

BLM3022
Ağ Teknolojileri
Proje Raporu
Proje İsmi: Sunucu Tarafı Otomatik Servis Keşfi

2021-2022 Bahar Dönemi

Grup Sorumlusu	Metin Usta	%25	Sistem tasarımı ve Sistemin test edilmesi
	İsmet Güngör	%25	Sistem tasarımı ve Servislerin gerçekleştirilmesi
	Emre Arslanoğlu	%25	Sistem hakkında ön bilgi toplanması
	Umutcan Sevdı	%25	Sistem tasarımı ve Sistemin gerçekleştirilmesi

İçindekiler

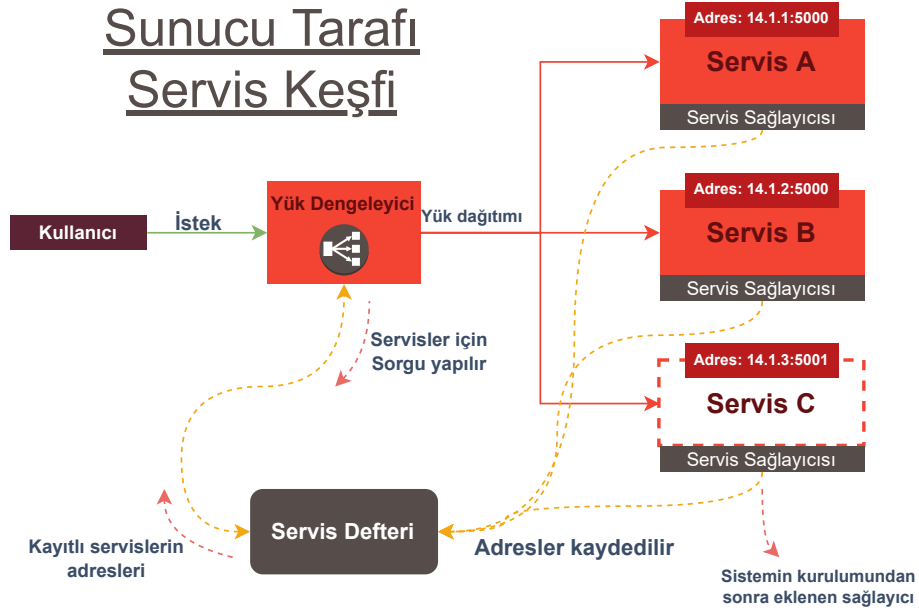
1	Proje Tanımı	2
1.1	Ön İnceleme	3
1.2	Artılar	3
1.3	Eksiler	3
2	Sistem Mimarisi	4
2.1	Kullanıcı	4
2.2	Sunucu	5
2.3	Servis	5
3	Uygulama	6
4	Performans Analizi	7
4.1	Alternatif Senaryolar	7
5	Sonuç	9

1 Proje Tanımı

Herhangi bir alanda bir hizmet veren sistemler, ilk kurulduklarında küçük boyutlu başlayabilseler de zaman içinde birçok değişime ve genişlemeye maruz kalırlar. Hatta bazen bu değişiklikler ve genişlemeler, sistemin yaşam döngüsü boyunca devam eder. Bu tür durumlarda sistemden hizmet almak için sürekli olarak yeni bir yol kullanmak gerekebilir.

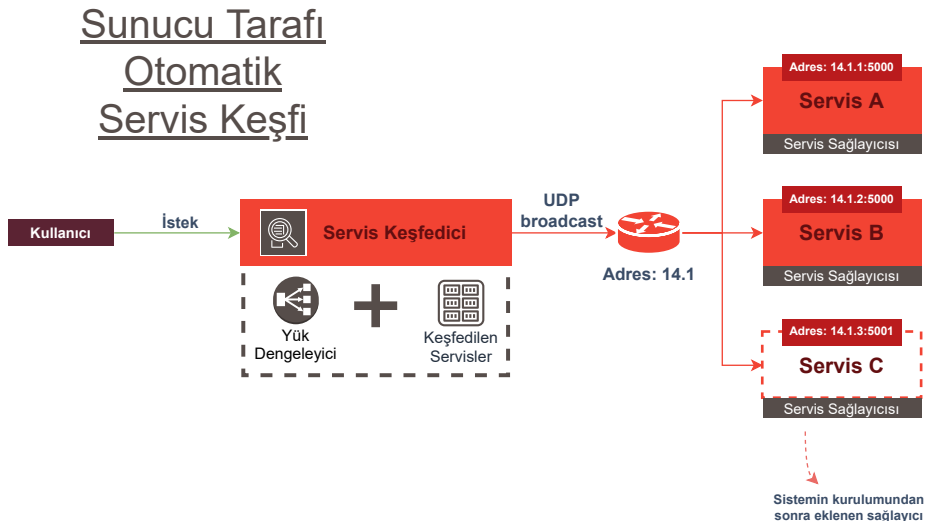
Sistemlerin yaşadığı değişimlerin ve genişlemelerin, o sistemden hizmet alınmasının önüne geçmemesi için ek araçlara ve yapılara ihtiyaç duyulur. Bu yapıya “Servis Keşfi” adı verilir. Eğer bu yapının gerçekleştirdiği işlevler tamamen sunucu tarafında yapılarak kullanıcının yükü hafifletiliyorsa buna “Sunucu Tarafı Servis Keşfi” denir.

Şekil 1’de kullanıcı, servis keşfi yapısı ve servis hizmeti sağlayan sunucular ve aradaki ilişkiler gösterilmiştir.



Şekil 1: Sunucu Tarafı Servis Keşfi

Proje kapsamında Servis Defteri(Service Registry) yapısı ile Yük Dağıtıcısı(Load Balancer) birleştirilecek ve var olan servis sağlayıcıları veritabanı benzeri bir sistemde kaydedilmek yerine UDP ile yapılacak olan yayın(broadcast) sonucu bulunacaktır. Şekil 2’de projede gerçekleştirilecek olan sistemin yapısı verilmiştir.



Şekil 2: Sunucu Tarafı Otomatik Servis Keşfi

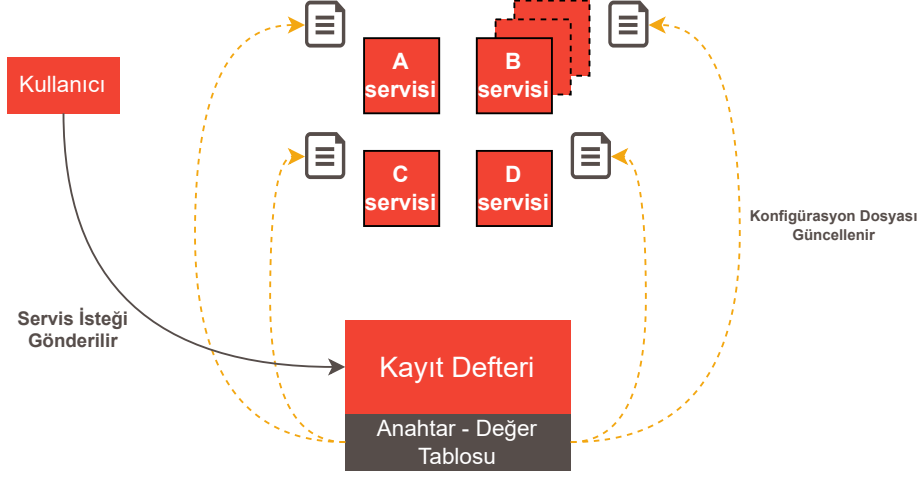
1.1 Ön İnceleme

Bu bölümde, sunucu tarafı servis keşfinde kullanılan uygulamalar ve mimarileri incelenmiştir.

Consul

Consul, servis ağlarının yönetimi, keşfi, konfigüre edilmesi, güvenliğinin sağlanması ve sağlık durumlarının kontrol edilmesi gibi birçok görevi yerine getirir.

Consul mimarisinin servis keşfi ve servislerin konfigüre edilmesi ile alakalı kısımları Şekil 3’de verilmiştir.



Şekil 3: Consul Mimarisi

Bu mimaride Kayıt Defteri yapısı proje kapsamında gerçekleştirilecek olan mimari ile benzer olarak aynı zamanda yük dengeleme işleminden de sorumludur.

Consul dışında servis keşfi ve daha birçok işlevin yerine getirilmesini sağlayan araçlara örnek olarak:

- Apache Zookeeper
- Etcd
- Eureka

verilebilir.

1.2 Artılar

Bu kısımda geliştirilecek olan sistemin diğer sistemlere ve geleneksel servis keşfi metotlarına göre artıları incelenmiştir.

1. Var olan servisleri UDP kullanarak keşfettiği için dinamik değişimlerden etkilenmez.
2. Yük dengeleyici ve servis defteri aynı yapı içerisinde bulunduğu için kullanıcının servise erişimini dolaysız bir şekilde gerçekleştirmesini sağlar.
3. Keşfedilen servislerden sağlık değeri yüksek olan ile kullanıcı tarafı bağlantı kuracağı için sunucu üzerindeki yük kaldırılmış olur.

1.3 Eksiler

Bu kısımda geliştirilecek olan sistemin diğer sistemlere ve geleneksel servis keşfi metotlarına göre eksileri incelenmiştir.

1. Birçok yapının tek bir yapı altında birleştirilmesinden dolayı single-point of failure probleminin getirdiği sorunlardan etkilenir.

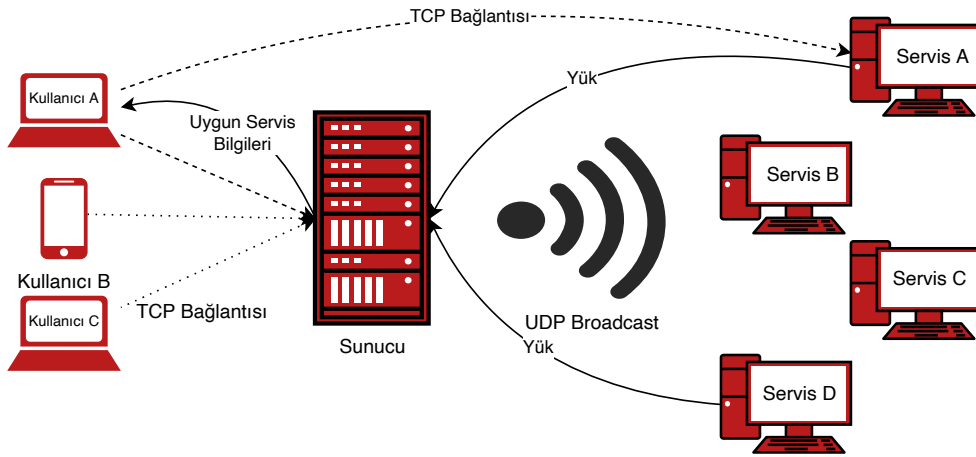
2. Adresleri yüksek bitlerde birbirinden farklı olan ve aynı alt ağ içerisinde toplanamayan servis sayısı arttıkça yayın yapılacak noktaların kaydedilmesi gerekir.

2 Sistem Mimarisi

Sistem 3 farklı aktörün ortak çalışması ile sağlanmaktadır. Bu aktörler:

- Kullanıcı
- Sunucu
- Servis

Bu aktörlerin iletişimi; kullanıcı-servis arasında TCP ile, servis-sunucu arasında UDP ile ve kullanıcı-sunucu arasında da TCP ile sağlanmaktadır. Kullanıcının sunucu tarafından belirlenmiş ve sistem içerisinde bulunan servislerden seçim yaparak bunu sunucuya iletmesi ile başlamaktadır. Sunucu tarafından UDP ile broadcast metodunda istenilen hizmet servislere iletilecek ve uygun olan servisler tarafından yoğunlukları ile birlikte geri dönüş yapılacaktır. Bu sayede sunucu kullanıcı tarafından istenilen servise uygun sunucuyu seçerken, sunucuların yoğunluklarını da inceleyerek sunucuları overwhelm etmeden "servis keşfi" ve "yük dengeleme" sağlanacaktır. Sunucu tarafından uygun bulunan servisin bağlantı için gerekli bilgileri kullanıcı ile paylaşmaktadır. Kullanıcı, sunucudan aldığı bilgiler ile servise istek atarak TCP protokolünde bağlantı kuracak ve talep ettiği hizmetten yararlanmaktadır. İletişim şeması Şekil 4'te gösterilmiştir.



Şekil 4: Sistem Şeması

2.1 Kullanıcı

Kullanıcı, sistem içerisinde hizmet almak için bulunmaktadır. Kullanıcı TCP bağlantı ilkelerini sağlayabilen herhangi bir cihaz olabilir. Kullanıcının sistemde görevi Sunucu ile TCP protokolü ile iletişim kurarak, sunucuda izin verilen hizmetler birini istemek ve dönüt beklemektir. Dönüt olarak sunucudan "en uygun" servisle iletişim kurması için gerekli bilgiler dönmektedir. Bu sayede uygun servisin bulunması için fazladan efor harcamadan istediği hizmete uygun servise ulaşabilmektedir. Sunucu tarafından servislerin sahip olduğu yük değerlendirildiği için en uygun servisten hizmet alarak trafikte zaman kaybetmeden istediği hizmete erişebilmektedir.

– Kullanıcı-Sunucu iletişimde TCP protokolünün seçilme sebepleri:

- Kullanıcının sadece TCP protokolü ile sunucu üzerinden iletişim kurması sayesinde sunucu ile olan iletişim güvenilir şekilde sağlanmaktadır ve hatalara karşı daha duyarlıdır.

- Kullanıcı-Sunucu iletişiminde veri aktarım düzeyi sınırlı bir seviyede olduğu için UDP protokolünün sunduğu hızın aksine TCP protokolünün sunduğu güvenilirlik tercih edilmiştir.
- Kullanıcı-Servis iletişiminde TCP protokolünün seçilme sebepleri:
 - Servis ile bağlantının TCP protokolü ile gerçekleştirilmesi sayesinde uygulama katmanında ekstra eklentilere gerek kalmadan güvenilir iletişim kurulmaktadır.
 - Aktarılan veri miktarının fazla olmasına rağmen kurulacak olan uçtan-uca iletişimin karşılıklı veri aktarımı sağlaması sebebiyle TCP protokolü tercih edilmiştir.

2.2 Sunucu

Sunucu, sistem içerisinde hizmet almak isteyen kullanıcıya en uygun servisin bağlantı için gerekli bilgilerini iletmek için bulunmaktadır. Sunucu, sistemden hizmet almak isteyen kullanıcı ile bağlantı kurar ve sistem içerisinde erişilebilir olan hizmetlerden seçilmiş olan hizmetin bilgisini alır. Bu hizmete uygun servisleri bulmak adına UDP protokolünde broadcast yayın yapar. Hizmeti sağlayabilecek olan servisler tarafından sunucuya yük bilgileri iletilir. Sunucu gelen yük bilgilerini karşılaştırarak en uygun servisi seçer. Seçilen servisin iletişim için gerekli olan bilgileri hizmet almak isteyen kullanıcıya iletilir.

- Sunucu-Kullanıcı iletişiminde TCP protokolünün seçilme sebepleri:
 - Sunucu ile iletişime geçen kullanıcının takibinin yapılmasını sağlamaktadır.
 - TCP bağlantısının kopma süresi tamamlanmadan bağlantıya dönüş yapılarak ek bağlantılara ve yeni iletişim başlatmaya gerek kalmadan veri aktarımını sağlamak adına tercih edilmiştir.
- Sunucu-Servis iletişiminde UDP protokolünün seçilme sebepleri:
 - Sisteme servislerin dinamik bir şekilde eklenebilmesi ve erişim için ek bağlantılar kurulması için broadcast yayın tercih edilmiştir.
 - Broadcast yayın yapılabilmesi için UDP protokolünün tercih edilmesi gerekmektedir.

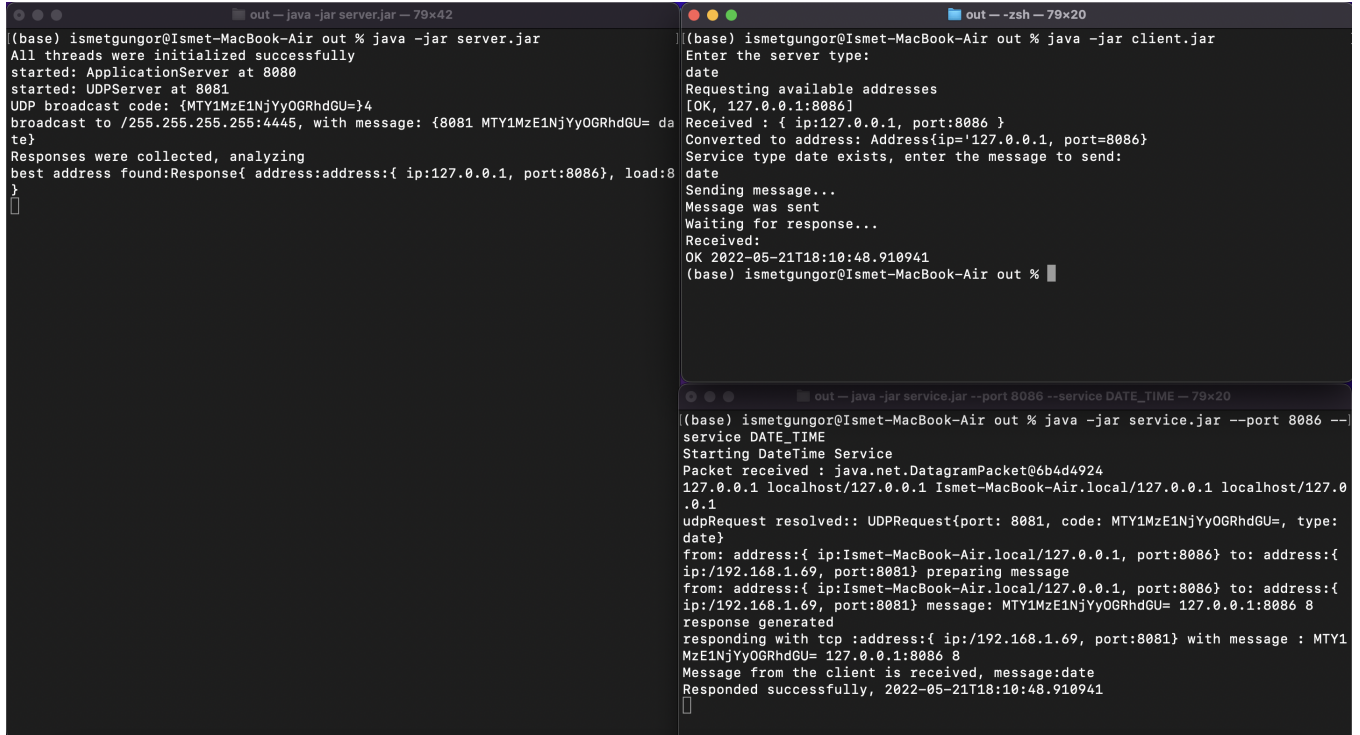
2.3 Servis

Servis, hizmet almak isteyen kullanıcıya hizmet vermek için bulunmaktadır. Servis, sunucu tarafından yapılan UDP protokolündeki broadcast yayındaki istenen hizmetin bilgisini alır. Alınan veri incelenir eğer servisin sağlayabildiği bir hizmet ise sunucu ile TCP protokolünde bağlantı kurularak servisin sahip olduğu yük bilgisi sunucuya gönderilir.

- Servis-Kullanıcı iletişiminde TCP protokolünün seçilme sebepleri:
 - Kullanıcıdan alınacak olan verinin güvenilirliği için TCP protokolü tercih edilmiştir.
 - Kullanıcıya verilecek olan hizmetin doğruluğu önemli olduğu için daha güvenilir iletişime sahip olan TCP protokolü tercih edilmiştir.
- Servis-Sunucu iletişiminde TCP protokolünün seçilme sebepleri:
 - UDP protokolü ile yapılan broadcast yayınına servisin sahip olduğu yükün ve IP adresinin sunucuya iletiminin güvenilir şekilde gerçekleştirilmesi için tercih edilmiştir.
 - Sunucuya birden fazla servisin cevap verdiği durumda gelen cevapların karışmaması ve servis sağlayıcılarının karışmaması için tercih edilmiştir.

3 Uygulama

Bu bölümde tasarlanan sistem **Java** dili ile kodlanarak çalışmaya hazır hale getirilmiştir. Yapı test edilmek için halinde saat bilgisi, verilen adresin dns bilgisini döndürme gibi servisler prototip halde kodlanmıştır. Servis çalıştırılmaya başladıktan sonra farklı işletim sistemi ve bilgisayarda da test edilmiştir. Ayrıca servis ve sunucu iki farklı cihazda da çalışabilmektedir.

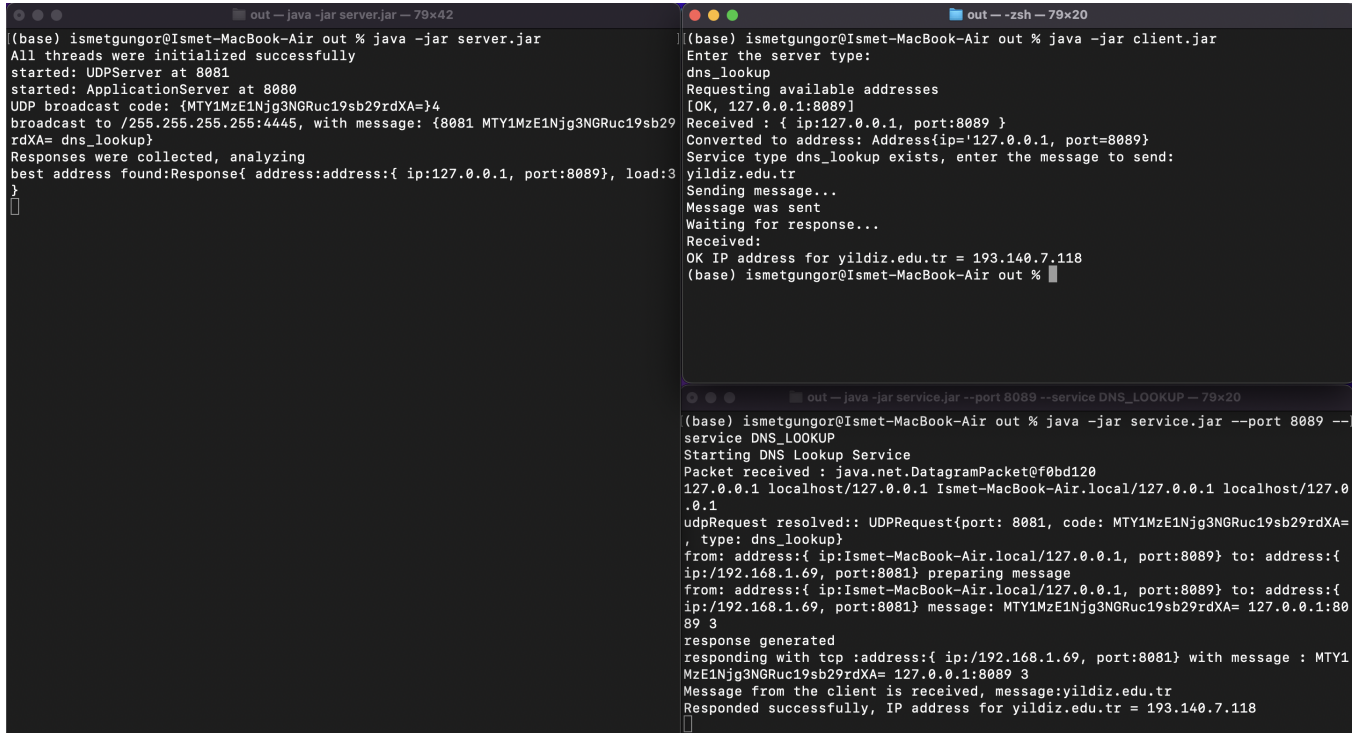


```
out — java -jar server.jar — 79x42
((base) ismetgungor@Ismet-MacBook-Air out % java -jar server.jar
All threads were initialized successfully
started: ApplicationServer at 8080
started: UDPServer at 8081
UDP broadcast code: {MTY1MzE1NjYyOGRhdGU=}{4
broadcast to /255.255.255.255:4445, with message: {8081 MTY1MzE1NjYyOGRhdGU= da
te}
Responses were collected, analyzing
best address found:Response{ address:address:{ ip:127.0.0.1, port:8086}, load:8
}
}

out — zsh — 79x20
((base) ismetgungor@Ismet-MacBook-Air out % java -jar client.jar
Enter the server type:
date
Requesting available addresses
[OK, 127.0.0.1:8086]
Received : { ip:127.0.0.1, port:8086 }
Converted to address: Address{ip='127.0.0.1, port=8086}
Service type date exists, enter the message to send:
date
Sending message...
Message was sent
Waiting for response...
Received:
OK 2022-05-21T18:10:48.910941
(base) ismetgungor@Ismet-MacBook-Air out %

out — java -jar service.jar --port 8086 --service DATE_TIME — 79x20
((base) ismetgungor@Ismet-MacBook-Air out % java -jar service.jar --port 8086 --
service DATE_TIME
Starting DateTime Service
Packet received : java.net.DatagramPacket@6b4d4924
127.0.0.1 localhost/127.0.0.1 Ismet-MacBook-Air.local/127.0.0.1 localhost/127.0
.0.1
udpRequest resolved:: UDPRequest{port: 8081, code: MTY1MzE1NjYyOGRhdGU=, type:
date}
from: address:{ ip:Ismet-MacBook-Air.local/127.0.0.1, port:8086} to: address:{
ip:192.168.1.69, port:8081} preparing message
from: address:{ ip:Ismet-MacBook-Air.local/127.0.0.1, port:8086} to: address:{
ip:192.168.1.69, port:8081} message: MTY1MzE1NjYyOGRhdGU= 127.0.0.1:8086 8
response generated
responding with tcp :address:{ ip:/192.168.1.69, port:8081} with message : MTY1
MzE1NjYyOGRhdGU= 127.0.0.1:8086 8
Message from the client is received, message:date
Responded successfully, 2022-05-21T18:10:48.910941
}
```

Şekil 5: MacOS işletim sisteminde saat servisi

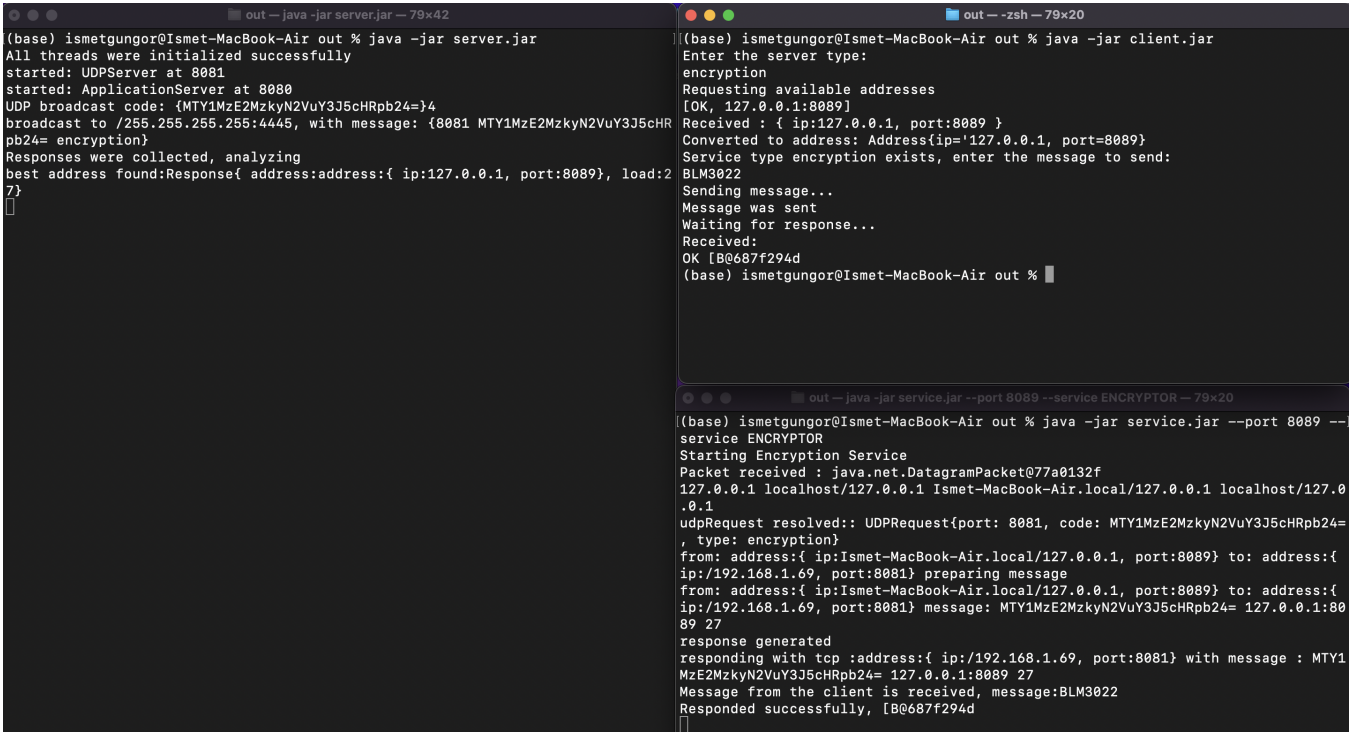


```
out — java -jar server.jar — 79x42
((base) ismetgungor@Ismet-MacBook-Air out % java -jar server.jar
All threads were initialized successfully
started: UDPServer at 8081
started: ApplicationServer at 8080
UDP broadcast code: {MTY1MzE1Njg3NGRuc19sb29rdXA=}{4
broadcast to /255.255.255.255:4445, with message: {8081 MTY1MzE1Njg3NGRuc19sb29
rdXA= dns_lookup}
Responses were collected, analyzing
best address found:Response{ address:address:{ ip:127.0.0.1, port:8089}, load:3
}
}

out — zsh — 79x20
((base) ismetgungor@Ismet-MacBook-Air out % java -jar client.jar
Enter the server type:
dns_lookup
Requesting available addresses
[OK, 127.0.0.1:8089]
Received : { ip:127.0.0.1, port:8089 }
Converted to address: Address{ip='127.0.0.1, port=8089}
Service type dns_lookup exists, enter the message to send:
yildiz.edu.tr
Sending message...
Message was sent
Waiting for response...
Received:
OK IP address for yildiz.edu.tr = 193.140.7.118
(base) ismetgungor@Ismet-MacBook-Air out %

out — java -jar service.jar --port 8089 --service DNS_LOOKUP — 79x20
((base) ismetgungor@Ismet-MacBook-Air out % java -jar service.jar --port 8089 --
service DNS_LOOKUP
Starting DNS Lookup Service
Packet received : java.net.DatagramPacket@f0bd120
127.0.0.1 localhost/127.0.0.1 Ismet-MacBook-Air.local/127.0.0.1 localhost/127.0
.0.1
udpRequest resolved:: UDPRequest{port: 8081, code: MTY1MzE1Njg3NGRuc19sb29rdXA=
, type: dns_lookup}
from: address:{ ip:Ismet-MacBook-Air.local/127.0.0.1, port:8089} to: address:{
ip:192.168.1.69, port:8081} preparing message
from: address:{ ip:Ismet-MacBook-Air.local/127.0.0.1, port:8089} to: address:{
ip:192.168.1.69, port:8081} message: MTY1MzE1Njg3NGRuc19sb29rdXA= 127.0.0.1:80
89 3
response generated
responding with tcp :address:{ ip:/192.168.1.69, port:8081} with message : MTY1
MzE1Njg3NGRuc19sb29rdXA= 127.0.0.1:8089 3
Message from the client is received, message:yildiz.edu.tr
Responded successfully, IP address for yildiz.edu.tr = 193.140.7.118
}
```

Şekil 6: MacOS işletim sisteminde DNS servisi



```
(base) ismetgungor@Ismet-MacBook-Air out % java -jar server.jar
All threads were initialized successfully
started: UDPServer at 8081
started: ApplicationServer at 8080
UDP broadcast code: {MTY1MzE2MzkyN2VuY3J5cHRpb24=}4
broadcast to /255.255.255.255:4445, with message: {8081 MTY1MzE2MzkyN2VuY3J5cHRpb24= encryption}
Responses were collected, analyzing
best address found:Response{ address:address:{ ip:127.0.0.1, port:8089}, load:27}

(base) ismetgungor@Ismet-MacBook-Air out % java -jar client.jar
Enter the server type:
encryption
Requesting available addresses
[OK, 127.0.0.1:8089]
Received : { ip:127.0.0.1, port:8089 }
Converted to address: Address(ip='127.0.0.1, port=8089)
Service type encryption exists, enter the message to send:
BLM3022
Sending message...
Message was sent
Waiting for response...
Received:
OK [B0687f294d]
(base) ismetgungor@Ismet-MacBook-Air out %

(base) ismetgungor@Ismet-MacBook-Air out % java -jar service.jar --port 8089 --service ENCRYPTOR
Starting Encryption Service
Packet received : java.net.DatagramPacket@77a0132f
127.0.0.1 localhost/127.0.0.1 Ismet-MacBook-Air.local/127.0.0.1 localhost/127.0.0.1
udpRequest resolved:: UDPRequest{port: 8081, code: MTY1MzE2MzkyN2VuY3J5cHRpb24=, type: encryption}
from: address:{ ip:Ismet-MacBook-Air.local/127.0.0.1, port:8089} to: address:{ ip:/192.168.1.69, port:8081} preparing message
from: address:{ ip:Ismet-MacBook-Air.local/127.0.0.1, port:8089} to: address:{ ip:/192.168.1.69, port:8081} message: MTY1MzE2MzkyN2VuY3J5cHRpb24= 127.0.0.1:8089 27
response generated
responding with tcp :address:{ ip:/192.168.1.69, port:8081} with message : MTY1MzE2MzkyN2VuY3J5cHRpb24= 127.0.0.1:8089 27
Message from the client is received, message:BLM3022
Responded successfully, [B0687f294d]
```

Şekil 7: MacOS işletim sisteminde Encryption servisi

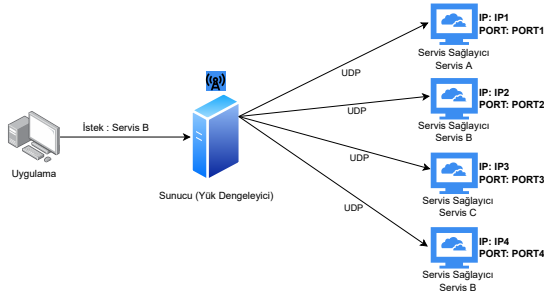
4 Performans Analizi

Bu bölümde tasarlanan ve gerçekleşen sistem performansı değerlendirilmiş, sistemin pozitif ve negatif yönleriyle analiz edilmiştir. Ayrıca negatif sayılabilecek durumlara karşı geliştirilen alternatif yöntemler geliştirilmiştir.

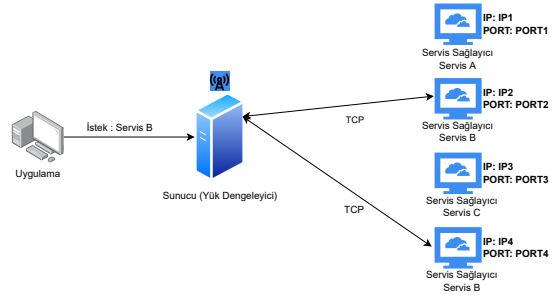
Sistemin performansını değerlendirirken sayısal bir metrik belirlemek oldukça güçtür. Bunun sebebi sistem anlık olarak servisleri keşfetmekte ekstra bir işlem gücü harcarken, sunucuları ve bu sunucuların verdiği servisleri belirlemek adına yapılan güncelleme işlemlerinden kurtarılmaktadır. Ayrıca anlık olarak sisteme eklenen, sistemden düşen, bozulan veya ağır yük altına giren sunucuların durumu yük dengeleyici olarak kullanılan sunucu tarafından hızlıca güncellenecektir. Sistemin performansını belirleme için ise bir yük dengeleyici olarak kullanıldığında saniyede verilen hizmet sayısı bir değerlendirme fonksiyonu olarak kullanılabilir.

4.1 Alternatif Senaryolar

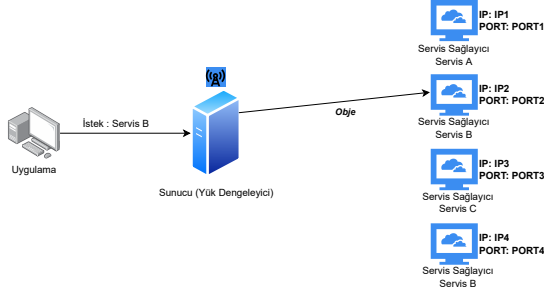
Sistem ilk tasarlandığında yük dağıtıcı olarak belirlenen sunucu uygulama tarafından aldığı veriyi uygun sunucuya ilettikten sonra hizmet sonucunu tekrar yük dağıtıcı üzerinden uygulamaya iletmekteydi. Şekil 8’de sırayla bağlantı adımları gösterilmiştir.



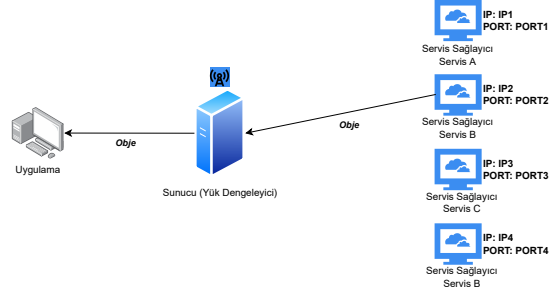
(a) UDP Bağlantısı



(b) TCP Bağlantısı



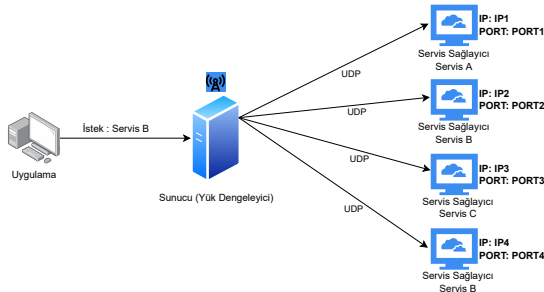
(c) Servise Sağlayıcıya Veri Aktarımı



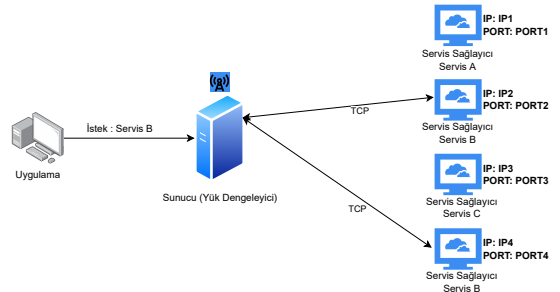
(d) Uygulamaya Veri Aktarımı

Şekil 8: Yük Dengeliyici Servis Sağlayıcı İletişim Modeli

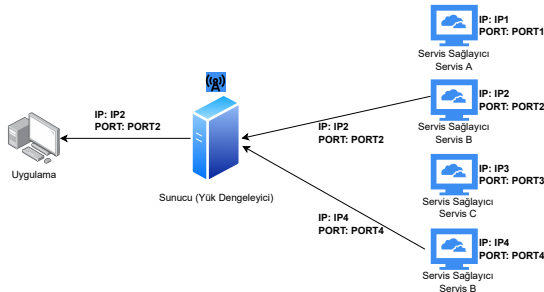
Bu tasarım darboğaz, buffer overflow gibi sorunların oluşma olasılığını arttırdığı için İşlem kapasiteni arttırmak ve yük dağıtıcı yükünü azaltmak adına sistem tasarımında iyileştirmeler yapılmıştır. Yük dağıtıcı servis sağlayıcıları bulduktan sonra uygun servis sağlayıcının IP ve port adreslerini uygulamaya iletir. İşlemler sistemin uç noktalarında gerçekleştirilir. Bu sayede sunucu ağır yük altında bırakılmaz. 9’da tasarlanan bu sistem adım adım görülmektedir.



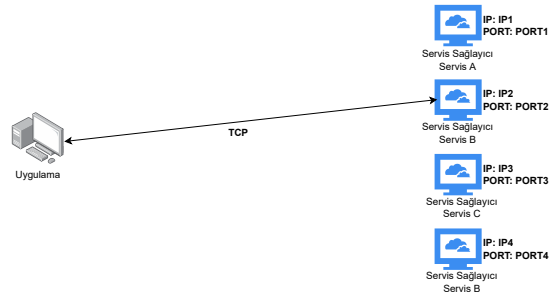
(a) UDP Broadcast



(b) TCP Bağlantısı



(c) Uygulama ile IP/Port Paylaşımı



(d) Uygulama ile Sistem arasında TCP

Şekil 9: Uygulama Servis Sağlayıcı İletişim Modeli

5 Sonuç

Tasarlanan sistem kodlanmış ve test edilmiştir. Gerçeklenen sistem stabil olarak çalışmakla birlikte farklı platformlar üzerinde çalışabilmektedir. Otomatik servis keşfi yöntemi bu rapor kapsamında yük dengeleyici görevinde kullanılmaktadır. Fakat servis bilgisini güncelleme, ağı ölçekleme gibi diğer farklı problemler için de çözüm olarak kullanılabilir. Az sayıda servis ve sunucuya sahip sistemlerde verimi ölçülebilir bir artım sağlamasa bile yüksek sayıda servis hizmeti veren sunucularda birçok probleme verimli çözümler üretebilmektedir.