliği de henüz istemci tarafından bilinmiyor. Abone olma istek mesajında mesaj yapıları kıyaslandıktan sonra, abone olma yanıt mesajı ile abone olunmak istenilen mesajın kimliği istemciye bildirilir. Böylece abone olma işlemi tamamlanmış olur. Abonelikten çıkma işlevi ise oldukça basittir. Abone olunan mesajların mesaj kimlikleri kullanılarak abonelikten çıkma istek mesajı ile abonelikten çıkılır.



Şekil 4.25 : Bir Olay Mesajının, Mesaja Abone Olan İstemcilere İletim Örneği.

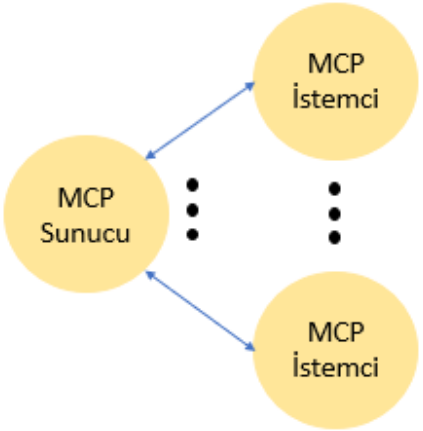
4.10 MCP Uç Nokta Tipleri

MCP, farklı iletişim modelleri için farklı uç nokta tipleri tanımlar. Bu uç nokta tipleri, Kısım 4.11'de açıklanan kullanıcı arayüzündeki fonksiyonların ve olayların ihtiyaca göre farklılaşmasını sağlar. MCP'de tanımlı 4 uç nokta tipi vardır:

- MCP Sunucu
- MCP İstemci
- MCP Düğüm
- MCP Monitör

4.10.1 MCP sunucu

MCP sunucusu, Şekil 4.26'da gösterildiği gibi birden çok MCP istemcisine hizmet vermek için kullanılır. Noktadan çok noktaya iletişim modeli için kullanılır. MCP sunucusu olay mesajı ve yanıt mesajı gönderebilir ancak talep mesajı gönderemez. Ayrıca, MCP sunucusu yanıt mesajını yalnızca ilgili istek mesajını aldığında gönderebilir.



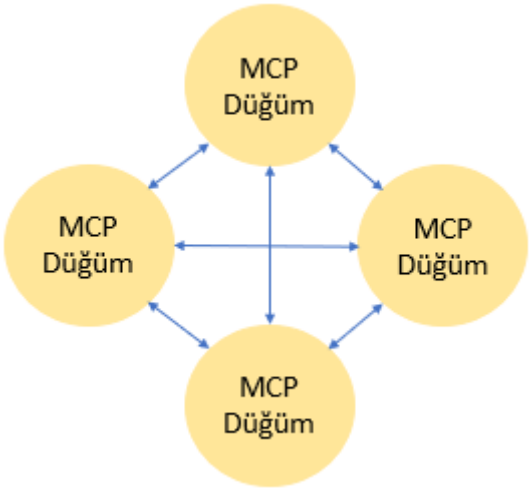
Şekil 4.26 : MCP İstemci-Sunucu Bağlantısı.

4.10.2 MCP istemci

MCP istemcisi, Şekil 4.26'da gösterildiği gibi MCP sunucusuna bağlanmak için kullanılır. MCP istemcisi yalnızca bir MCP sunucusuna bağlanabilir. Noktadan noktaya iletişim modeli için kullanılır. MCP istemcisi istek mesajı gönderebilir ancak yanıt mesajı ve olay mesajı gönderemez.

4.10.3 MCP düğüm

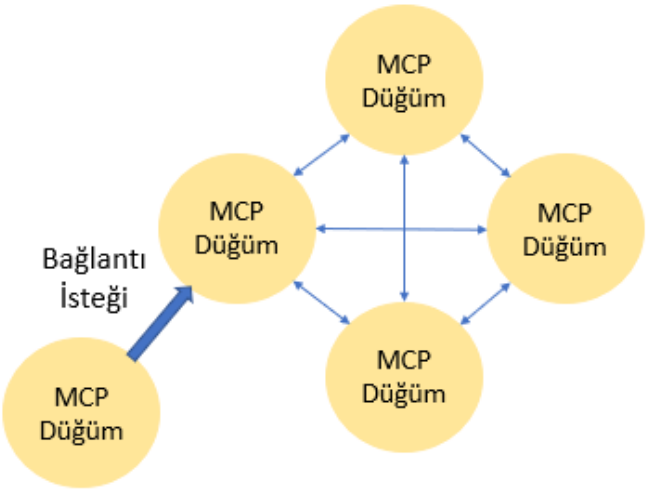
MCP düğümü, Şekil 4.27'de gösterildiği gibi birden çok uç noktanın birbirine mesaj göndermesi için kullanılır. Çok noktadan çok noktaya iletişim modelini destekler. Sunucu veya istemci rolü yoktur. Tüm MCP düğümleri eşdeğerdir. Bu, tüm MCP düğümlerinin istek mesajları, yanıt mesajları ve olay mesajları gönderip alabileceği anlamına gelir.



Şekil 4.27 : MCP Düğüm Bağlantılarının Oluşturduğu Tamamen Bağlı Ağ Örneği.

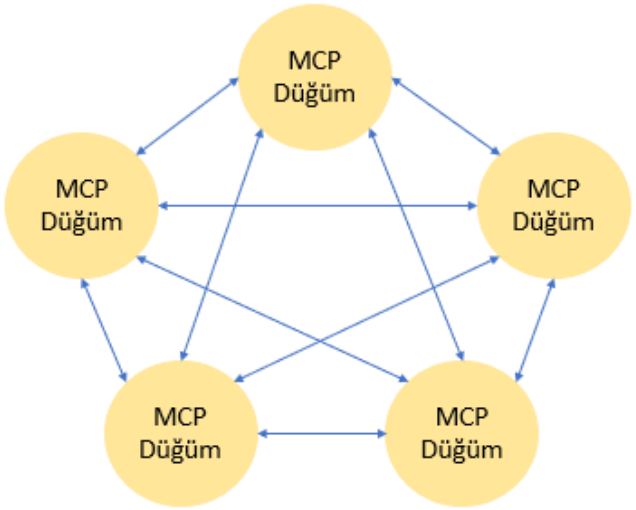
MCP düğümleri birbiriyle tamamen bağlantılı bir ağ oluşturur. Bir MCP Düğümü, Şekil 4.28'de gösterildiği gibi bir MCP düğüm ağındaki bir MCP düğümüne bağlanmak isterse, bağlanmak isteyen MCP düğümü otomatik bağlantılarla ağa dahil edilir.

Şekil 4.29'da gösterildiği gibi, MCP protokolü, bağlanılmak istenen düğümün haricindeki ağındaki diğer düğümlere otomatik bağlantı sağlar. MCP düğümleri arasında bağlantı kurulamazsa, bağlanamayan MCP Düğümleri arasındaki mesajlaşma



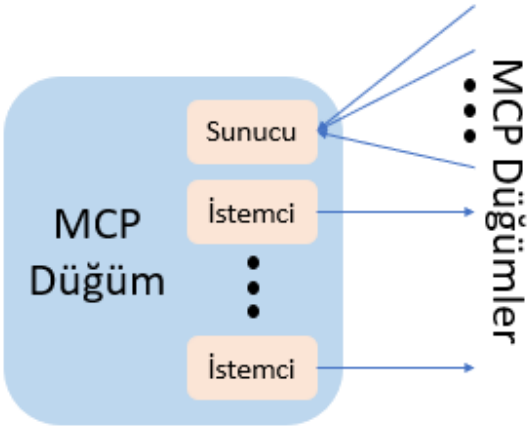
Şekil 4.28 : Bir MCP Düğümünün, Bir MCP Düğüm Ağına Dahil Olmak için Bağlantı Kurma Denemesi.

diğer MCP Düğümleri üzerinden gerçekleştirilir. Böylece bağlantıların manuel olarak yönetilmesine gerek kalmaz.



Şekil 4.29 : MCP Düğümünün Ağa Dahil Olması ile Ağdaki Diğer Düğümlerle Otomatik Bağlantı Kurulması Sonucu Oluşan Tamamen Bağlı MCP Düğüm Ağı.

Bir MCP düğümü, Şekil 4.30'da gösterildiği gibi bir sunucu ve bir istemci bileşeninden oluşur. Sunucu bileşeni mesajı alır ve bağlantıları kabul eder. İstemci bileşeni mesajı gönderir ve bağlantı isteklerini yollar.



Şekil 4.30 : MCP Düğüm Bileşenleri.

4.10.4 MCP monitör

MCP Monitör, iki uç nokta arasında gönderilen ve alınan mesajları izlemek için kullanılır. Bir iletişime bağlanırken MCP Monitör, MCP el sıkışma mesajı ile mesaj yapılarını metin olarak öğrenir. Böylece, veri iletimi ikili olsa bile, mesajları insan tarafından okunabilir ve metin olarak izleyebilir. MCP Monitör veri gönderemez, yalnızca Şekil 2.1'de gösterildiği gibi veri alabilir.

4.11 Kullanıcı/MCP Arayüzü

Bir MCP implementasyonunun, MCP'ye ait özellikleri eksiksiz sağlayabilmesi için eksiksiz olarak kullanıcı arayüzünü içermesi bir zorunluluktur. Kullanıcı arayüzüne ait 6 adet fonksiyon, 3 adet olay vardır. Her fonksiyon ve her olay, her uç nokta tipi için geçerli değildir.

4.11.1 Başlat fonksiyonu

void Start() : Bu fonksiyon, sunucu ve düğüm uç noktaları tarafından kullanılır. İstemci ve monitör uç noktaları için bu fonksiyon tanımlı değildir. Bu fonksiyon öncelikle, port 678'de bir TCP sunucusu başlatarak standart mesajları kabul etmeye başlar. Ayrıca alt katman protokollerine ait istemcileri kabul etmek için iş parçacıkları başlatılır. Bu iş parçacıkları, Kısım 4.3.1'de açıklanan alt katman protokolüne ait sunucu arayüzündeki

önce "Open", daha sonra "Accept" fonksiyonunu çağırır. Bir istemci kabul edilene kadar iş parçacığı bloklanır. İstemci kabul edildikten sonra, Kısım 4.11.8'de açıklanan bağlantı kuruldu olayı tetiklenir.

4.11.2 Bağlan fonksiyonu

void Connect(IPAddress) : Bu fonksiyon, istemci, düğüm ve monitör uç noktaları tarafından kullanılır. Sunucu uç noktaları için bu fonksiyon tanımlı değildir. Bu fonksiyon öncelikle, port 678'de bir TCP istemcisi başlatarak el sıkışma istek mesajını istenilen cihaza gönderir. El sıkışma yanıt mesajı ile onay alındıktan sonra, alt katman protokollerine ait sunuculara bağlanmak için alt katman protokollerine ait istemciler oluşturur. Kısım 4.3.2'de açıklanan alt katman protokolüne ait istemci arayüzündeki "Connect" fonksiyonu çağırılır. Böylece bağlantı kurulur. Bağlantı kurulduktan sonra, alt katman protokollerinden gelen verileri almak için iş parçacıkları başlatılır. Bu iş parçacıkları, alt katman protokolüne ait istemci arayüzündeki "Receive" fonksiyonunu çağırır. Veri alınana kadar iş parçacığı bloklanır. Veri alındıktan sonra, endianness tipi de kullanılarak, veri ayrıştırılır ve mesaj objesine dönüştürülür. Kısım 4.11.7'de açıklanan mesaj alındı olayı tetiklenir. Eğer "Receive" fonksiyonu bağlantının koptuğunu söylerse Kısım 4.11.7'de açıklanan bağlantı koptu olayı tetiklenir.

4.11.3 Durdur fonksiyonu

void Stop() : Bu fonksiyon, sunucu ve düğüm uç noktaları tarafından kullanılır. İstemci ve monitör uç noktaları için bu fonksiyon tanımlı değildir. Bu fonksiyon öncelikle, port 678'deki TCP sunucusu durdurulur. Daha sonra alt katman protokollerinin sunucuları durdurulur. Sunucuların durdurulması için Kısım 4.3.1'de açıklanan alt katman protokolüne ait sunucu arayüzündeki "Close" fonksiyonu çağırılır.

4.11.4 Bağlantıyı kes fonksiyonu

void Disconnect() : Bu fonksiyon, istemci, düğüm ve monitör uç noktaları tarafından kullanılır. Sunucu uç noktaları için bu fonksiyon tanımlı değildir. Bu fonksiyon öncelikle, port 678'deki TCP istemcisini sonlandırır. Alt katman protokollerine

ait sunuculardan da bağlantıyı koparmak için Kısım 4.3.2'de açıklanan alt katman protokolüne ait istemci arayüzündeki "Disconnect" fonksiyonu çağırılır.

4.11.5 Mesaj tanımla fonksiyonu

void Define<T>() : Bu fonksiyon, sunucu, istemci ve düğüm uç noktaları tarafından kullanılır. Monitör uç noktası için bu fonksiyon tanımlı değildir. Template bir fonksiyondur ve template sınıfı olarak uygulama kodunda tanımlanmış olan bir mesaj sınıfı bekler. Eğer bu fonksiyon sunucu uç noktası veya düğüm uç noktası için çağırılmışsa, mesaj için otomatik olarak bir mesaj kimliği belirlenir. Daha sonra reflection yöntemi ile mesaj sınıfı JSON formatına çevrilir. Mesaj kimliği ve JSON formatındaki mesaj yapısı, el sıkışma mesajında ve abone olma mesajında kullanılmak üzere saklanır. Eğer bu fonksiyon istemci uç noktası için çağırılmışsa, mesaj kimliği belirlenmez ancak aynı şekilde JSON formatında mesaj yapısı oluşturulup saklanır.

4.11.6 Mesaj gönder fonksiyonu

void Send<T>(T message) : Bu fonksiyon, sunucu, istemci ve düğüm uç noktaları tarafından kullanılır. Monitör uç noktası için bu fonksiyon tanımlı değildir. Template bir fonksiyondur ve template sınıfı olarak uygulama kodunda tanımlanmış olan bir mesaj sınıfı bekler. Parametre olarak mesaj sınıfına ait obje alır. Mesaj objesi serileştirilirken, el sıkışma yanıt mesajında elde edilen endianness tipine göre serileştirme yapılır. Serileştirilen veri, alt katman protokolüne ait istemci arayüzü veya bağlı uç nokta arayüzündeki "Send" fonksiyonu ile iletilir.

4.11.7 Mesaj alındı olayı

void OnMessageReceived<T>(T receivedMessage) : Bu olay, tüm uç nokta tipleri tarafından kullanılır. Bir mesaj alındığında tetiklenir. Bir mesaj alındığında yapılacak işler, bu olay için yazılan callback fonksiyonlarında gerçekleştirilebilir.

4.11.8 Bağlantı kuruldu olayı

void OnConnectionAccpeted(RemoteMcpClient client) : Bu olay, sunucu ve düğüm uç noktaları tarafından kullanılır. İstemci ve monitör uç noktası için bu olay

tanımlı değildir. Bir istemci kabul edildiğinde bu olay tetiklenir. Böylece, kullanıcı tüm kabul edilen istemcilerden haberdardır. İstemci kabul edildiğinde, uygulama seviyesinde yapılacak işler bu olay ile tetiklenir. Olayın callback fonksiyonunda, RemoteMcpClient sınıfı sayesinde istemciler, IP adreslerine ve rollerine göre ayrıştırılabilir. Bağlantı kurulduktan sonra, alt katman protokollerinden gelen verileri almak için iş parçacıkları başlatılır. Bu iş parçacıkları, alt katman protokolüne ait bağlı uç nokta arayüzündeki "Receive" fonksiyonunu çağırır. Veri alınana kadar iş parçacığı bloklanır. Veri alındıktan sonra, endianness tipi de kullanılarak, veri ayrıştırılır ve mesaj objesine dönüştürülür. Kısım 4.11.7'de açıklanan mesaj alındı olayı tetiklenir. Eğer "Receive" fonksiyonu bağlantının koptuğunu söylerse Kısım 4.11.7'de açıklanan bağlantı koptu olayı tetiklenir.

4.11.9 Bağlantı koptu olayı

void OnConnectionLost([RemoteMcpClient client]) : Bu olay, tüm uç nokta tipleri tarafından kullanılır. Bağlantı koptuğunda tetiklenir. İstemci ve monitör uç nokta tiplerinde, RemoteMcpClient parametresi bulunmaz. Sunucu ve düğüm uç nokta tipleri, RemoteMcpClient parametresi sayesinde, hangi bağlı uç noktanın bağlantısının koptuğunu bilir. Alt protokollerdeki bağlantı problemi veya 678 portundaki TCP soketinin TCP FIN mesajı alması sonucu bu olay tetiklenir.

5. PERFORMANS ANALİZİ

Bu çalışmada tasarlanan MCP protokolünün kazandırdığı özelliklerin yanında, performans için getirdiği yükler bu başlık altında analiz edildi. Performans ölçümü, bağlantı kurma süresi ve veri transferi gecikmesi olarak iki başlık altında incelendi. Bu analiz sonucu, performans ve özelliklerin kazanılması arasındaki denge hakkında bir fikir oluşturmak amaçlanmıştır. MCP'nin kullanılacağı ortamın koşullarına uygunluğuna karar vermek için yol gösterici olacaktır. Test düzeneği, 1 Gbps veri transfer hızını destekleyen ethernet adaptörüne sahip uç noktalar ve 1 Gbps veri transfer hızını destekleyen 2 adet Cisco switch ile oluşturulmuş orta ölçekli bir yerel ağdır.

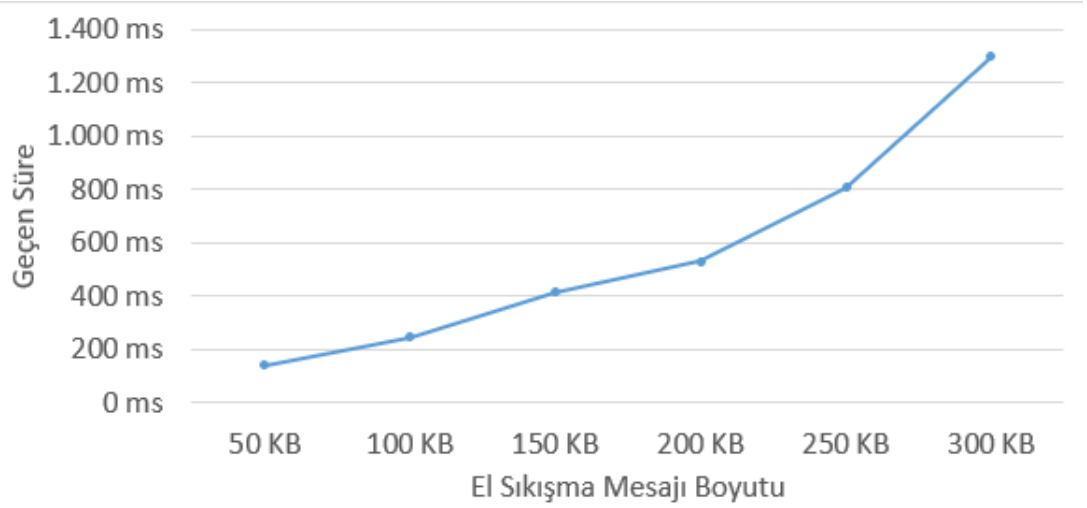
5.1 Bağlantı Kurma Süresi

Bağlantı kurulmadan önce ölçülen süre ile bağlantı kurulduktan sonra ölçülen süre arasındaki fark, bağlantı kurma süresine karşılık gelir. Bağlantı kurma süreleri ölçülürken, bu sürelerin farkı kullanılmıştır. MCP'de bağlantı kurma süresi, el sıkışma mesajının değişebilir boyutlu olmasından dolayı sabit bir süre değildir. Şekil 5.1'de gösterilen grafik ile, tanımlı mesajların boyutu arttıkça el sıkışma mesajının boyutu da artacağı gösterilmek istenmiştir. [1]

MCP'de bağlantı kurma süresi, Şekil 5.1'den anlaşılacağı üzere, artan bir grafik olduğundan, World Wide Web alanına uygun değildir. Artan grafiğin lineer değil de üstel olmasının sebebi, mesaj uyumluklarının karşılaştırılmasının zaman karmaşıklığının O(n^2) olmasıdır. [2]

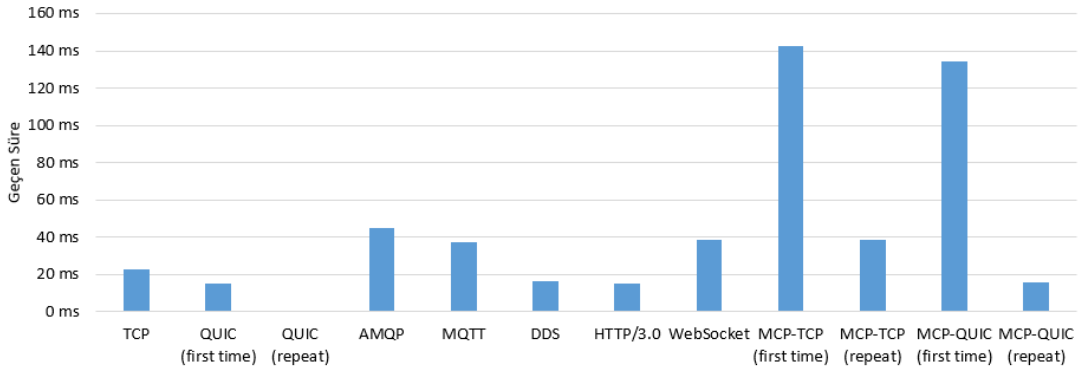
MCP'nin bağlantı kurma süresi, diğer protokollerin bağlantı kurma süresi ile kıyaslanarak, kazanılan özelliklerin bağlantı kurma performansına etkisi ölçülmek

[1] Şekil 5.1'de ölçülen değerler, TCP tabanlı MCP ile alınmıştır.
[2] n, tanımlı mesajların boyutudur.



Şekil 5.1 : MCP Bağlantı Kurma Süresinin MCP El Sıkışma Mesajının Boyutuna Göre Değişimi.

istenmiştir. Bu ölçümlerde, MCP için 46 KB boyutunda el sıkışma mesajı kullanılmıştır. Ölçümler sonucunda Şekil 5.2'deki grafik oluşturulmuştur.

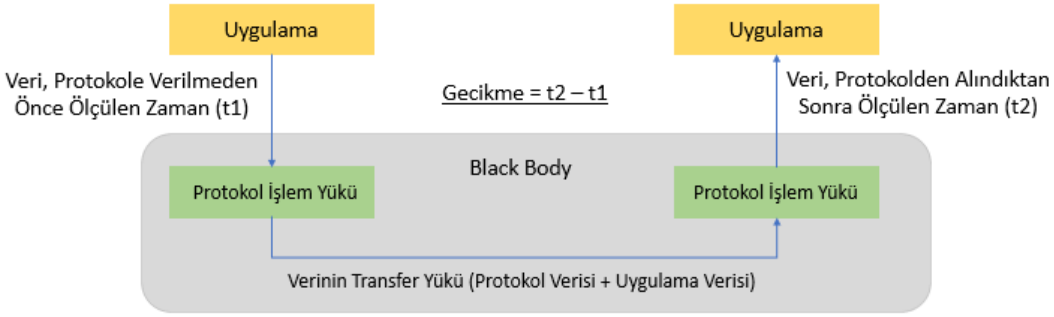


Şekil 5.2 : Protokollerin Bağlantı Kurma Sürelerinin Kıyaslanması.

Şekil 5.2'deki grafikte görüldüğü gibi, MCP'nin ilk bağlantı kurma maliyeti diğer protokollere göre oldukça yüksektir. Ancak Kısım 4.7.1'de bahsedilen bağlantı anahtarı sayesinde, MCP için sonraki bağlantı kurma süreleri, diğer protokoller ile yaklaşık bir bağlantı kurma süresi değerine sahiptir. Bu da birçok yerel ağ haberleşmesi için kabul edilebilir bir performanstır. Mesaj yapıları değişmediği sürece, her MCP bağlantısı kurulacağı zaman, bağlantı anahtarı kullanılarak bağlantı süreleri ideal seviyede tutulabilir.

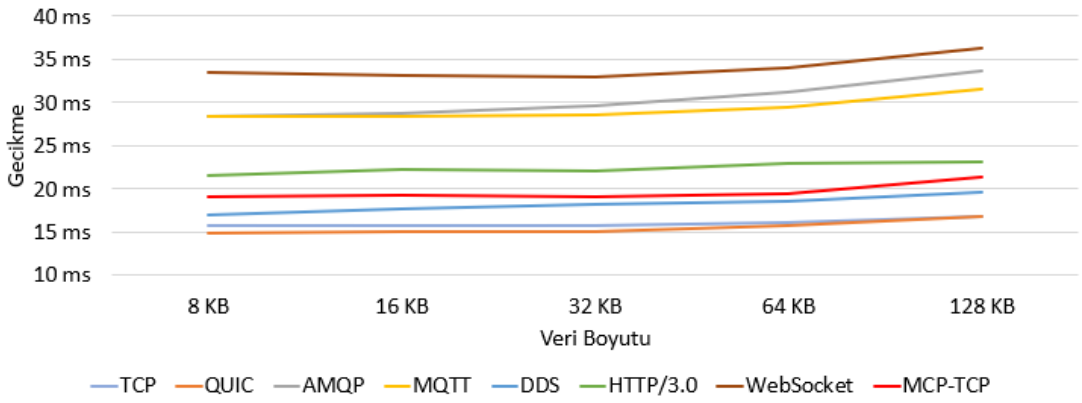
5.2 Veri İletim Gecikmesi

Bir protokol kullanıldığında, o protokolün getirdiği toplam gecikmenin ne kadar olduğu, kıyaslanan her protokol için ölçülmek istenmektedir. Bu gecikmeler kıyaslanarak, protokol ile kazanılan özelliklerin, veri iletimine getirdiği maliyet değerlendirilecektir. Veri iletim gecikmesinin ölçümü için, Benayache [27]'in uyguladığı yöntem ve kullandığı metrikler referans alınmıştır. Bir protokolün veri iletimine getirdiği toplam yükü hesaplamak için, protokolün veriyi göndermek için veriyi işleme yükü, protokol verilerinin ağa getirdiği yük ve protokolün veriyi almak için veriyi işleme yükü birlikte ölçülmüştür. Protokolün getirdiği toplam yük, verinin gecikme süresi ile doğru orantılı olacaktır. Şekil 5.3'de gösterildiği gibi, veri uygulama katmanından protokole verilmeden hemen önceki zaman t1 olarak kaydedildi. Veriyi alan uç noktada, verinin uygulama katmanında alındıktan hemen sonraki zaman t2 olarak kaydedildi. Toplam gecikme, t2 ile t1 arasındaki fark olarak ölçüldü. Bu ölçümler, tüm protokoller ile yapılıp Şekil 5.4'te gösterilen grafik oluşturuldu.



Şekil 5.3 : Veri İletim Gecikmesi Hesaplama Yöntemi.

Şekil 5.2'deki grafikte görüldüğü gibi, MCP, DDS gibi merkeziyetsiz bir publish-subscribe modelini desteklediği için, MQTT ve AMQP'den daha hızlı veri iletimi gerçekleştirmiştir. Veriler iki hope yerine, tek bir hope ile iletilmiştir. MCP'nin veri iletiminin ikili olması ve başlık boyutunun küçük olması, HTTP ve WebSocket protokollerine karşı avantaj sağlamıştır. QUIC, TCP ve DDS, performans olarak MCP'den daha iyi olsalar da, MCP'nin sağladığı tüm özellikleri sağlayamazlar. Eğer MCP'nin sağladığı ama QUIC, TCP ve DDS'in sağlayamadığı bir özellik,



Şekil 5.4 : Protokollerin Veri İletimi Gecikmelerinin Kıyaslanması.

geliştirilecek uygulama için zorunluluksa, performansı biraz daha yavaş olsa da MCP seçilir.

6. SONUÇ

Uygulama katmanında oluşturulan MCP protokolü, mesajlaşma problemlerini protokol kodunda çözerek Bölüm 2'de açıklanan problemlerin çözümünü standart hale getirir. MCP çözümleri standarlaştırarak, uygulama kodunun karmaşıklığını azaltır, mesajlaşma özelliklerinde oluşabilecek hataları minimize eder, çözümleri tüm uç noktalarda ortaklar. Diğer uygulama katmanı protokollerinin, MCP'nin çözdüğü tüm sorunları çözemediği Bölüm 3'te gösterildi. Sadece MCP'nin sağladığı özelliklerin zorunluluk olduğu iletişimlerde, MCP tek seçenektir. MCP'nin öncelikli kullanım alanı yerel ağlardır. MCP yerel ağ iletişimlerinde, herhangi bir ara katman yazılımına ihtiyaç duymadan hem yayınla-abone ol modelini hem de sunucu-istemci modelini destekleyen, gömülü sistemlerde kullanılmaya uygun, asenkron iletişimleri destekleyen ve non-stateless iletişimleri destekleyen bir protokoldür. MCP sadece mesajlaşma için birçok özellik sunmakla kalmaz, aynı zamanda performansa da önem verir. Performans için, MCP asenkron iletişimleri destekler, dinamik başlık boyutunu kullanır ve MCP ikili protokoldür. Ayrıca, yayınla-abone ol modelinde merkeziyetsiz bir yaklaşım kullanarak gecikmeleri azaltmayı hedefler. MCP, aynı zamanda performansı önemsediği için yerel ağ haberleşmelerinin haricinde IoT uygulamalarında da kullanılabilir. IoT ağlarının closed-loop mesh yapısı yerel ağ yapılarına benzer. Benzer şekilde, IoT alanlarında da gömülü sistemler ağırlıklıdır. Gelecek çalışma olarak, IoT ağında MCP'yi kullanabilmek için, MCP'ye discovery yöntemi ve QoS özellikleri eklenerek IoT implementasyonu test edilebilir. Sonuç olarak MCP ile, hızlı yerel ağ iletişim kodu geliştirme, diğer uç noktalarla hızlı entegrasyon, verilerin izlenebilirliği sayesinde hızlı hata ayıklama, yetkilendirme ve mesaj odaklı özellikler sayesinde yüksek yönetilebilirlik, karmaşıklığı azaltarak sadelik, alt katman bileşenlerinden soyutlanarak esneklik kazanımları sağlanmış olur.

KAYNAKLAR

[1] **Marsden, B.W.**, (1986). Why are standards necessary?, bölümSection 6.1, Chartwell-Bratt Publishing, 2nd ed. sürüm, s.64–65.

[2] **Tanenbaum, A.S.** (1981). Network Protocols, *ACM Computing Surveys*, 453–489.

[3] **Thekkath, C.A.**, **Nguyen, T.D.**, **Moy, E. ve Lazowska, E.D.** (1993). Implementing network protocols at user level, *ACM SIGCOMM Conference*, s.64–73.

[4] **Xu, J.**, (2023), Handling Message Boundaries in Socket Programming, `https://enzircle.hashnode.dev/handling-message-boundaries-in-socket-programming`, erişim tarihi: 2 April 2023.

[5] **Kirch, O.**, (2002), Binary Representation Protocols, `https://web.archive.org/web/20100530225453/http://www.lst.de/~okir/blackhats/node77.html`, erişim tarihi: 10 January 2023.

[6] **Kirch, O.**, (2002), Text Based Protocols, `https://web.archive.org/web/20100530140215/http://www.lst.de/~okir/blackhats/node76.html`, erişim tarihi: 10 January 2023.

[7] **Oluwatosin, H.S.** (2014). Client-Server Model, *IOSR Journal of Computer Engineering*, 67–71.

[8] **Livens, J.**, (2023), What is Message Queuing?, `https://www.cloudamqp.com/blog/what-is-message-queuing.html`, erişim tarihi: 10 January 2023.

[9] **Eugster, P.T.**, **Felber, P.A.**, **Guerraoui, R. ve Kermarrec, A.M.** (2003). The many faces of publish/subscribe, *ACM Computing Surveys*, 114–131.

[10] **Gautam, S.**, Publisher-Subscriber Pattern, `http://web.archive.org/web/20230102074723/https://www.enjoyalgorithms.com/blog/publisher-subscriber-pattern`, erişim tarihi: 22 January 2023.

[11] **Carvalho, N.**, **Araujo, F. ve Rodrigues, L.** (2005). Scalable QoS-Based Event Routing in Publish-Subscribe Systems, *IEEE International Symposium on Network Computing and Applications*, 4, s.101–108.

[12] **Castro, M.**, **Druschel, P.**, **Kermarrec, A.M. ve Rowstron, A.** (2001). SCRIBE: A large-scale and decentralized publish-subscribe infrastructure, *International Workshop on Networked Group Communication (NGC2001)*, 3, s.101–108.

[13] **Shen, H.**, (2010). Content-Based Publish/Subscribe Systems, bölümPart XII Advanced P2P Computing and Networking, Springer, Boston, MA, 1st ed. sürüm, s.1333–1366.

[14] **Hohpe, G. ve Woolf, B.**, (2003). Correlation Identifier, bölümChapter 5 Message Construction, Addison Wesley, 1st ed. sürüm, s.324–327.

[15] **Ferraiolo, D.F. ve Kuhn, D.R.** (1992). Role-Based Access Controls, *National Computer Security Conference*, 15, s.554–563.

[16] **Sandhu, R.S.**, **Coyne, E.J.**, **Feinstein, H.L. ve Youman, C.E.** (1996). Role-Based Access Control Models, *IEEE Computer*, *29*(2), 38–47.

[17] **Bishop, M.** (2022). RFC 9114: HTTP/3, *Internet Engineering Task Force (IETF)*, s.1–57.

[18] **Fette, I. ve Melnikov, A.** (2011). RFC 6455: The websocket protocol, *Internet Engineering Task Force (IETF*, s.1–71.

[19] **Iyengar, J. ve Thomson, M.** (2021). RFC 9000: QUIC: A UDP-based multiplexed and secure transport, *Internet Engineering Task Force (IETF*, s.1–151.

[20] **Nielsen, H.**, **Mendelsohn, N.**, **Gudgin, M.**, **Hadley, M. ve Moreau, J.** (2007). SOAP Version 1.2 Part 1: Messaging Framework, *W3C REC REC-soap12-part1-20030624*, s.1–151.

[21] **Shelby, Z.**, **Hartke, K. ve Bormann, C.** (2014). RFC 7252: The Constrained Application Protocol (CoAP), *Internet Engineering Task Force (IETF)*, s.1–112.

[22] **URL-1**, AMQP Specifications, `http://web.archive.org/web/20230329193920/https://www.amqp.org/resources/specifications`, erişim tarihi: 7 April 2023.

[23] **URL-2**, MQTT Specifications, `http://web.archive.org/web/20230405181705/https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.pdf`, erişim tarihi: 7 April 2023.

[24] **URL-3**, DDS Specifications, `http://web.archive.org/web/20220120061144/https://www.omg.org/cgi-bin/doc?formal/15-04-10.pdf`, erişim tarihi: 7 April 2023.

[25] **Cotton, M.**, **Eggert, L.**, **Touch, D.J.D.**, **Westerlund, M. ve Cheshire, S.**, (2011), Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry, RFC 6335, `https://www.rfceditor.org/info/rfc6335`.

[26] **Smid, M. ve Branstad, D.** (1988). Data Encryption Standard: past and future, *Proceedings of the IEEE*, *76*, 550 – 559.

[27] **Benayache, A.**, **Bilami, A.**, **Barkat, S.**, **Lorenz, P. ve Taleb, H.** (2019). MsM: A microservice middleware for smart WSN-based IoT application, *Journal of Network and Computer Applications*, *144*, 138–154.

ÖZGEÇMİŞ

Ad Soyad: Muhammed Salih KALKAN

ÖĞRENİM DURUMU:

- **Lisans:** 2018, Hacettepe Üniversitesi, Bilgisayar Mühendisliği Bölümü.
- **Y. Lisans:** 2023, İstanbul Teknik Üniversitesi, Bilgisayar Mühendisliği Anabilim Dalı, Bilgisayar Mühendisliği Programı.

MESLEKİ DENEYİMLER:

- Kasım 2018-Ocak 2019, Yazılım Mühendisi, Tübitak Bilgem YTE
- Şubat 2019-Halen, Yazılım Mühendisi, ASELSAN

YÜKSEK LİSANS TEZİNDEN TÜRETİLEN YAYINLAR, SUNUMLAR VE PATENTLER:

- **Kalkan M. S.** and Seçinti G., "Standardization of Basic Innovative Solutions in Network Communications", 2nd International Graduate Research Symposium (IGRS), 16-18 May, 2023, Istanbul, Turkey.