T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

İÇERİK DAĞITIM AĞLARININ İNTERNETİN DİNAMİKLİĞİNE UYUMLU HALE GETİRİLMESİ: EN YAKIN KAYNAK SUNUCU

Nima NAJAFLOU

Uygundur

DANIŞMAN
Dr. Öğr. Üyesi Gülsüm Zeynep GÜRKAŞ AYDIN

II. DANIŞMAN
Doç. Dr. Berk CANBERK

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

İSTANBUL-2019

PhotoScan by Google Photos

T.C.
İSTANBUL ÜNİVERSİTESİ-CERRAHPAŞA
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

İçerik Dağıtım Ağlarının İnternetin Dinamikliğine Uyumlu Hale Getirilmesi: En Yakın Kaynak Sunucu

Nima NAJAFLOU

DANIŞMAN
Dr. Öğr. Üyesi Gülsüm ZeynepGÜRKAŞ AYDIN

II. DANIŞMAN
Doç. Dr. Berk CANBERK

Bilgisayar Mühendisliği Anabilim Dalı

Bilgisayar Mühendisliği Programı

İSTANBUL-2019

Bu çalışma, Tarih girmek için burayı tıklatın. tarihinde aşağıdaki jüri tarafından ....., ...nda ..... olarak kabul edilmiştir.

Tez Jürisi

Unvan Adı SOYADI(Danışman)
İstanbul Üniversitesi-Cerrahpaşa
Fakülte

Unvan Adı SOYADI
Üniversite
Fakülte

Unvan Adı SOYADI
Üniversite
Fakülte

Unvan Adı SOYADI
Üniversite
Fakülte

Unvan Adı SOYADI
Üniversite
Fakülte

20.04.2016 tarihli Resmi Gazete'de yayımlanan Lisansüstü Eğitim ve Öğretim Yönetmeliğinin 9/2 ve 22/2 maddeleri gereğince; Bu Lisansüstü teze, İstanbul Üniversitesi-Cerrahpaşa'nın abonesi olduğu intihal yazılım programı kullanılarak Lisansüstü Eğitim Enstitüsü'nün belirlemiş olduğu ölçütlere uygun rapor alınmıştır.

Bu tez, İstanbul Üniversitesi-Cerrahpaşa Bilimsel Araştırma Projeleri Yürütücü Sekreterliğinin ..... numaralı projesi ile desteklenmiştir.
Bu tez, ... numaralı ..... projesi ile desteklenmiştir.

ÖNSÖZ

Bu tez çalışması süresince bana her konuda destek olan, sabırla ve özveriyle tüm sorularımı cevaplayıp beni yönlendiren saygıdeğer danışman hocalarım Dr. Öğr. Üyesi GÜLSÜM ZEYNEP GÜRKAŞ AYDIN hanım'a ve Doç.Dr. Berk Canberk'e , uzakta olsalar bile her daim destekleriyle yanımda hissettiğim, bitmek tükenmek bilmeyen sabır ve sevgileriyle beni bugünlere getiren aileme, hayatıma girdiği andan itibaren sevgisi ve desteğiyle güzelleştiren ve zenginleştiren eşim Hülya Najaflou'ya ve minik kalbiyle hayatıma anlam katan, geleceğe dair planlarımı yeniden şekillendiren oğlum Parsa'ya sonsuz teşekkürlerimi sunarım.

Haziran 2019 Nima NAJAFLOU

İÇİNDEKİLER

Sayfa No

ÖNSÖZ ..... **iv**
İÇİNDEKİLER ..... **v**
ŞEKİL LİSTESİ ..... **vii**
TABLO LİSTESİ ..... **viii**
SİMGE VE KISALTMA LİSTESİ ..... **ix**
ÖZET ..... **x**
SUMMARY ..... **xii**
1. GİRİŞ **1**
2. GENEL KISIMLAR **3**
2.1 İÇERİK DAĞITIM AĞLARI TANIMI 3
2.2 İÇERİK DAĞITIM AĞLARI TERMİNOLOJİSİ 3
2.3 GENEL İÇERİK DAĞITIM AĞI MİMARİSİ 4
2.4 KONVANSİYONEL İÇERİK DAĞITIM AĞI MİMARİLERİ 5
2.4.1 İstemci-Sunucu Mimarisi 5
2.4.2 Uçbirimden Uçbirime Mimari 6
2.4.3 Çoklu İçerik Dağıtım Ağı 6
2.5 İÇERİKLERİN İÇERİK DAĞITIM AĞINA KONUMLANDIRILMASI 7
2.5.1 Kaynaktan Çekme Yöntemi 7
2.5.2 Kopya Sunucusuna İletme Yöntemi 8
2.6 VEKALETEN ÖNBELLEĞE ALMAK 8
2.6.1 Vekil Sunucular 8
2.6.2 İleri Vekil Sunucuları 9
2.6.3 Ters Vekil Sunucuları 9
2.6.4 İçerik Dağıtım Ağlarında Ters Vekil Sunucular 9
2.7 YÜK DAĞITIMI 10
2.7.1 Yük Dağıtım Türleri 11
2.7.2 Sağlık Kontrolü ve Yük Dengeleme Algoritmaları 11
2.8 DNS 12
2.8.1 Aramak ve Yönlendirmek 12
2.8.2 DNS'in CDN'lerdeki rolü 13

2.8.3 DNS Yük Dengelemesi 13
2.8.4 Anycast DNS 13
2.8.5 İstek Yönlendirme 13
3. MALZEME VE YÖNTEM 15
3.1 PROBLEM TANIMI 15
3.2 ÖNERİLEN GENEL SİSTEM MİMARİSİ 15
3.2.1 Yardımcı Modül 19
3.2.2 Merkezi Modül 19
3.3 ÖNERİLEN ALGORİTMA 21
3.3.1 NOS Engine 21
3.3.2 Gecikme API 23
3.3.3 Sunucu Yük API 24
3.4 KULLANILAN ARAÇLAR 25
3.4.1 Medianova 25
3.4.2 NGINX 26
3.4.3 SSH 26
3.4.4 Apache JMeter 26
3.4.5 Vultr 27
3.4.6 Python 28
3.5 TEKNOLOJİNİN BİLİNEN GÜNCEL DURUMU 29
3.6 TEST ORTAMI 31
3.7 GERÇEKLEŞTİRİLEN TESTLER 33
4. BULGULAR 34
4.1 ANALİZLER 34
4.2 SONUÇLAR 36
5. TARTIŞMA VE SONUÇ 39
KAYNAKLAR 40
EKLER 42
ÖZGEÇMİŞ 44

ŞEKİL LİSTESİ

Sayfa No

Şekil 2.1: İçerik Dağıtım Ağı Genel Mimarisi. 4
Şekil 2.2: Origin Pull CDN. 7
Şekil 2.3: Push CDN. 8
Şekil 3.1: Önerilen Bindirmeli Ağ Mimarisi. 16
Şekil 3.2: NOS tabanlı CDN'in çalışması. 16
Şekil 3.3: Yardımcı Modül. 16
Şekil 3.4: Merkezi Modül. 20
Şekil 3.5: Merkezi ve Yardımcı Modülleri. 20
Şekil 3.6: Gecikme Duyarlı ve Dalgasız Bindirmeli Ağ Yönlendirme Algoritması. 22
Şekil 3.7: Test Ortamı. 31
Şekil 4.1: Kaynak sunucu ile iki OriginShield arasındaki gecikme. 34
Şekil 4.2: Bir OriginShield'e ait yük değeri. 35
Şekil 4.3: OriginShield A ile diğer OriginShield'ler arasındaki gecikme. 36
Şekil 4.4: Hit Oranı NOS ile ve NOS Olmadan. 37
Şekil 4.5: Uçtan uca yönlendirme yolu gecikmeleri NOS ile ve NOS Olmadan. 37

TABLO LİSTESİ

Sayfa No

Tablo 3.1: Gecikme Duyarlı ve Dalgasız Bindirmeli Ağ Yönlendirme Algoritması kullanılan kısaltmalar 21

Tablo 3.2: Test Parametreleri 32

SİMGE VE KISALTMA LİSTESİ

Simgeler **Açıklama**

a : Aaaa
b : Aaaa
c : Aaaa
ds : Aaaa
es : Aaaa
F : Aaaa

Kısaltmalar **Açıklama**

A : Aaaa
bk : Aaaa
CL : Aaaa
km : Aaaa
N : Aaaa
x : Aaaa
zy : Aaaa

ÖZET

YÜKSEK LİSANS TEZİ

İçerik Dağıtım Ağlarının İnternetin Dinamikliğine Uyumlu Hale Getirilmesi: En Yakın Kaynak Sunucu

Nima NAJAFLOU

İstanbul Üniversitesi-Cerrahpaşa

Lisansüstü Eğitim Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman : Dr. Öğr. Üyesi Gülsüm Zeynep GÜRKAŞ AYDIN

II. Danışman : Doç. Dr. Berk CANBERK

İçerik Dağıtım Ağları (CDN) içerik üreticilerinin (Origin) resim, video, vb. içeriklerini dünyanın farklı noktalarındaki kullanıcılara hızlı bir şekilde ulaştırmayı hedefler. İçerik Dağıtım Ağlar performanslı içerik servisi sağlamak için Origin tarafını optimize ederler. İçerikleri dünya üzerinde stratejik noktalarda önbelleğe alırlar. Bunları yapsalar bile içerik servisi performansında düşüşler yaşamaktadırlar. Bunun nedeni Origin ile CDN arasındaki ağın birden çok İnternet Servis Sağlayıcı (ISP) tarafından yönetilmesidir. ISP'ler kendi aralarında dinamik olarak bağlantılar kurmakta ve yönlendirmeler yapmaktadır. Bu kararlarında fiyat, kullanıcı sayısı ve rekabet gibi parametreleri göz önüne almaktadırlar. En kısa yolu düşünmemektedirler. Bu nedenle içeriğin Origin'den CDN platformuna ulaşma yolu sık sık değişmekte ve en kısa yol olmamaktadır. Bu yüzden CDN'ler müşterilere vaat ettikleri sürelere uyamamaktadırlar. ISP'lerin yönettikleri fiziksel ağlar kontrolümüzde olmadığından çalışmamızda bunların üzerine bir overlay ağ kurup overlay ağını yönetecek bir Ağ Yönetim Sistemi geliştirdik. Bu sistem ISP'lerin yaptıkları değişiklikleri hızlıca anlamaktadır. İçeriğin Origin sunucusundan en hızlı CDN platformumuza hangi yoldan geleceğine karar vermektedir. Bunun için sunucuların yüklerine (CPU ve ağ trafiği yükü, bellek ve disk doluluğu) ve Origin'e olan gecikmesine baktık. Bunlara göre de overlay ağda dinamik olarak yolları belirledik. Overlay ağındaki sunucularımızın fazla yüklenmelerine engel olmak için sunucuların işlemci ve trafik yoğunluklarının, disk ve bellek doluluklarının da hesaplamamıza kattık. Ağ Yönetim

Sistemimiz gecikme süreleri ve sunucu yoğunluklarına göre en uygun yolları belirlemektedir. Böylece CDN hizmetinin performansı daha stabil öldüğünü gözlemledik.

Mayıs 2019, ... sayfa.

Anahtar kelimeler: İçerik Dağıtım Ağı, CDN, Content Delivery Network, bindirmeli ağ, Overlay Network, Request Routing

SUMMARY

M.Sc. THESIS

THESIS TITLE

Nima NAJAFLOU

Istanbul University-Cerrahpasa

Institute of Graduate Studies

Department of Computer Engineering

Supervisor : Assist. Prof. Dr. Gülsüm Zeynep GÜRKAŞ AYDIN

Co-Supervisor : Assoc. Prof. Dr. Berk CANBERK

Although Content Delivery Networks (CDN) do their best to quickly deliver the contents, Internet Service Providers (ISP) and network conditions cause unexpected changes on the routing paths. Hence, it gets difficult for CDNs to adhere to the delay promises within the Service Level Agreements. This paper presents a jitter-free overlay routing framework for CDNs, which struggle to adapt timely content delivery to end users due to the dynamicity of the Internet. The Nearest Origin-Shield (NOS) framework which we propose in this paper overcomes this issue by creating an overlay network on top of the existing physical ISP networks and dynamically determining the fastest routes to the content sources (Origins) on this overlay network. NOS keeps track of changes in underlay network through periodically measuring the end-to-end routing paths delays via Helper Modules. Based on the underlay network delay measures and the load (i.e., CPU, disk I/O and network usage) of the overlay

network entities, NOS Central Module determines the fastest routes to the Origins using our novel Delay-Aware and Jitter-Free Overlay Routing Algorithm. Hence making it possible for CDNs to provide jitter-free and timely content deliveries meanwhile ensuring the cache servers not to be overloaded. We created a new testbed for performance evalua- tions with real CDN servers, Origin accounts and measurements. Results show that, NOS provides shorter, more stable and jitter- free routing paths with jitter gains up to 98% and improves the Hit ratio by 0.74%.May 2019, ... pages.

Keywords: CDN, ISP, routing path, overlay, jitter, delay, hit ratio

1. GİRİŞ

İçerik Dağıtım ağları dünyanın farklı noktalarında dağıtık bir şekilde varlık noktaları bulunmaktadır. Bu dağıtık yapı sayesinde hizmet alanların içerikleri kullanıcılara hızlı bir şekilde ulaştırmaktadır. Sistemimizde kullanıcıların istekte bulundukları içerikler kaynak sunucusundan kullanıcılara en yakın varlık noktasına taşınarak önbelleğe alınmaktadır. Bundan sonra o bölgeden aynı içeriğe gelen istekler, içerik hazır olduğundan çok hızlı bir şekilde cevaplanabilmektedir.

İçerik Dağıtım Ağları bu dağıtık yapıyla kullanıcılara yakınlaşabilseler de CDN hizmetlerinin performansında düşüşler yaşamaktalar. Bunun sebebi kaynak sunucular ile Dağıtım ağlarındaki sunucular arasındaki ağların birçok ISP tarafından yönetilmesidir. ISP'ler aradaki ağlar üzerinde dinamik olarak yönlendirmeler yapmaktadır. Bu da içeriklerin Origin'den CDN platformuna ulaşma sürelerinde büyük dalgalanmalara neden olmaktadır. Bu yüzden kullanıcılara içerikleri hızlı ulaştırılması zorlaşmaktadır. Süre kısıtlarına uyamamalarına neden olmaktadır. Çalışmamıza bu gerekçeyle başlatmaya karar verdik. Hızlı içerik servisiyle global ölçekte başarılı olabilmek için CDN performansını ISP'lerin yaptıkları değişikliklerden olabildiğince yalıtmamız gerekmektedir.

Hedeflediğimiz başarı, içeriklerin Origin'lerden sunucularımıza her zaman en kısa yol üzerinden gelmeleriyle olacaktır.

ISP'ler 3. katman'da çalışırlar ve kontrolümüzde değillerdir. CDN hizmeti ise 7. Katman'dadır. Çalışmamızda kontrol edemediğimiz ve performansımızı düşüren 3. Katman sorunlarına daha üst katmanlarda bir bindirmeli ağ (Overlay Network) kurarak yeni bir çözüm geliştirilgeliştirdik. Önerdiğimiz çözüm sadece gecikme sürelerine bakmamaktadır. Buna ek olarak sunucu yoğunluklarını da düşünmektedir. Hızlı içerik servisi için belirlenen yol bir sunucunun başka bir sunucu vasıtasıyla Origin'e ulaşmasını gerektirebilir.

ISP'lerin alttaki ağlarda yaptıkları değişikliklere göre overlay ağını yönetecek bir de Ağ Yönetim Sistemi tasarlandı. Bu sistem alttaki ağda meydana gelen değişiklikleri dinlemektedir. Buna göre overlay ağında Origin'lere en kısa yolları belirlemektedir. En kısa yol için sunucularımızla Origin'ler arası mesafelerin gecikmelerine bakılması lazımdır. Overlay

ağındaki sunucularımızın fazla yüklenmelerine engel olmak için sunucularımızın işlemci ve trafik yoğunluklarının, disk ve bellek doluluklarının da hesaplamalarımıza kattık. Ağ Yönetim Sistemi gecikme süreleri ve sunucu yoğunluklarına göre en uygun yolları belirlemektedir.

Geliştirdiğimiz yeni Ağ Yönetim Sistemi sayesinde ISP'ler nedeniyle içerik servisi performansımızda düşüşler yaşanmayacaktır. Her zaman gecikmesi daha düşük yollar seçileceğinden içerik servis hızı daha yüksek olacaktır. Sistemimiz en kısa yolu belirlemede sunucu yoğunluklarına da bakacağı için sunucularımız satürasyona uğramayacaklardır. Tüm bunlar içerik servisi kalitesinin artmasını sağlayacaktır.

2. GENEL KISIMLAR

2.1 İÇERİK DAĞITIM AĞLARI TANIMI

İçerik Dağıtım Ağı, içeriği en yakın noktadan kullanıcılara ulaştırmayı amaçlayan ortak bir ağ kümesidir. İçerik Dağıtım Ağlarında (CDN) içerikler, Uç Sunucular (Edge Servers) veya Önbellek sunucuları (Cache Servers) adı verilen ayna sunucularında önbelleğe alınarak kullanıcılara daha hızlı bir şekilde en yakın noktadan sunulur.

2.2 İÇERİK DAĞITIM AĞLARI TERMİNOLOJİSİ

• **İçerik Dağıtımı (Content Delivery)** : Son kullanıcılara isteklerine göre içerik sunma eylemidir.

• **İçerik (Content):** Her tür dijital veri kaynağına atıfta bulunur ve iki ana bölümden oluşur: Kodlanmış Medya ve Meta Veriler. Kodlanmış Medya, ses, video, resimler, web sayfaları veya belgeler gibi statik, dinamik ve sürekli medya verileri içerir. Meta veriler, içerik tanımının ve bu içeriğin tanımlanmasına veya yönetimine izin veren içeriğe ilişkin verilerdir.

• **İçerik Dağıtım Ağ Sağlayıcısı (CDN Provider):** İçerik Dağıtım Ağ (CDN) Sağlayıcı terimi, içeriğini zamanında ve güvenilir bir şekilde sunmak için, ortak altyapı veya kurum içi altyapı üzerinde, İçerik Sağlayıcılarına altyapı olanakları sağlayan kuruluş veya şirket anlamına gelir. CDN Tedarikçi örnekleri Akamai, Cloudfront, Limelight vb. firmalardır.

• **İçerik Sağlayıcı (Content Provider):** içerik sağlayıcı veya müşteri, dağıtılacak web nesnelerinin Tekdüzen Kaynak Bulucu (URL) ad alanını delege eden kısımdır. Aslında, bir İçerik Dağıtım Ağı ile herhangi bir tür veri iletmeyi hedefleyen herhangi bir müşteriye, İçerik Sağlayıcı denilebilir.

• **Kaynak Sunucu (Origin Server):** içeriğin temel olarak tutulduğu ve CDN sağlayıcısı tarafından iletilmesi beklenen sunucu.

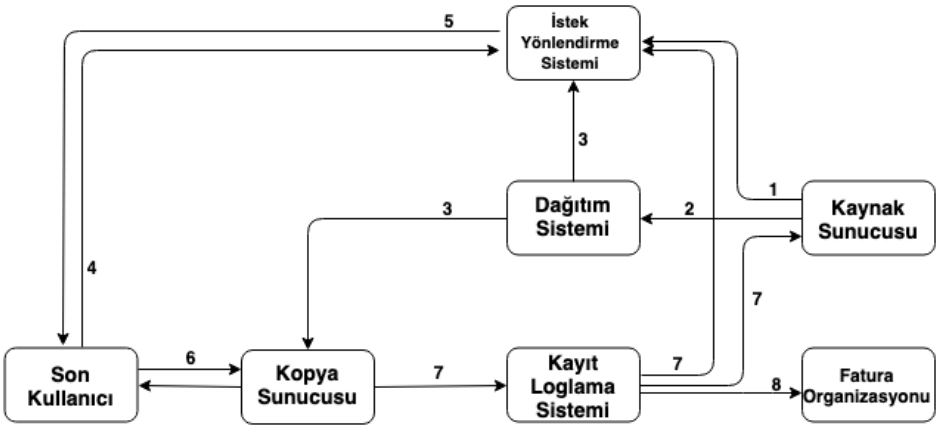
• **Kenar Sunucusu (Edge Server):** veya Vekil Sunucu (Surrogate Server), kaynak sunucu adına istekleri alacak ve istenen içeriği iletecek olan kaynak sunucu dışında başka bir sunucudur.

• **Kaynak Sunucu Kalkan (OriginShield) Sunucusu:** Kenar sunucusuna benzeyen ancak CDN mimarisindeki kenar sunucuları ve kaynak sunucuları arasında oturan ve birden fazla kenar sunucusundan gönderilen isteklere cevap vermekle yükümlü olan bir sunucudur. Kaynak kalkanları (OriginShield), Kaynak sunucusunun yükünü azaltmaya yardımcı olur ve ekstra bir önbellek düzeyi sağlar (Akamai, n.d.) (Limelight Networks, n.d.).

• **Son Kullanıcı:** veya müşteriler, içerik sağlayıcısının içeriğine erişme eğiliminde olan varlıklardır (insan veya botlar).

2.3 GENEL İÇERİK DAĞITIM AĞI MİMARİSİ

CDN için genel bir mimari, şekil 2.1'de gösterilmektedir. Bu resimde görebileceğiniz gibi, CDN için genel bir mimari, yedi bileşenden oluşur: son kullanıcı, kopya sunucuları (Replica Servers), kaynak sunucuları, fatura organizasyonu, istek yönlendirme sistemi, dağıtım sistemi ve kayıt ve loglama sistemi. Bu resimdeki çizgiler, bu bileşenlerin her biri arasındaki ilişkiyi göstermektedir (M. Green, November 2000) (M. Day, March 2001).



Şekil 2.1: İçerik Dağıtım Ağı Genel Mimarisi.

1. Origin Server, CDN tarafından yayınlamak istediği içeriğin URI'sini Talep Yönlendirme Sisteminden alır.
2. Kaynak Sunucu, Dağıtılmasını istediği içeriği Dağıtım Sistemine iletir.
3. İçerik, Kopya Sunucularına kopyalanır; Ek olarak, Dağıtım Sistemi her Kullanıcının isteği için en uygun Kopya Sunucularını seçmede yardımcı olmak için İstek Yönlendirme Sistemi ile etkileşime girer.
4. Son Kullanıcı, doğrudan bir CDN'i işaret eden bir URI ile veya CName vasıtasıyla içeriği ister. Bu istekler İstek Yönlendirme Sistemine yönlendirilir.
5. İstek Yönlendirme Sistemi, istekleri en iyi Çoğaltma Sunucusuna- genellikle en yakın Çoğaltma Sunucusu- yönlendirir.
6. Seçilen Çoğaltma Sunucusu (Kopya sunucusu) içeriği istemi yapan son kullanıcıya gönderir, ayrıca isteğin detayları ve içeriğin bilgilerini Kayıt ve Loglala Sistemine kaydeder.
7. Kayıt ve Loglama Sistemi, talep edilen içeriklerin istatistiklerini ve bilgilerini oluşturur ve bunları Faturalama organizasyonuna gönderir. Bu istatistikler ayrıca, Kaynak Sunucular ve İstek Yönlendirme Sistemi tarafından da kullanılır.
8. Faturalama Organizasyonu bu istatistikleri, içerik dağıtımına katılan her bir tarafın kullanımını ve tüketim payını hesaplamak için kullanır.

2.4 KONVANSİYONEL İÇERİK DAĞITIM AĞI MİMARİLERİ

2.4.1 İstemci-Sunucu Mimarisi

Akamai şirketi muhtemelen istemci-sunucu mimarisini (Client Server Architecture) kullanan en ünlü CDN firması örneğidir. Akamai dünyada yaklaşık 139 ülkedeki 1.600 ağda 239.000'den fazla sunucu işletiyor (John Dilley, 07 November 2002) (Akamai, n.d.).

İstemci-sunucu mimarisinde, CDN firmaları dünyanın çeşitli noktalarında kendi kopya sunucularını (Replica Servers) çalıştırır. Bu mimaride müşterilerin veya son kullanıcıların içerikleri bu sunuculardan dağıtılmaktadır. Son kullanıcılardan gelen tüm istekler bu kopya sunucularına iletilir.

2.4.2 Uçbirimden Uçbirime Mimari

Uçbirimden uçbirime (Peer-to-Peer) kelimesi, geleneksel istemci-sunucu mimarisine dayanmayan ve bunun yerine hizmetçi tabanlı (Servant Based) bir mimariyi kullanan merkezi olmayan dağıtılmış bir sistemi ifade eder. Uçbirimden uçbirime mimariler (peer-topeer architecture), İstemci ve İçerik arasındaki merkezi ara sunucuları ortadan kaldırmak için kullanılır. En önemli özellikleri, kabul edilebilir bir performans, gecikme ve bağlantı seviyesine ulaşırken, farklı konumlardaki sunucu arızasına kendilerini uyarlama yetenekleridir (Stephanos Androutsellis-Theotokis, December 2004).

Bazı P2P çözümleri, süper eşler olarak adlandırılan özel sunucular kullanır. Bu sunucular ağ oluşturma, içerik indeksleme ve arama gibi görevler için merkezi kontrol noktaları olarak kullanılır. Verilen bu tanımla P2P mimarisi çözümlerini iki farklı gruba ayırabiliriz; Hibrit ve Saf P2P mimarileri (T. Oh-ishi, 2003).

Yukarıda açıklanan aynı teknik, bir İçerik Teslimi mimarisi geliştirmek için de uygulanmıştır. Bir P2P İçerik Dağıtım Ağında içerikler dağıtım ağını oluşturan son kullanıcı terminallerinden gönderilir ve alınır. P2P İçerik Dağıtım Ağların avantajları: P2P CDN'ler sunucu maliyetini düşürür, istemci-sunucu CDN'lerindeki merkezi mekanizmaya bağlı olmadan çalışabilirler, ayrıca daha dayanıklı ve ağı daha kullanışlı kullanırlar (Stephanos Androutsellis-Theotokis, December 2004).

2.4.3 Çoklu İçerik Dağıtım Ağı

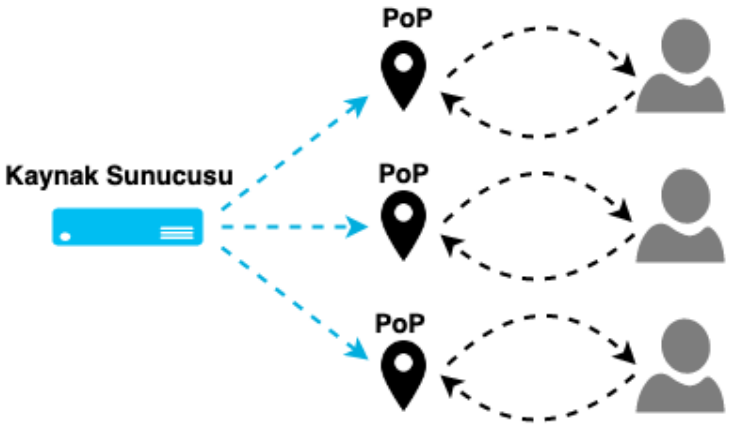
Çoklu içerik dağıtım ağları (Multi CDN) tek bir CDN firmasını kullanarak içerik dağıtımı yapmanın dezavantajlarından döğmüş bir CDN mimarisidir. CDN'ler aslında coğrafi olarak geniş bir bölgede daha hızlı hizmet verebilmek için kullanılan servislerdir. Ama tek bir CDN firması farklı zamanlarda ve farklı bölgelerde farklı performans sergileyebilirler. Örneğin bir CDN firması belli bir zaman diliminde yük altında olup isteklere daha yavaş cevap verebilir. Veya aynı CDN A bölgesinde 30ms gecikme ile cevap verirken B bölgesinde 300ms gecikme gösterebilir. Bu çekinceler içerik sağlayıcı firmaların birden fazla CDN kullanmaya daha sıcak bakmasına yol açmıştır. Bir Multi CDN stratejisi ile sizin içerikler birden fazla CDN firması tarafından sunulmakta olup size daha iyi bir performans daha fazla yedeklilik ve çoğu zaman daha iyi bir fiyatı garanti edebilir.

2.5 İÇERİKLERİN İÇERİK DAĞITIM AĞINA KONUMLANDIRILMASI

İçerik dağıtım ağları, içerik sağlayıcıların ihtiyacına göre iki farklı içerik konumlandırma yöntemi: İletmek (Push CDN) ve Çekmek (Origin Pull CDN) sunuyorlar. Bu yöntemler içeriklerin nasıl içerik sağlayıcıların kaynak sunucularından, Kopya sunucularına (Replica Servers) aktarılmasında kullanılır. CDN firmaları kendi altyapılarına göre bu iki yöntemden birisi veya her ikisini müşterileri için sunarlar.

2.5.1 Kaynaktan Çekme Yöntemi

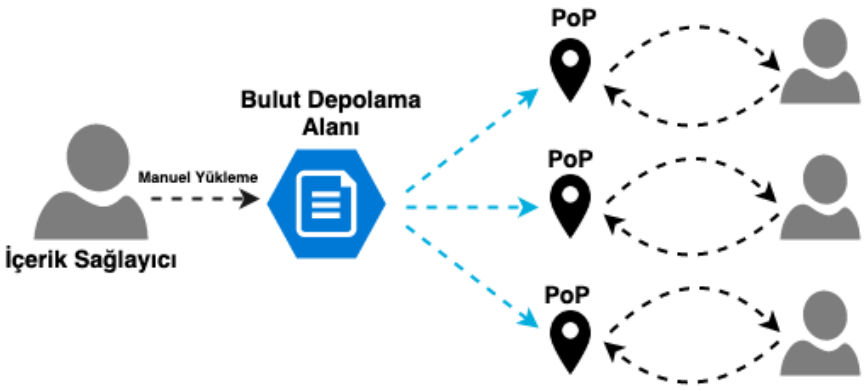
Kaynaktan Çekme Yöntemini (Origin Pull CDN) kullanmakta olan CDN müşterisi, yani içerik üreticisi, CDN hizmetine başlamak, herhangi bir işlem yapmasına gerek yoktur. Yapmaları gereken URL'leri CDN firmasını gösterecek şekilde tekrardan düzenlemektir. Bir kullanıcı belirli bir içerik istediğinde, o içerik herhangi bir Kopya Sunucusunda bulunmadığı zaman, CDN sunucuları ilk önce o içeriğin Kaynak Sunucusuna istek göndererek, o içeriği çeker (alır) ve daha sonra kullanıcıya sunar. Ardından, çoğaltma sunucusunun "En az Kullanım" (min-use) ayarına bağlı olarak, CDN bu içeriği önbelleğe alabilir. Bu yöntemde, içeriğe yönelik ilk istekler her zaman kaynak sunucusu üzerinden gerçekleştirilir, ancak içerik önbellek sunucusunda mevcut olduğu durumda sonrasında gelen istekler önbellek sunucusundan sunulur. Bu yöntem, resimler gibi küçük statik nesneler için idealdir (CDN net, n.d.) (Max CDN, n.d.) (MaxCDN FAQ, n.d.).



Şekil 2.2: Origin Pull CDN.

2.5.2 Kopya Sunucusuna İletme Yöntemi

Genel olarak, bir Kopya Sunucusuna İletme Yöntemini kullanan CDN (Push CDN), bir bulut depolama alanı veya içeriğin kullanıcılara yakın konumlarda depolandığı ikinci bir sunucu gibidir. Bu yöntemde, içerik sahipleri talep edilmeden önce içerikleri Kopya sunucularına iletmeli veya yüklemelidir. Bu İletme işlemi manuel veya otomatik olarak yapılabilir. Push yöntemi sağlayan CDN firmaların müşterileri FTP ile içeriklerini Kopya sunucularına veya depolama alanlarına iletebilirler. Bu yöntem büyük içerikler için idealdir (MaxCDN FAQ, n.d.) (Max CDN, n.d.).



Şekil 2.3: Push CDN.

2.6 VEKALETEN ÖNBELLEĞE ALMAK

İçerik Dağıtım Ağları bir içeriği ilk elde etmek için kaynak sunucusuna istek yapar. Bu görev aslında kopya sunucusu tarafından yapılır ve burada kopya sunucusu proxy'lik yaparak içeriği kaynak sunucudan alır. İlk istekten sonra gelen istekler için kopya sunucusu önbellekten cevap verir (Cache). Vekaleten önbelleğe almak (Proxy Caching) CDN'deki bir sunucuya içerik üreticisi ve son kullanıcı arasında aracı rolünü oynamaya izin verir.

2.6.1 Vekil Sunucular

Vekaleten önbelleğe almak konusunu anlamak için, önce bilgisayar ağlarında vekil sunucusu (Proxy Server) tanımını açıklayacağız. Proxy sunucusu, son kullanıcı ile diğer sunucular

arasında veya iki sunucu arasında oturan bir sunucudur. Proxy sunucuları aracı görevi görür. Son kullanıcı – burada başka bir sunucu da olabilir- proxy sunucusuna bağlanır ve aradığı içeriği talep eder. İstenilen içerik bir servis, dosya, bağlantı, web sayfası veya diğer sunucularda bulunan diğer kaynaklar olabilir. Daha sonra proxy sunucusu, isteği yapan son kullanıcı yerine vekaleten başka bir sunucudan o içeriği ister, içeriği alır ve son kullanıcıya iletir. Proxy sunucuları, dağıtılmış sistemlere kapsülleme ve yapılanma ekler (Wikipedia - Proxy server , n.d.).

Farklı proxy sunucusu türleri mevcuttur ama biz ilerleyen bölümlerde proxy sunucuların iki önemli türüne odaklanacağız; ileri vekil sunucu (Forward Proxy Server) ve ters vekil sunucu (Reverse Proxy Server).

2.6.2 İleri Vekil Sunucuları

İleri vekil sunucuları (Forward Proxy Servers), ağın üçlarında (Edge) son kullanıcılarla diğer sunucular arasında bulunur ve işleri önceden belirlenmiş kurallara göre giden trafiği düzenlemektir. Buna ek olarak, Reverse Proxy sunucuları bir müşterinin IP adresini saklayabilir. Ayrıca, önbelleğe alınmış verileri sunarak çıkan trafik hacmini de azaltır. Reverse Proxy sunucuları istenmeyen sitelere kullanıcı erişimini kısıtlamak için de kullanılabilir (Wikipedia - Proxy server , n.d.).

2.6.3 Ters Vekil Sunucuları

Ters vekil sunucuları (Reverse Proxy Servers), ileri vekil sunucularının tam tersi yönde çalişir. Ters vekil sunucuları son kullanıcı ve kaynak sunucuları (Origin Servers) veya diğer sunucular arasında bulunur, son kullanıcılardan istek alır, bu istekleri kaynak sunucusuna iletir ve yanıtı kaynak sunucusundan aldıklarında hemen sonra doğrudan cevabı son kullanıcıya iletirler. Ters vekil sunucuları ayrıca yük dengeleme (Load Balancing) ve yüksek kullanılabilirlik ve ya ulaşılabilirlik elde (High Availability)etmek için kullanılırlar (What is Reverse Proxy Server, n.d.).

2.6.4 İçerik Dağıtım Ağlarında Ters Vekil Sunucular

Ters vekil sunucuları, gelen ve giden trafiği ele almak ve idare etmek için içerik dağıtım ağlarına kullanılırlar. Ters vekil sunucuları, coğrafi olarak dağınık konumlarda bulunur ve bu

yerlerde, önceden talep edilen ve sonrasında aynı ters vekil sunucular tarafından sunulan içeriğin kopyaları veya başka bir deyişle, önbelleğe alınmış hallerini barındırırlar. Bu eylem, Vekaleten Önbelleğe Alma (Proxy Caching) olarak adlandırılır ve vekil sunucusu üzerinde Ters vekil sunucusu tarafından istenen içeriği depolar.

Bu eylemi yaparak, Ters Vekil Sunucuları, CDN'lerin önbelleğe alınmış kaynakları daha fazla kullanıcıya hizmet vermek için kullanmalarını sağlar. Bu ters vekil sunucuları, CDN'ler için bir ağ geçidi (Gateway) görevi görür.

Bir kullanıcı bir kaynak istediğinde, vekil sunucuları ilk önce istenen içeriğin önbellekte olup olmadığını kontrol eder, önbelleğe alında halde derhal son kullanıcıya sunar. Önbellekte bulunmadığı takdirde, içeriği kaynak sunucusundan alır, aynı anda önbelleğe yazıp kullanıcıya iletir (What is Proxy Caching? , 2016) (What is Reverse Proxy Server, n.d.).

2.7 YÜK DAĞITIMI

Yük dağıtma veya yük dengeleme (Load Balancing) terimi çoğunlukla bilgisayar ağlarında kullanılır ve iş yükünü kaynak kullanımı ve optimizasyonu için birden fazla cihaz veya cihaz kümesi - yani sunucu çiftlikleri veya sunucu havuzu - ağ bağlantıları , Merkezi İşlem Birimleri veya ağ bant genişliği arasında dağıtma yöntemidir. Verimi en üst düzeye çıkarmak, yanıt süresini ve maliyetini en aza indirmek ve farklı faktörlerin neden olduğu genel giderlerden kaçınmak için kullanılır (NGINX - What Is Load Balancing? , n.d.) (Cereghetti, November 9th, 2012) .

Modern yüksek trafikli web uygulamaları ve İçerik Dağıtım Ağları, müşterilerden milyonlarca eşzamanlı istekle ilgilenir ve hizmet düzeyi anlaşmalarına dayanarak, bu bileşenleri hızlı ve güvenilir bir şekilde müşterilere ve son kullanıcılara sunmaları gerekir. Bu gereksinimlerle başa çıkmak için CDN'ler daha fazla sunucu eklemek zorundadır. CDN'ler tarafından kullanılan bu teknik onların erişilebilirlik durumunu yüksek seviyelerde tutur.

Bir CDN'de bir yük dağıtıcı (Load Balancer) genellikle kopya sunucularının (Replica Servers) önüne oturur ve hızı en üst düzeye çıkarmak, gecikmeyi azaltmak ve kapasiteyi kullanmak için istemcilerden gelen tüm istekleri kullanılabilir kaynaklara dağıtır.

Yük dengeleyicinin arkasındaki sunucu havuzundan bir sunucu çevrimdışı olduğunda, yük dengeleyici otomatik olarak bu sunucuya trafik göndermeyi durdurur ve trafiği kalan sunuculara yönlendirir, sunucu kümesine yeni bir sunucu eklenmişse de yük dengeleyici bu sunucuya sunucu havuzuna ekleyerek ona istek gönderecektir.

Yük dengeleyiciler, içerik dağıtım ağlarında tek bir başarısızlık noktasını (Single Point of Failure) azaltmak için de önemlidir (NGINX - What Is Load Balancing? , n.d.) (DigitalOcean - What is Load Balancing? , 2017). Yük Dengeleyiciler HTTP, HTTPS, TCP ve UDP trafik işlemlerini gerçekleştirebilir.

2.7.1 Yük Dağıtım Türleri

İki tür yük dengeleme vardır: Donanımsal Yük Dağıtma ve Yazılımsal Yük Dağıtma.

Donanımsal yük dengeleme, donanım satıcısı tarafından sağlanan makineye yüklenen özel bir yazılımla olur, bu donanım çoğunlukla özel işlemciler kullanır. Bu yöntemi kullandığınızda trafiğiniz arttığı durumunda, makinenizi yükseltmeli veya yeni makineler eklemelisiniz.

Yazılımsal yük dengelemesinde kendi ticari donanımınızın üzerinde seçtiğiniz yazılımı çalıştırabilirsiniz. Bu seçenek, yazılım yük dengelemesini daha esnek ve daha ucuz hale getirir.

2.7.2 Sağlık Kontrolü ve Yük Dengeleme Algoritmaları

Kendi yük dengeleme algoritmanızı ilerleyen bölümde açıklayacağımız yük dengeleme algoritmalarından gereksinimlerinize göre seçebilirsiniz. Bu algoritmaların amacı mevcut sağlıklı sunuculardan seçim yapmaktır. Sağlıklı sunucu, dinleyebilen ve yanıtlayabilen bir sunucuyu ifade eder, sağlık kontrolleri düzenli olarak yapılmaktadır. Bu süreçte yük dengeleyici, tanımlanmış protokol ve port üzerinden sunuculara (Replica server) bağlanmaya çalışır. Bu aşamada bir sunucu bu sağlık kontrolünü geçemezse, bir sonraki sağlık kontrolüne kadar sunucu havuzundan (sunucu grubu) kaldırılır.

Popüler yük dengeleme algoritmalarından birkaçı:

Round Robin: Bu algoritmada, son kullanıcılardan (veya diğer sunuculardan) gelen tüm talepler sırayla sunuculara dağıtılır. Yük dengeleyici, ilk istek için sağlıklı sunucular

listesindeki ilk sunucuyu seçer, ardından listenin sonuna kadar sırayla hareket eder ve listenin sonuna ulaştığında, en baştan başlar.

Least Connections: Bu yöntemde en az etkin bağlantıya sahip sunucu seçilir ve yeni istek bu sunucuya gönderilir. Bu algoritma, mevcut trafik daha uzun oturumlara neden olduğunda tercih edilir.

IP Hash: Her müşterinin IP adresi istekleri alacak olan sunucuyla eşleştirilir. Bu yöntem belirli IP'leri belirli sunuculara tahsis etmek gibidir. Bu algoritmada aynı istemcinin her zaman aynı sunucuya bağlanacağı garanti edilir.

2.8 DNS

Alanı Adı Sistemi (DNS), internete bağlı herhangi bir hizmet veya kaynak için kullanılan hiyerarşik bir adlandırma sistemidir. En temel seviyelerdeki DNS'ler, addan adrese bir eşleme sağlayan dağıtılmış veritabanlarıdır. Bu servis İnternet telefon rehberi gibi düşünülebilir. Bir DNS sorgusunun geçmesi gereken farklı donanım bileşenleri hakkında bilgi edinmek, DNS çözümlemesinin arkasındaki işlemi tam olarak anlamak için önemlidir.

Bir web sayfasını yüklerken veya bir URL açarken 4 DNS sunucusu var: DNS Recursor,Root Nameserver, TLD Nameserver, Authoritative Nameserver (Cloudflare - What is DNS ? , n.d.).

2.8.1 Aramak ve Yönlendirmek

Bir istemci (kullanıcı) webden bir URL istediğinde ilk adım, sunucu adını çözmek ve bu URL'nin sunucusuna karşılık gelen IP adresini döndürmektir. Bu işlem arama Alanı Adı Sistemi (DNS) tarafından yapılır. Bu aşamada yerel DNS sunucusuna bir arama gönderilir. Eğer yerel DNS sunucusunda bu URL ile eşleşen bir adres mevcut değilse, Yetkili DNS Sunucusuna (Authoritative DNS Server) gönderilir.

Bir CDN'de DNS sunucuları farklı yerlerde bulunduğundan, CDN'lerin ilgilenmesi gereken en önemli sorunlardan biri, CDN içindeki bir isteğin ilgili sunucuya nasıl yönlendirileceğidir.

Daha önce bahsedilen sorunu CDN’lerde çözmek için en popüler iki yöntem DNS Yeniden Yönlendirme (DNS Redirection) ve URL Yeniden Yazmadır (URL Rewriting) (Cloudflare - What is DNS ? , n.d.).

2.8.2 DNS'in CDN'lerdeki rolü

DNS yönlendirmesi (DNS Redirection) çoğunlukla CDN'lerde kullanılır. Bu yöntemde Yetkili Ad Sunucusu (Authoritative Nameserver) CDN tarafından kontrol edilir. Bu adı vermenin nedeni, DNS'i çözmek için Yetkili Ad Sunucusuna (Authoritative Nameserver) bir istek gönderildiğinde, istekleri CDN'nin kopya sunucularından birine yönlendirmesidir. Bu çözüm, ağ durumu ve kaynakların kullanılabilirliği gibi bazı faktöre bağlıdır (B. Molina, 2004).

2.8.3 DNS Yük Dengelemesi

DNS Yük Dengeleme (DNS Load Balancing), yükü farklı kopya sunucuları arasında dağıtmak için çoğu CDN tarafından yapılan bir işlemdir. DNS Yük Dengeleme'de, son kullanıcı ve ya istemci isteklerini tek bir makineye yönlendirmek yerine, istemcilerden gelen istekler bir grup sunucu makinesine dağıtılır. CDN şirketlerinin bunu yapmasının nedeni ölçeklenebilirlik ve yüksek kullanılabilirlik elde etmektir (NGINX - What Is DNS Load Balancing? , n.d.).

2.8.4 Anycast DNS

Daha önce de belirtildiği gibi, DNS çözülmesi (DNS Resolution) CDN'ler için çok önemlidir. CDN şirketleri, DNS Çözülmesi süresini azaltmak için Anycast DNS adlı bir yöntem kullanır. Anycast'ta, birçok DNS sunucusuna tek bir IP atanır. Bu sunuculardan herhangi biri istemciler tarafından gönderilen DNS sorgularına cevap verebilir ama temel olarak, müşteriye coğrafi olarak en yakın olan sunucu taleplere cevap verir. Bu işlem gecikmeyi azaltığ, DNS çözümleme hizmetleri için çalışma süresini de iyileştirir. Ayrıca CDN firmaları bu yöntemi kullanarak müşterilerine DNS Flood DDoS saldırı koruması sağlarlar (Cloudflare - What is Anycast DNS? | How Anycast Works With DNS , n.d.).

2.8.5 İstek Yönlendirme

CDN'lerde istek yönlendirme (Request Routing), HTTP ve DNS yeniden yönlendirme (DNS Redirection) mekanizmalarının bir kombinasyonudur. Daha önce de belirtildiği gibi, CDN'de bulunan bir içerik için bir istek alındığında, Anycast DNS aracılığıyla en yakın Replica sunucusu seçilir. Kopya sunucusu seçimi ve isteği yönlendirme sırasında içerik kullanılabilirliği, sunucunun yükü ve coğrafi yakınlığı gibi çeşitli faktörler göz önünde

bulundurulur. Seçilen sunucu yük altındaysa veya istemci ile bu kopya sunucusu arasındaki ağ yolunda bir sorun varsa, istek alternatif bir kopya sunucusuna yönlendirilir.

3. MALZEME VE YÖNTEM

3.1 PROBLEM TANIMI

İçerik Dağıtım Ağları (CDN'ler) dağıtık yapıları sayesinde kaynak (Origin) içerikleri kullanıcılara hızlı bir şekilde ulaştırmaktadırlar. Bu sistemde kullanıcıların istekte bulundukları içerikler, Origin sunucusundan kullanıcılara en yakın varlık noktasına taşınarak önbelleğe alınmaktadır. Bundan sonra o bölgeden aynı içeriğe gelen istekler, içerik hazır olduğundan çok hızlı bir şekilde cevaplanabilmektedir.

Dağıtık yapıyla kullanıcılara yakınlaşabilseler de CDN hizmeti performansında düşüşler yaşamaktalar. Bunun sebebi Origin'ler ile İçerik Dağıtım Ağları arasındaki ağların birçok ISP tarafından yönetilmesidir. ISP'ler aradaki ağlar üzerinde dinamik olarak yönlendirmeler yapmaktadır. Bu da içeriklerin Origin'den CDN platformuna ulaşma sürelerinde büyük dalgalanmalara neden olmaktadır. Bu yüzden kullanıcılara içerikleri hızlı ulaştırılması zorlaşmaktadır. Süre kısıtlarına uyamamalarına neden olmaktadır. Bu çalışmanın başlatma sebeplerinden en önemlisi bu sorundur.

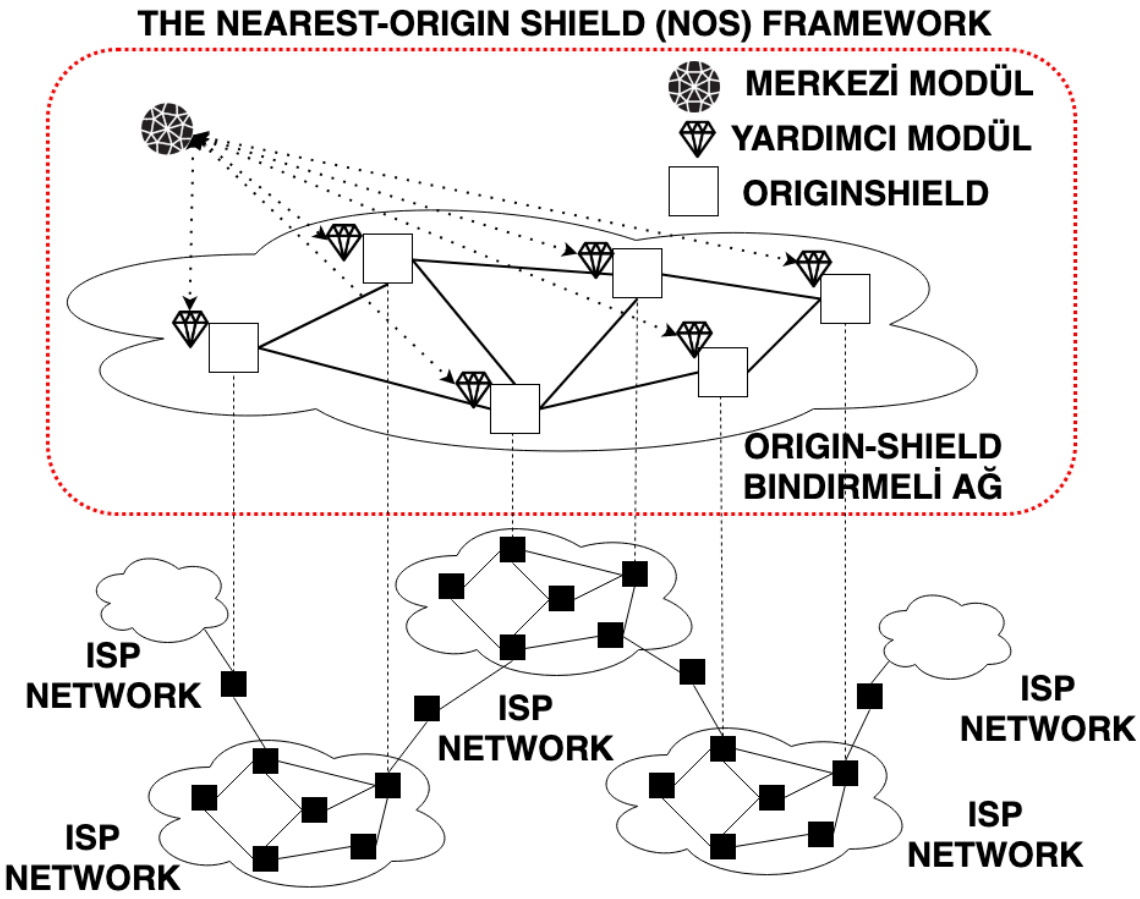
Hızlı içerik servisiyle global ölçekte başarılı olabilmek için performansı ISP'lerin yaptıkları değişikliklerden olabildiğince yalıtmamız gerekmektedir. Amacımız içerik servisinin performansını ISP'lerin yaptıkları değişikliklerden olabildiğince yalıtmaktır. Bu, içeriklerin Origin'lerden CDN sunucularına her zaman en kısa yol üzerinden gelmeleriyle olacaktır. Çalışmamızda ISP'lerin kontrol ettiği ağlar üzerinde bileşenlerimizin oluşturduğu bir bindirmeli ağ (Overlay Network) kurduk. Bu overlay alttaki fiziksel ağdaki değişikliklerden ve gecikmeleri arttıran faktörlerden etkilenmemektedir (Cloudflare - What is DNS ? , tarih yok).

Alttaki fiziksel ağdaki değişikliklere göre overlay'i yönetecek bir Ağ Yönetim Sistemi geliştirdik. Ağ Yönetim Sistemi Merkezi Modül ve Destek Modüllerinden oluşmaktadır. Overlay ağı, Merkezi Modül (Central Module) ve Destek Modülü 'ne (Helper Module) ait sistem modelleri bir sonraki bolümde açıklanacaktır.

3.2 ÖNERİLEN GENEL SİSTEM MİMARİSİ

Bu çalışmada önerdiğimiz En Yakın Kaynak Sunucu-Kalkan (Nearest Origin-Shield NOS) yapısı bir bindirmeli ağ (Overlay Network) oluşturarak bu sorunun üstesinden gelmektedir.

NOS mevcut fiziksel ISP ağlarının üzerinde bir overlay oluşturur. Bu overlay CDN de olan Sunucu-Kalkanlardan (OriginShield Server) oluşmaktadır. Önerdiğimiz NOS yapısı Merkezi ve Yardımcı Modüllerden oluşmaktadır. Bu yapı şekil 3.1'de gösterilmiştir. Bu yapı sayesinde OriginShield' lerden kaynak sunucularına giden en hızlı rotalar dinamik olarak belirlenir.

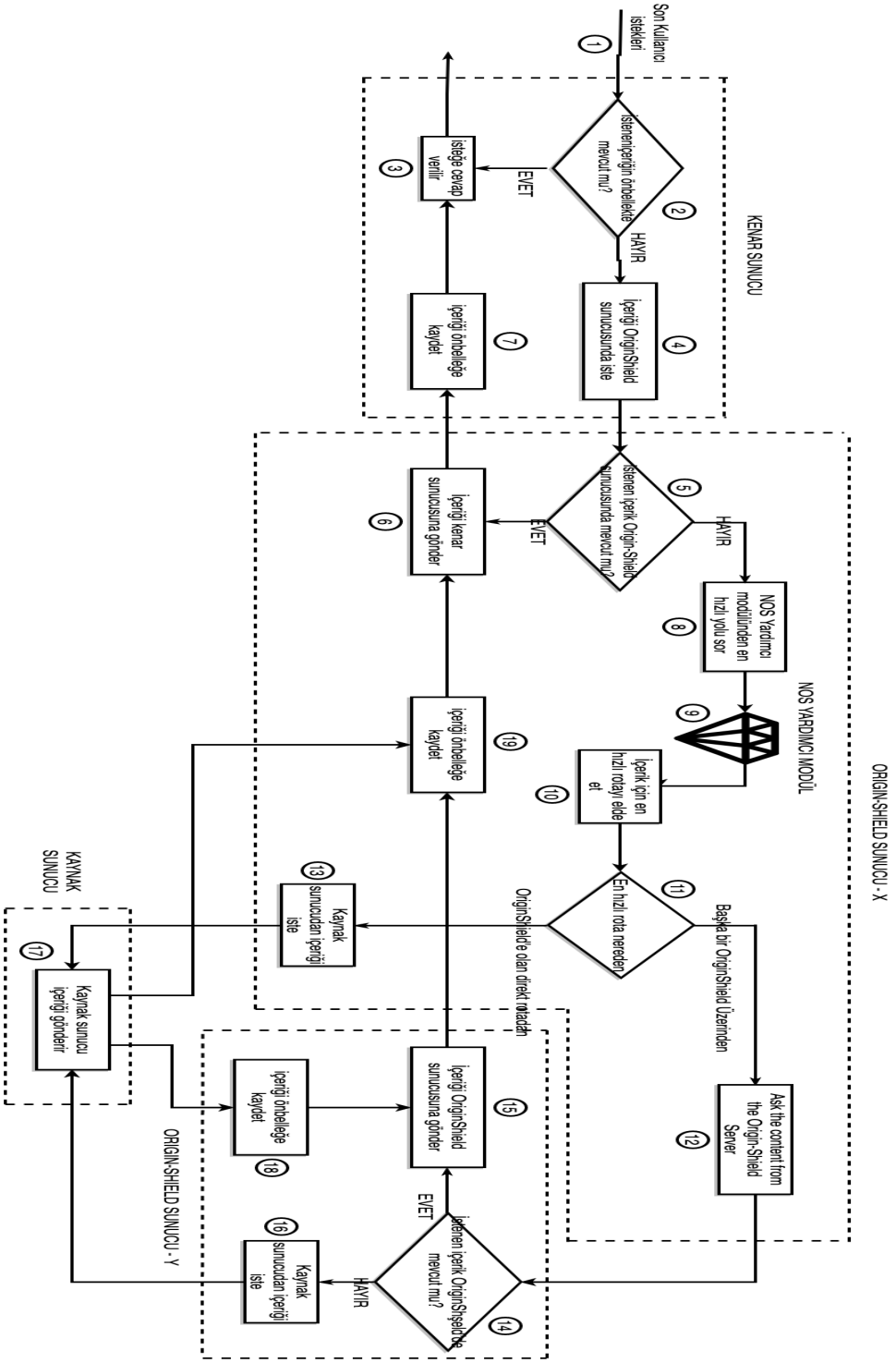


Şekil 3.1: Önerilen Bindirmeli Ağ Mimarisi.

Her Origin-Shield üzerinde bir Yardımcı Modül yerleştirilmiştir. Bu yardımcı modül periyodik olarak uçtan uca rota yolu gecikmelerini (end-to-end routing path delays) ve OriginShield'in load'ını (CPU, Network,Disk I/O vb.) izler. Yardımcı Modüller ölçümlerini Merkezi Molüle gönderir.

Merkezi modül en hızlı yönlendirme rotasını her Origin- OriginSheield çifti için hesaplar ve bu yönlendirme bilgilerini Yardımcı Modüllere gönderir. Bu nedenle, Origin-Shield sunucuları,

Origins'e giden en az gecikmeli ve en az dalgalı (Jitter Free) yolu bilir. NOS tabanlı CDN'lerimizin çalışması Şekil 3.2'de gösterilmiştir.



Şekil 3.2: NOS tabanlı CDN'in çalışması.

(1) Son kullanıcının içerik istekleri önce bir Kendar (Edge) Sunucusuna gelir.

(2) İstenen içeriğin önbellekte olup olmadığı kontrol edilir

(3) Ve eğer o içerik önbellekte varsa, kullanıcıya geri gönderir.

(4) Bu içerik Edge sunucusunun önbelleğinde mevcut değilse, içeriği ilgili Origin-Shield Sunucusundan ister. Bu şekildeki OriginShield Server -X'tür.

(5) OriginShield - X bu sefer önbelleğini kontrol eder.

(6) Eğer o içerik varsa, içeriği Edge Server'a geri gönderir.

(7) Kenar sunucu (Edge Server) daha sonra içeriği önbelleğe alır ve (3) son kullanıcıya gönderir.

Eğer OriginShield -X bu içeriğe sahip değilse

(8) NOS Yardımcı Modülünden, kaynak sunucusuna giden en hızlı yolu sorar. (10) NOS yardımcı modülü OriginShield'den Origin'e en hızlı rotayı sağlar. Bu rota OriginShield'den Origin'e doğrudan bir rota veya bir başka OriginShield aracılığıyla oluşan bir rota olabilir. Bir sonraki bölümlerde ayrıntılı olarak açıklayacağımız bu rota NOS Merkezi Modül tarafından belirlenir ve periyodik olarak NOS Yardımcı Modüllerine gönderilir. Eğer en hızlı rota aracı bir OriginShield'den geçiyorsa; şekilde OriginShield -Y gösterilmiştir.

(12) Daha sonra Origin-Shield - X, Origin-Shield-Y 'deki içeriği sorar. Eğer en hızlı rota, Origin'den OriginShield'e doğrudan bir rotaysa,

(13) O zaman doğrudan Origin'den o içeriği ister.

(14) Origin-Shield - Y, OriginShield-X'ten gelen içerik isteğini aldığında önbelleğini kontrol eder ve

(15) Eğer o içeriğe sahipse, içeriği Origin-Shield – X'e geri gönderir. Aksi takdirde

(16) İçerik doğrudan kaynak sunucuya sorulur. Kaynak sunucu (Origin) içeriği OriginShield -Y'ye gönderir (eüer isteği yapan OriginShield -Y ise)

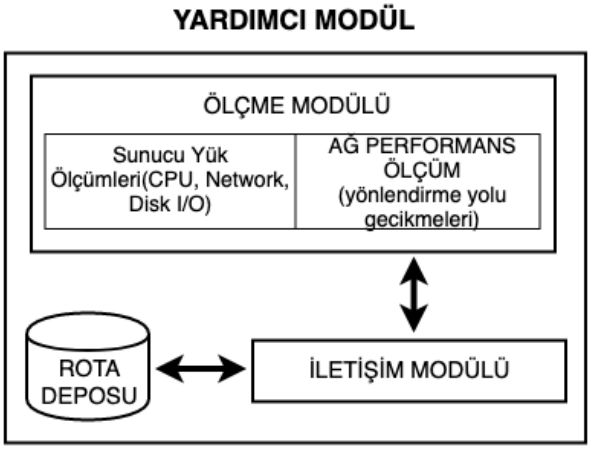
(18) OriginShield -Y gelen içeriği önbelleğine alır ve (15) OriginShield -X 'e gönderir.

(19) OriginShield -X de içeriği önbelleğine alır ve (6) isteği yapan ilgili Edge Server (kenar sunucuya) gönderir.

Yardımcı Modül, Merkezi Modül ve alt modülleri bir sonraki bolümde açıklanacaktır.

3.2.1 Yardımcı Modül

NOS Yardımcı Modülleri, tüm OriginShield sunucularında konumlanmıştır. Yardımcı Modülün sistem modeli Şekil 3.3'te gösterilmiştir. Yardımcı Modül, Ölçme alt modülü aracılığıyla uçtan uca yönlendirme yolu gecikmelerini ve sunucu yükünü (yani CPU, ağ ve disk G / Ç yükleri) ölçer. Bu ölçümler NOS yapısı için hayati öneme sahiptir. Yardımcı Modüller ölçümlerini, İletişim alt modülü aracılığıyla, periyodik olarak Merkezi Modüle gönderir. Dolayısıyla Merkezi Modül CDN ağı ve her OriginShield sunucusunun durumu hakkında genel bir görüşe sahip olabilir. Son olarak, iletişim alt modülü vasıtasıyla, Yardımcı Modüller, Merkez Modül tarafından belirlenmiş olan, Origin'lere en hızlı ve dalgasız (Jitter-Free) rotaları elde edinir. Bu rotalar Rota Deposu alt modülünde (Routes Store sub-module) tutulur.

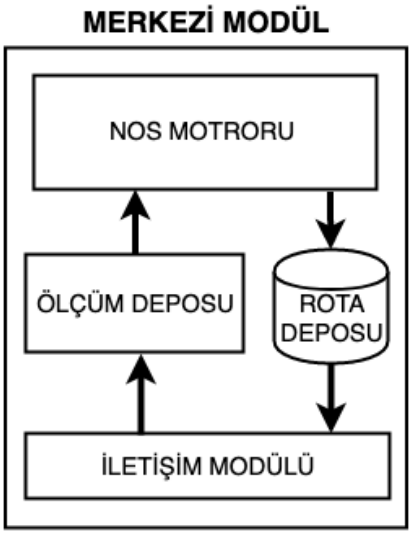


Şekil 3.3: Yardımcı Modül.

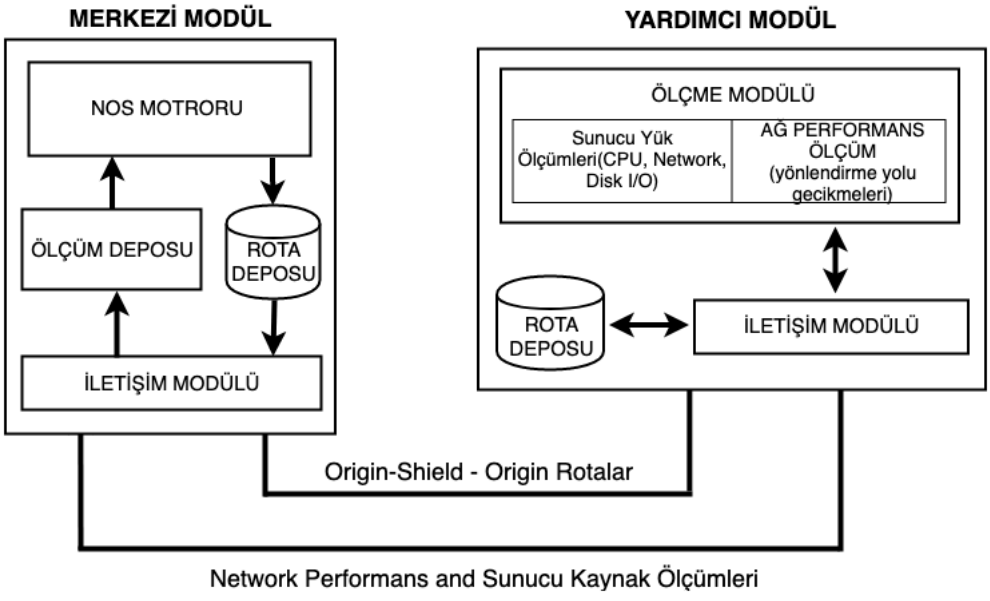
3.2.2 Merkezi Modül

NOS Merkezi Modülün sistem modeli Şekil 3.4'te gösterilmiştir. Merkezi Modül İletişim (Communication) alt modülünü kullanarak periyodik olarak Yardımcı Modüllerden ölçümler almaktadır. Bu ölçümler daha sonra Ölçüm Deposunda (Measurments Store) toplanır. NOS Motor alt modülü (NOS Engine Sub-module) bu ölçümleri alır ve her OriginShield'den Origin'e giden en hızlı ve en dalgasız rotaları hesaplar. Merkezi ve yardımcı modülün beraber dizilişi şekil 3.5'te gösterilmektedir.

NOS motorunun, Origin sunucularının yanı sıra, OriginShield sunucularının güncel bir listesine sahip olduğunu varsayıyoruz.



Şekil 3.4: Merkezi Modül



Şekil 3.5.: Merkezi ve Yardımcı Modülleri

3.3 ÖNERİLEN ALGORİTMA

Önerdiğimiz sistemde ve bir önceki bolümde bahsettiğimiz modüllerde iki farklı algoritma kullandık. Bu algoritmalar sıra ile Yardımcı Modül ve Merkezi Modülde kullanıldı. Yardımcı modülde kullanılan algoritmanın amacı NOS Engine veya merkezi modüldeki karar mekanizmasını değerli ve güncel bilgiler ile beslemek ve ona gereken bilgileri sağlamaktır. Yardımcı modülündeki algoritmanın amacı ise aşağıdaki ISP'lerin değişikliklerini minimum hale getirip aynı zamanda dalgalanmayı olabildiği kadar aradan kaldırmaktır. Aşağıda kullanılan algoritmalar detaylı bir şekilde açıklanacaktır.

3.3.1 NOS Engine

Merkezi Modülde hesaplamaların yapıldığı yer NOS Motoru adlandırdığımız bir alt modüldür. Tüm hesaplamalar aslında bu alt modülde yani NOS Motorunda yapılmaktadır. Bu hesaplamaların amacı en hızlı ve en dalgasız rota ile bir bindirmeli ağ kurup trafiği oradan yönlendirmektir. Bu amaca varmak için Gecikme Duyarlı ve Dalgasız (Delay aware and Jitter-Free) bindirmeli ağ yönlendirme algoritmasını tasarladık, bu algoritma **Alg 1** şekil 3.6'da gösterilmiştir. **Tablo 3.1**'de bu algoritmada kullanılan parametre gösterimlerini vermektedir.

Tablo 3.1: Gecikme Duyarlı ve Dalgasız Bindirmeli Ağ Yönlendirme Algoritması kullanılan kısaltmalar

Parametre	**Açıklama**
$M=\{ M_1, \ldots, M_n\}$	OriginShield'lerin listesi
$O = \{O_1, \ldots, O_k\}$	Origin'lerin Listesi
$< M_i, O_j >$	i OriginShield'i ve j Origin çifti
$D(M_i\text{-}O_j)$	OriginShield i ve Origin J arasındaki uçtan uca yönlendirme yolu gecikmesi
L_{Mi}	OriginShield i için hesaplanan yuk
α, β, γ	CPU, Ağ ve Disk I/O ağırlıkları
T	Eşik
D_{ARP}	Alternatif en hızlı rotan gecikmesi
$U_{Mx}^{C}, U_{Mx}^{D}, U_{Mx}^{CN}$	OriginShield'deki disk,CPU ve Ağ kullanım yüzdesi

Algorithm 1 Delay-Aware and Jitter-Free Overlay Routing Algorithm for CDNs.

Require: List of Origin-Shields (M), list of Origins (O), Helper Module measurements of end-to-end routing path delays and node loads,

Ensure: Fastest jitter-free routes for each $< M_i, O_j >$

1: **for all** $< M_i, O_j >$ **do**
2: Calculate $D(M_i\text{-}O_j)$
3: **for all** M_x, $x \neq i$ **do**
4: $D(M_i\text{-}M_x\text{-}O_j) = D(M_i\text{-}M_x) + D(M_x\text{-}O_j)$
5: **end for**
6: $D_{ARP} = min(D(M_i\text{-}M_x\text{-}O_j))$
7: **if** $(D(M_i\text{-}O_j) - D_{ARP}) \geq T_{Delay}$ **then**
8: Calculate $L_{M_x} = \alpha \times U^C_{M_x} + \beta \times U^N_{M_x} + \gamma \times U^D_{M_x}$
9: **if** $L_{M_x} \leq T_{Load}$ **then**
10: assign $Route < M_i, O_j > = M_x$
11: **else**
12: assign $Route < M_i, O_j > = O_j$
13: **end if**
14: **end if**
15: **end for**

Şekil 3.6: Gecikme Duyarlı ve Dalgasız Bindirmeli Ağ Yönlendirme Algoritması.

Bu algoritmanın girdileri ise; OrigiShield listeleri, Origin'lerin listesi, uçtan uca yönlendirme yolu gecikmeleri (Routind Path Delay) ve Yardımcı Modüllerden elde edilen sunucu yükü ölçümleridir. Her Origin-Shield - Origin çifti için en hızlı ve dalgasız rotayı belirler.

Her OriginShield – Origin çifti için ilk önce doğrudan OriginShield'den Origin'e olan yönlendirme rotasının gecikmesini belirler (Satır 2). Sonrasında başka bir OriginShield vasıtasıyla (Üzerinden) daha hızlı bir alternatif rota olup olmadığını kontrol eder (Satır 3-7). Bu alternatif rotanın gecikmesini elde etmek için, birer birer her OriginShield'den hedef Origin'e gecikmeler hesaplanır (Satır 3-5). Ve en az gecikmeye sahip olan rota en hızlı alternatif rota (D_{ARP}) olarak ayarlanır (Satır 6).

Eğer D_{ARP} direkt olan rotadan (OriginShield'den Origin'e olan direkt rota) daha kısa bir uçtan uca yönlendirme yolu gecikmesi sağlıyorsa ve fark bir eşiğin üstünde ise (T_{Delay}) en hızlı alternatif rotada kullanılacak OriginShield'in yükü kontrol edilir (Satır 8-13). Öncelikle ölçülen CPU, ağ ve disk G / Ç yüklerine göre ve hesaplamalarda elde edilen bu yüklerin ağırlıklarına göre Origin-Shield'in toplam yük ağırlığı hesaplanır. Hesaplanan toplam yük ağırlığı bir eşiğin

altında ise (Satır 9) yönlendirmenin o OriginShield üzerinden yapılması düşünülebilir. Aksi takdirde bir önceki adımlarda elde edilen OriginShield'in aşırı yük altında olmasını engellemek için, OriginShield'den Origin'e olan doğrudan rota seçilip ayarlanır (Satır 12). Bu algoritmada gerçekçi olmayan yönlendirme yolu güncellemelerini (Routing Path Updates) engellemek için çok eşikli (Multi-Threshold) bir algoritma tercih edilmiştir.

3.3.2 Gecikme API

Bir önceki bölümde de bahsettiğimiz gibi Yardımcı modülün altında çalışan ölçme alt modülü iki bolümden uluşmaktadır. Bunların biri gecikme değerlerini ölçerken diğeri sunucuların yükünü ölçmektedir. Gecikme API'si kısaca Yardımcı modüllerin üzerinde çalışmakta olan iki alt bolümden biridir. Bu API belirlenen zaman aralığındaki gecikme bilgilerini Merkezi modül için temin etmektedir. Tasarladığımız bu API tüm OriginShield'lerin üzerinde olan Yardımcı Modüller üzerinde, bir alt modül olup, bir proses gibi çalışmaktadır. Bu API OriginShield'ler ile tüm Origin sunucular arasındaki gecikmeyi ölçmenin yanı sıra OriginShield'ler arasındaki gecikmeyi de elde etmektedir. ICMP paketleri TCP/UDP paketlerinden farklı yönlendirildikleri için ve ICMP paketleri daha yoğun trafiğe sahip olan rotalardan yönlendirildiği için ICMP ping yöntemini kullanmak gecikmeyi elde etmek çok doğru ve nitelikli bir yöntem olmadığı için bizim kullandığımız yöntem Wireshark uygulamasının sunduğu yönteme benzemekte ve ICMP paket trafiği yerine TCP Paket trafiği ve gecikmesini ölçmeye çalıştık.

Bu komut Linux işletim sisteminde:

```
traceroute -T domainname.tld
```

Şeklinde çalıştırılmaktadır.

Geliştirdiğimiz API bazı komutları aşağıda açıklanmaktadır:

Tüm OriginShield'lerin listesini aşağıdaki URI ile elde edebiliriz:

```
http://31.3.0.28:7021/give-midcache-list
```

CDN'imizde çalışmakta olan Kaynak sunucuların (Origin) listesi için:

```
http://31.3.0.28:7021/give-origin-list
```

Kaydedilen en son işlemin numarasını (Task ID) almak için:

```
http://31.3.0.28:7021/give-current-task
```

Ayrıca bir zaman aralığındaki Task ID'leri elde etmek ve daha sonra bu ID'leri kullanarak o aralığa ait olan verileri işleme almak için de bir yöntem geliştirdik. Aşağıda belirlenen URL ile başlangıç (Start Date) ve Bitiş Tarihi (End Date) arasındaki TASK ID leri elde edebiliriz.

```
http://31.3.0.28:7021/give-taks-ids?start_date=2019-02-
11/00:00:00&end_date=2019-02-11/23:59:00
```

Yukarıda da belirlediğimiz gibi Yardımcı Modülün en önemli işlevi OriginShield'ler arasındaki gecikmeyi elde etmektir. Bir zamanın yanı bir TASK ID'ye ait olan OriginShield'ler arasındaki gecikmeyi elde etmek için:

```
http://31.3.0.28:7021/give-m2m-result/TASKID
```

Benzer şekilde bir zamanın yani bir TASK ID'ye ait olan OriginShield ile Origin sunucları arasındaki gecikmeyi elde etmek için:

```
http://31.3.0.28:7021/give-m2o-result/TASKID
```

komutları kullanılabilir.

3.3.3 Sunucu Yük API

Yardımcı modülün bir diğer alt modülü Sunucuların Yükünü ölçmek için tasarlanmıştır. Sunucuların yükü bizim için önemli bir metrik olup hesaplamalarımızda kullanılmaktadır. Bu

sebeple tüm sunucularımızın yükü yani CPU, Network, Disk I/O vb. değerleri gecikmeler ile beraber elde etmemiz gerekmektedir.

Bu verileri elde etmek için Yardımcı modülde çalışan bir API tasarladık. Bu API vasıtasıyla OriginShield'lerin bir zaman aralığındaki Load Average gibi değerlerini elde edip bu değerleri hesaplama yapmak için Merkezi modüle gönderebiliriz. Bu API çıktısı olarak satırlarda OriginShield ID'si ve sütunlarda o OriginShield'e ait olan yük değerleridir.

Bu API kullanmak için **3.3.2** bölümünde anlatıldığı gibi TASK ID ve diğer gereken bilgileri elde ettikten sonra bir zamana ait yani bir TASK ID ile o zamana ait olan OriginShield sunucularının yük verilerini elde etmek için:

```
http://31.3.0.28:7021/give-m2l-result/TASKID
```

komutu kullanılabilir.

3.4 KULLANILAN ARAÇLAR

3.4.1 Medianova

Medianova yüksek performanslı CDN ve bulut bilişim hizmeti veren bir teknoloji şirketidir. Global ölçekte en yeni teknolojileri kullanan, üreten ve bilinen bir CDN olmak istiyorlar. Bu sayede Türkiye'ye ithal ikamesi sağlamayı amaçlıyor. Dünya üzerinde dikkatle seçilmiş 40'unüzerinde varlık noktası, 17 ülkede bulunmaktadır. Medianova altyapısında en güncel teknolojiler ile müşterilerine hizmet vermektedir. HTTP/2, TCP Acceleration, Brotli ve benzeri teknolojileri agile bir şekilde altyapısına entegre etmeyi başarmıştır.

Medianova günlük 7 TBps trafiği 22 milyondan fazla özgün kullanıcıya sunmaktadır. Medianova sunucuları saniyede 6 milyon istek cevaplamaktadır. Geniş müşteri profili sayesinde Türkiye'nin çoğu büyük e-ticaret, Telekom ve oyun firmalarının statik ve dinamik içeriklerini dünyanın farklı noktalarından erişim hızını arttırmaktadır. Günlük 8 milyar imaj, 9 milyon şarkı 11 milyon video dosyası Medianova sunucularından farklı müşterilerin kullanıcılarına sunulmaktadır.

3.4.2 NGINX

NGINX, Medianova'nın yanı sıra birçok CDN firması tarafından tercih edilen bir web sunucusudur. Ters proxy (Reverse Proxy), yük dengeleyici (Load Balancer), e-posta proxy (Mail Proxy) ve HTTP sunucusu (HTTP Server) olarak kullanılabilen açık kaynaklı, yüksek performanslı bir web sunucusudur. Güvenilirliği, yüksek performansı ve zengin özellikleri nedeniyle birçok sektörde firmalar tarafından tercih edilen sunucu haline gelmiştir. NGINX'in NGINX ve NGINX Plus olmak üzere iki sürümü mevcuttur. NGINX Plus, kullanıcılar için ekstra özelliklere sahip olan NGINX'in ücretli bir versiyondur.

3.4.3 SSH

Secure Shell bir kriptografik ağ protokolüdür. Bu protokol güvenli olmayan bir ağ üzerinde güvenli bir şekilde ağ hizmetlerini çalıştırmak için kullanılır. SSH İstemci-Sunucu (Client-Server) mimarisine sahiptir. SSH bir SSH Kullanıcını bir diğer SSH sunucusuna güvenli olmayan bir ağ üzerinde, güvenli bir kanal vasıtasıyla birbirine bağlar. SSH genellikle Unix benzeri işletim sistemlerine bağlanmak için kullanılır ama Microsoft Windows için de kullanılabilir. Bizim SSH kullanarak uzakta olan başka bir makineye giriş yapmak ve o makine üzerinde komutlar çalıştırmak için kullandık. SSH bağlantısı kurabilmek için bir SSH istemci (Client) programı gerekmektedir. Bu program çoğu macOS ve Linux işletim sistemleri üzerinde mevcuttur. Windows işletim sisteminde PuTTy veya benzeri program SSH için kullanılabilir.

3.4.4 Apache JMeter

Apache JMeter ™, yük testi ve performansı ölçmek için saf java'da geliştirilen açık kaynak bir uygulamadır. İlk amacı olarak web uygulamalarını test etmek için geliştirilse de, kabiliyetleri daha sonra başka amaçlar için daha da genişletildi. Bu uygulama kullanılarak, web uygulamasının performansı veya diğer metrikleri farklı koşullar altında test edilebilir. Örneğin, bu uygulamayı farklı yük tipleri altında, toplam performans analizi için kullanabiliriz.

Apache JMeter ™ 'ı tercih etme sebeplerimiz:

- **Apache JMeter ™ açık kaynak bir uygulamadır**. Bu yüzden kullanmak için herhangi bir ücret ödememiz gerekmiyor.

- **Çok çeşitli kabiliyetleri.** Temel olarak yük testi için kullanılır, ancak stres testi veya ihtiyaç duyduğumuz simülasyon talepleri gibi diğer amaçlar için de kullanılabilir.
- **Kapsamlı dokümantasyon.** Kurulumdan test etmeye ve nihai raporlar üretmeye kadar gereken her adım hakkında bilgi edinmemize olanak sağlayan zengin bir dokümantasyona sahiptir.
- **GUI Dışı Modu.** Bu özellik bu uygulamayı seçmemizin nedenlerinden biriydi çünkü testlerimizi SSH bağlantısı üzerinden farklı sunucularda aynı komutu tek bir kere tüm sunucularda çalıştırarak yapabiliyorduk.

JMeter'in Elemanları:

JMeter'in farklı bileşenlerini JMeter Elemanları olarak adlandırıyoruz. Her eleman belirli bir amaç için tasarlanmıştır. Bazı ana elemanlar aşağıda listelenmiştir:

- **İplik Grubu (Thread Group):** İplik grupları birkaç iplik yığınından uluşur. Her iş parçacığı (Thread), sunucuya gerçek bir kullanıcı isteğini simüle eder.
- **Örnekleyiciler (Samplers)**: JMeter HTTP, HTTPS ve diğer birçok protokolü destekler. Daha önce de belirttiğimiz gibi, iplik grupları, kullanıcıların sunuculara isteklerini simüle eder. Örnekleyiciler, bu iş parçacığı gruplarının yapacağı isteklerin türünü gösterir.
- **Dinleyiciler** (**Listeners**): Test sonuçlarını göstermek için dinleyiciler kullanılır. Sonuçlar ağaç, tablo, grafik veya log dosyası biçiminde gösterilebilir.
- • **Yapılandırma (Configuration)**: buradaki özellikler örnekleyiciler tarafından kullanılan değişkenlerdir.

3.4.5 Vultr

Vultr global bulut yer sağlayıcı firmasıdır. Dünyanın 15 farklı noktasında veri merkezlerinde sunucuları vardır. Vultr Sanal Hususi Sunucu hizmeti (VPS) de sağlamaktadır. Bu hizmet sayesinde önceden belirlediğimiz 15 noktada ihtiyacınıza göre sunucu açıp üzerinde istediğiniz programı çalıştırabilirsiniz. Bu VPS'lerin CPU, RAM ve Disk boyutlarını seçebilirsiniz. Bu sanal sunucular gerçek ortama dokunmadan, benzer ortamlar yaratarak test ortamı yaratmanızı kolaylaştırır.

3.4.6 Python

Python çok amaçlı, yüksek-seviye bir programlama dilidir. Python modüler bir yapıya sahiptir ve etkili bir garbage collection sistemine sahiptir. Python kullanarak nesne yönelimli ve ya fonksiyonel programlama için kullanılabilir. Python 2.7 de yazılan kodlar için destek 2020ye kadar uzatılmasına rağmen bir tüm kodlarımız için **Python 3.X** kullanmayı tercih ettik. Python programlama dilini tercih etmemiz için sebepler arasında zengin kütüphanelere sahip olması ve direkt socket programlama imkanını sağlamasını sayabiliriz.

Kullandığımız bazı kütüphaneler:

Pandas: veri manipulasiyon ve farklı veri türlerini kullanmak için kullanılan kütüphane. Ayrıca bu kütüphane kullanılarak time-series ve sayısal tablolar üzerinde manipülasyon işlemleri kolaylıkla yapılabilir.

NumPy: büyük boyuttakı verilerin işlenmesi için kullanılan bir kütüphane. Ayrıca NumPy kullanarak çok boyutlu matrisler ve dizi veri yapıları üzerinde daha hızlı bir şekilde matematiksel işlemler yapılmaktadır. Bunlara ek olarak NumPy komplike matematiksek işlemlerin yapılması için kolay fonksiyonlar sunmaktadır.

SciPy: Python'ın başka bir bilimsel kütüphanesidir. SciPy kütüphanesi NumPy kütüphanesi ile bağlı olan bir kütüphanedir. N-Boyutlu diziler için kolay bir manipülasyon tool olarak kullanılmaktadır. SciPy kütüphanesi optimizasyon, lineer cebir entegrasyon ve benzeri işler için modüller sağlamaktadır.

Scikit-learn: Python dilinde makine öğrenmesi için kullanılan en popüler kütüphanedir. Scilit-learn içinde sınıflandırma, regresyon ve kümeleme için farklı algoritmalar barındırıyor. Bu algoritmalardan k-means, Support Vector Machine, Random Forest algoritmaları örnek verilebilir. Scikit-learn bizim kullandığımız regresyon için de kolaylık sağlıyor. Ayrıca NumPy ve SciPy ile uyumlu çalışmaktadır.

Matplotlib: Elde ettiğimiz veriler ve sonuçları görselleştirmek için Matplotlib algoritmasını kullandık. Matplotlib sayesinde 2D grafikler çıkartıp farkı ayarlar ile detaylandırabiliriz.

3.5 TEKNOLOJİNİN BİLİNEN GÜNCEL DURUMU

Gecikme ve isabet oranı, CDN'ler için en önemli performans ölçütlerinden ikisidir (S. J. Jafari, Investigating Performance Metrics for Evaluation of Content Delivery Networks , 2018) (D. Wang, 2018) (C. Huang, 2008). Gecikmeyi azaltmak için, CDN'ler dünyanın çeşitli yerlerindeki PoP'larını dikkatlice seçerler ve yönlendirme mekanizmaları (örneğin, Anycast DNS) aracılığıyla yakındaki son kullanıcılara önbellekteki içeriği sunmaya çalışırlar. Ancak, son kullanıcılar, CDN PoP'ları ve Kaynak sunucular (Origin) arasındaki yönlendirme yolları, ISP'ler ve ağ koşulları nedeniyle beklenmedik bir şekilde farklı zamanlarda değişiklikler gösterebiliyorlar. Dolayısıyla, bir ara yönlendiriciden (Router) geçen bir yol, bu yönlendiriciden geçmeden doğrudan giden bir yoldan daha kısa gecikmeye sahip olabilir. Bu konu daha önce Üçgen Eşitsizlik İhlali Problemi (Triangle Inequality Violation) adı altında literatürde gözlemlenmiştir (H. H. Zheng, 2005) (C. Lumezanu, 2009) (Y. Zhu, New York, USA). Bu problem ISP yönlendirme politikalarının neden olduğu, İnternet'in doğal bir sorun olduğu tespitine varılmıştır. Ancak, CDN'lerin bakış açısından bu konu son zamanlara kadar çalışılmamıştır ve CDN'ler için bu problemi çözmeye çalışan sadece birkaç çalışma var. Bu çalışmaların aynı olmasa da benzer problemlere odaklandığı anlaşılmıştır.

Bu bağlamda, son kullanıcılar ile CDN PoP'ler arasındaki ISP maliyetlerine ve transit yol gecikmelerine dayanan rota yollarının optimizasyonu yapan Ahmed vd. (F. Ahmed, 2016) çalışılması mevcuttur. Bu çalışmada CDN sunucuları ile son kullanıcılar arasındaki yolu optimize etmeye çalışmışlardır. Optimizasyonda ISP maliyetlerini ve performans (gecikme) değerlerini göz önüne almışlardır. Buradaki CDN platformu son kullanıcılara servis ettiği web sayfalarına JavaScript kodu gömmektedir. Bu kod da farklı yollar üzerinden CDN sunucularına istekte bulunarak sunucuların kullanıcıya hangi ISP üzerinden en kısa sürede ulaşabileceği bilgisini sağlamaktadır. Ölçümler, CDN'in ISP'lerle transit bağlantı yaparken ödemesi gereken ücretlerle birleştirilerek son kullanıcıya erişmek için optimum yol belirlenmektedir.

P.Medagliani vd. (P. Medagliani, 2017) çalışmasında video yönelimli bir CDN mimarisi için bir bindirmeli ağ yönlendirme mekanizması önerildi, burada en kısa yol bağlantı ve düğüm ağ kapasiteleri, başarılı paket aktarma olasılığı, gecikme ve dalgalanma açısından belirlendi. Bu çalışma video akışları için özelleşmiş bir CDN mimarisini düşünmektedirler. Bu mimaride ISP'lerin yaptıkları değişiklikleri göz önüne alarak en uygun yolu seçmeye çalışmaktadırlar. Seçimde birçok performans parametresi göz önüne alınmaktadır. Çalışmadaki CDN mimarisi

bizim çalıştığımız mimariden oldukça farklıdır. Yazarla önerilen sistemin gerçeklemesi hakkında yeterince bilgi vermemektedir. Önerilen yapının CDN mimarisinde tam olarak nerede çalıştığı ve gerçeklendiği konusunda herhangi bir bilgi bulunmamaktadır.

Literatürdeki bu iki çalışmaya ek olarak IEEE Next Generation Service Overlay Networks (Group, 2018) çalışma grubu geleceğin CDN mimarisi konusunda standartlar yayımlamıştır. Önerilen standarttaki mimaride son kullanıcılar, içerik üreticileri ve farklı kuruluşlar yer almaktadır. Bu bileşenler aynı anda hem içerik üreticisi hem de kullanıcı rollerine bürünebilmektedir. İçerik dağıtımı bağlamla ilişkilendirilmektedir. İçerik dağıtım ağı standartlaşmış bir overlay ağı haline gelmektedir.

Hit oranı açısından araştırmalar, içeriğin popülerliğini tahmin etme ve ona göre önbelleğe kabul veya tahliye etme politikalarına odaklanmıştır. Bu alanda diğer araştırmalar içeriğin önbellekten hangi koşullara göre silinmesine odaklanmıştır.

D. S. Berger vd. çalışmasında (D. S. Berger, 2016) büyüklüğe duyarlı bir dinamik önbelleğe kabul politikası çalışıldı. Çalışmanın ana fikri, içeriğin popülaritesinin ve popüler içeriğin boyutunun Zipf dağılımına (Huberman, 2002) uymasıdır, ve bu dağılım önbelleğe kabul ve tahliyeler yapılırken göz önünde bulundurulması gerekiyor. CDN'ler için sayfa Hit oranı, obje Hit oranı yerine (J. Wang, 2019)'te önerildi. Çalışma bir sayfanın içindeki tek tek nesnelerin Hit oranını hesaplamak yerine sayfadaki nesnelerin bir oranın üzerinde önbellekten geldiğinde Hit oranını hesaplamaktadır.

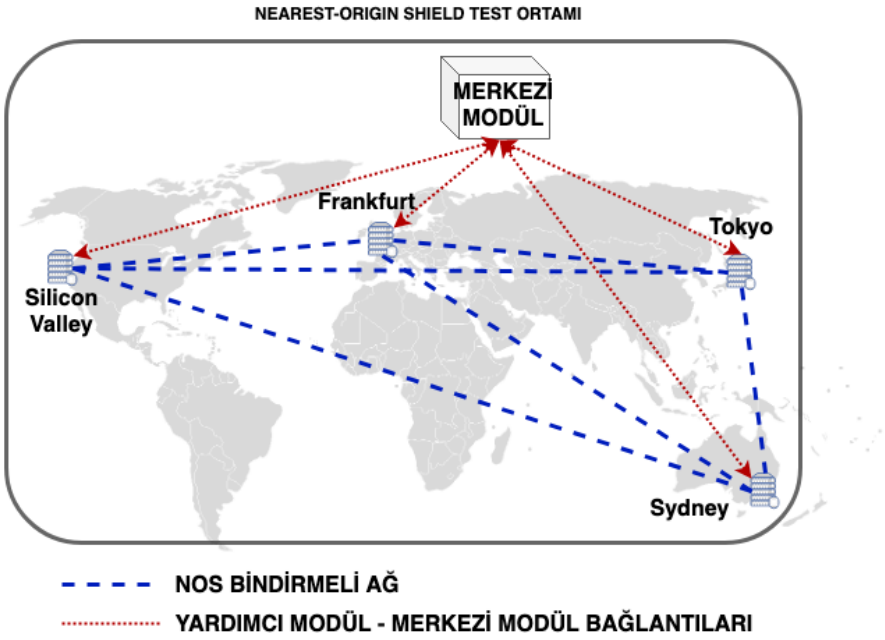
(N. B. Hassine, 2015) 'te Youtube videoların geçmişteki popülerlik değişikliklerine dayanarak video içeriğinin gelecekteki popülerliğini tahmin etmeyi amaçlayan CDN'ler için bir popülerlik tahmin mekanizması önerildi.

Analiz ettiğimiz tüm çalışmalar sorunun farklı bir yönüne odaklanmaktadırlar. (D. S. Berger, 2016) (Huberman, 2002) (J. Wang, 2019) ve (N. B. Hassine, 2015) çalışmaları sadece Hit oranına odaklanmıştır. (F. Ahmed, 2016) çalışması son kullanıcı ve CDN kopya sunucu arasındaki yönlendirme yolu gecikmesine odaklanmıştır. Ancak bu çalışmada kullandıkları mimari Vide içeriği sunan CDN yapısına özgü bir mimari olduğundan resim gibi küçük objeler için etkili olayabilir. İçerik dağıtım ağlarında resimler gibi küçük objeler çok popülerdir CDN'lerde, resimler genellikle videodan farklı olarak çok sayıda isteme sahiptir. Bahsi geçen çalışmada gerçek dünya ölçümleri yapılmamış ve performans değerlendirmeleri için veri

setlerini kullanılmıştır. Gecikmeyi en aza indirmeyi hedefleseler de, CDN'lerde yaşanan dalgalanmalar için bir çalışma yapmamışlardır.

3.6 TEST ORTAMI

NOS çerçevesini (NOS Framework) uygulanması ve değerlendirilmesi için çoğrafi olarak dört farklı bölgede bulunan (Sydney, Frankfurt, Silicon Valley ve Tokyo) dört OriginShield sunucusuyla bir test ortamı oluşturduk. Bu OriginShield'leri oluşturmak için VPS servisi sağlayan Vultr bulut hizmetini kullandık. Makineler hepsi aynı özelliklere sahipler ve farklı noktalarda konumlandırılmışlar. Bunun yanı sıra dört farklı bölgedeki Medianova müşterilerinden dört kaynak (Origin) de seçildi. Test ortamı **şekil 3.7'**de gösterilmiştir. Bu test ortamında dört OriginShield ve bileşenleri ile, yani Merkezi ve Yardımcı Modül ile, NOS bindirmeli ağını oluşturmaktadır. Açık kaynaklı web sunucusu yazılımı Nginx'in [16] 1.13.12 sürümü, varsayılan önbellekleme (Caching) ayarları ile bu OriginShield sunucularda çalışmaktadır. Test parametreleri **Tablo 3.2**'de listelenmiştir.



Şekil 3.7: Test Ortamı.

Merkezi ve Yardımcı modüllerin yazılımı ve uygulanması Python 3.x de yapıldı. Uçtan uca yönlendirme yolu gecikmelerini periyodik olarak ölçmek için her OriginShield sunucusuna bir Yardımcı Modül yerleştirdik. (Diğer OriginShield ve Origin'lere olan gecikmeleri). Buna ek olarak Yardımcı modüller sunucu yüklerini (CPU, Network, DiskI/O kullanım yüzdesi) ölçüp periyodik olarak elde edilen ölçümleri Merkezi modüle göndermektedir. Ölçüm ve raporlama süresini 5 dakikaya ayarladık.

Tablo 3.2: Test Parametreleri.

Ölçüm ve raporlama süresi	5 dakika
Gecikme Eşiği (T_{Delay})	$0.01 \times D(M_i - O_j)$
Yük Eşiği (T_{Load})	300'den 270
α, β, γ ağırlıkları	1
Proxy Caching Uygulaması	NGINX (version 1.13.12)

Merkezi Modül uygulaması ayrı bir sunucuda çalışır. Yarımcı modüllerden elde edilen veriler, merkezi modülde işleme alınır. Merkezi modül yardımcı modüllerden ölçümler alır ve her bir OriginShield - Origin çifti için en hızlı rotaları belirlemek üzere periyodik olarak Gecikme Duyarlı ve Dalgasız (Delay aware and Jitter-Free) bindirmeli ağ yönlendirme algoritmasını çalıştırır.

Bu algoritmanın uygulanması için, gecikme eşiğini (T_{Delay}) sezgisel olarak her çift için 0,01 × D (M_i - O_j) belirledik. Algorithm 1 (**Şekil 3.6**) bu ayarı sayesinde en hızlı alternatif yol % 10'dan daha fazla gecikme kazancı sağlamıyorsa bindirmeli ağdaki mevcut rota değişmeyecektir.

CPU (α), ağ (β) ve disk G / Ç (γ) kullanım ağırlıkları sezgisel olarak 1'e ayarlandı, zira test bindirmeli ağımızdaki tüm OriginShield sunucuların CPU, ağ ve disk I / O kaynağı birbirleriyle aynıdır. Yük eşiği (TLoad) 270 olarak ayarlandı, bu da CPU, ağ ve disk G / Ç kullanım yüzdelerinin toplamının 300'den 270'in altında olması gerektiğini gösterir. Bu koşullar yerine geldiğinde Algorithm 1 (**Şekil 3.6**) bu koşulları sağlayan OriginShield'i ilgili Origin – OriginShield için en iyi rota olarak ayarlar.

3.7 GERÇEKLEŞTİRİLEN TESTLER

Bu çalışmada yönlendirme yolu gecikmesini ana performans metriği olarak kabul ediyoruz. Ve daha önce de belirtildiği gibi, öncelikli hedefimiz bu gecikmeyi azaltmak ve dalgalanmayı dengelemektir. Bu beklenmedik ve bazen ani değişiklikler, çoğunlukla ISP'lerin ve ağ koşullarının dinamik değişikliklerinden kaynaklanmaktadır. Bu metriğe ek olarak Hit oranını da analiz ediyoruz zira önerilen çerçevemiz OriginShield sunucularının işbirliğini sağlamakta ve bu durum Hit oranını etkileyebileceğini düşünüyoruz.

Hit oranı aşağıdaki gibi tanımlanır:

$$Hit\ Oranı = \frac{Replica\ Sunucu\ ve\ ya\ OriginShield'lerden\ cavaplanan\ isteklerin\ sayısı}{Toplam\ gelen\ istek\ sayısı}$$

NOS'un performansını değerlendirmek için **Tablo 3.2**'de belirtilen parametreleri kullanarak iki test tasarladık. İlk test uçtan uca yönlendirme yolundaki gecikme performansını ölçmeyi amaçlar. İkinci test NOS çerçevesinin Hit oranı başarısını değerlendirmek için uygulanır.

Uçtan uca yönlendirme yolu gecikmesini değerlendirmek için, önce NOS motorunu devreye almadan gecikmeyi ölçtük. Bu durumda herhangi bir OriginShield'e gelen talep, ISP'lerin yaptığı değişikliklere maruz kalan alttakı ağı (Underlay Network) kullanılarak doğrudan kaynak sunucularına (Origin) iletildi. Aynı zamanda, NOS'un önerdiği yolun gecikme süresini ölçtük. Önerilen yollar, uçtan uca rota gecikme süresine ve OriginShield sunucu yüklerine bağlı olarak, doğrudan Origins'e veya dolaylı olarak başka bir ara OriginShield üzerindendir.

NOS'un performansını Hit oranı açısından değerlendirmek için, Türkiye'de bulunan bir canlı ve kullanımda olan Origin hesabı seçtik. Test amacıyla 20 resim içeriğini rastgele seçildi. Her test için, Zipf dağılımı $\alpha = 1.35$ kullanarak resimlere rastgele popülerlik değerleri atadık ve bir önceki adımda elde edilen popülerliği kullanarak bu nesneler için HTTP istekleri oluşturduk. Her test çalışması için müşteri taleplerini simüle etmek için Apache JMeter'i kullanıldı. Her bir OriginShield sunucularından birini hedefleyen ve son kullanıcı davranışlarını ve isteklerini simüle eden toplam dört JMeter komut dosyası oluşturuldu.

Toplam 10 test yaparak Hit oranı sonuçlarını %95 güven aralığında (Confidence Interval) hesapladık. Bu işlem NOS framwork ile ve NOS framework olmadan ayrı olarak hesaplandı.

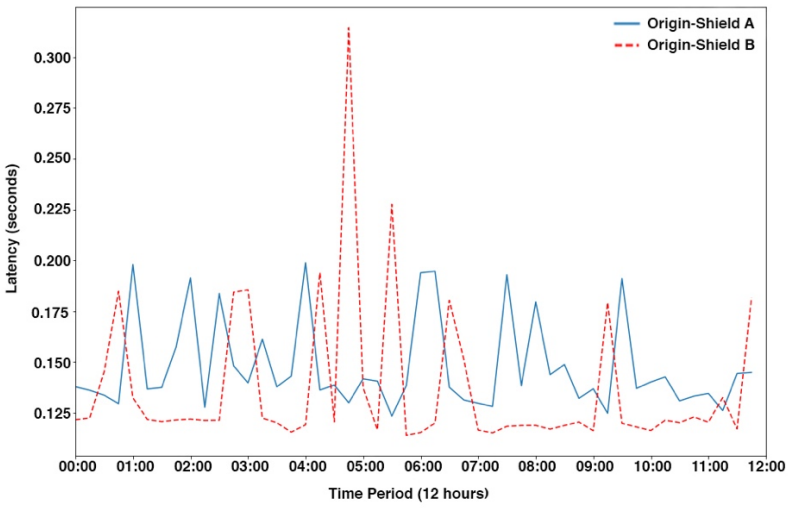
4. BULGULAR

Bir önceki bölümdeki Önerdiğimiz NOS Çerçevesinin asıl amacı gecikmelere sebep olan ISP'lerdeki yönlendirme değişikliklerini minimuma düşürüp aynı zamanda dalgalanmaları azaltmaktır. Önerdiğimiz yöntemin bu dalgalanmalar ve gecikmeler üzerinde nasıl bir etki ettiğini görmek için 3.6 bölümünde açıklanan testleri 3.5'de açıklanan test ortamında gerçekleştirdik.

Bu bölümün kalan kısmında, önce 4.1 bölümünde analizlerimizi açıklayacağız, sonra 4.2 bölümünde testlerimizin sonuçlarını göstereceğiz.

4.1 ANALİZLER

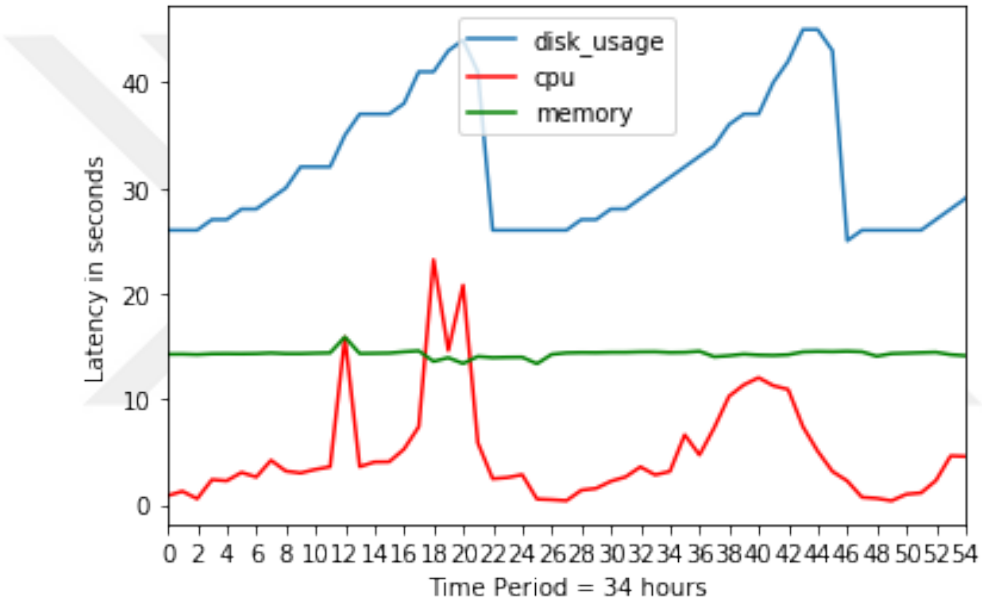
İlk yaptığımız analiz örnek iki OriginShield ile bir kaynak sunucu arasındaki gecikme değerleri oldu. Bu işlemi 12 saat süre için gerçekleştirdik ve sonuçlarda şekil 4.1'de mevcuttur. İki farklı OriginShield'den tek bir kaynak sunucusuna aynı zaman aralıklarda gecikme değerlerini ölçerek grafik halinde sunduk. Şekil 4.1'de göründüğü gibi OriginShield B genel olarak çoğu zaman daha iyi bir gecikmeye sahip olmasına rağmen bazı zaman dilimlerinde OriginShield A daha iyi bir performans göstermiştir. Bu durum aslında alttakı ağ üzerinde uygulanan bazı yönlendirme ve ya diğer değişikliklerden oluşmuş olabilir.



Şekil 4.1 Kaynak sunucu ile iki OriginShield arasındaki gecikme.

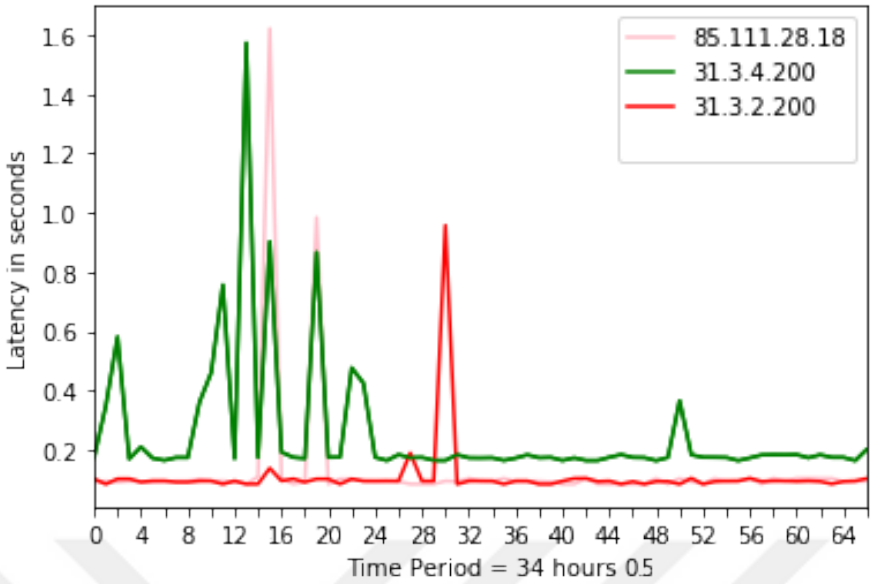
Şekil 4.1'de gösterilmiş olduğu durum bizim bu çalışmayı başlatma sebeplerimizdendir ve NOS çerçevesi ile bu dalgalanmaları ve gecikmelerde yükselmeleri engellemeye çalışacağır.

Gecikme değerine ek olarak, biz farklı zamanlarda sunucuların yük durumunu da geliştirdiğimiz API ile ölçtük. Örneğin şekil 4.2 bir OriginShield'in yük değerlerini göstermektedir. Bu yük değerindeki dalgalanma ve yükseliş ilgili zaman dilimindeki bu sunucuya gelen iş yükünü göstermektedir. Şekil 3.6, Algorithm 'de açıkladığımız gibi, NOS çerçevesinin bir başka amacı yönlendirmeleri yaptığında sunucuların yük değerini de hesaba katıp, sunucuların satürasyon yaşamasını engellemektir.



Şekil 4.2 Bir OriginShield'e ait yük değeri.

OriginShield'ler arasındaki ağ da bizim tarafımızdan kontrol edilmediği ve ISP'ler tarafından yönetildiği için, biz OriginShield'ler arasındaki gecikmelerde de dalgalanma ve bazen artış bekliyorduk. Bunun için kurduğumuz test ortamında, gecikme API'sini kullanarak 4 OriginShield arasındaki gecikmeyi de 34 saat için ölçtük. Şekil 4.3'te gösterilen veriler tek bir OriginShield'den diğer 3 OriginShield'e olan gecikmeyi göstermektedir. OriginShield'lerin IP'si bu grafikte gösterilmektedir. Bu şekilde gösterildiği gibi OriginShield A ile diğer OriginShield'ler arasındaki gecikme zaman zaman değişiklikler göstermektedir.



Şekil 4.3 OriginShield A ile diğer OriginShield'ler arasındaki gecikme.

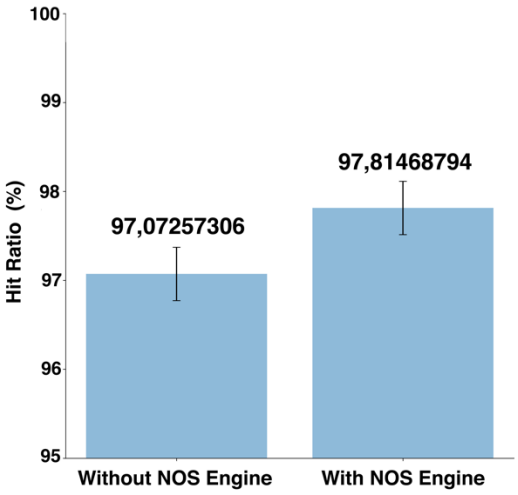
4.2 SONUÇLAR

Şekil 4.5 daha önce **Şekil 3.7** iletanımlanan 4 farklı Origins'den 4 OriginShield sunucusuna uçtan uca yönlendirme yolu gecikmelerini gösterir. Şekil 4.4 Hit oranı sonuçlarını göstermektedir.

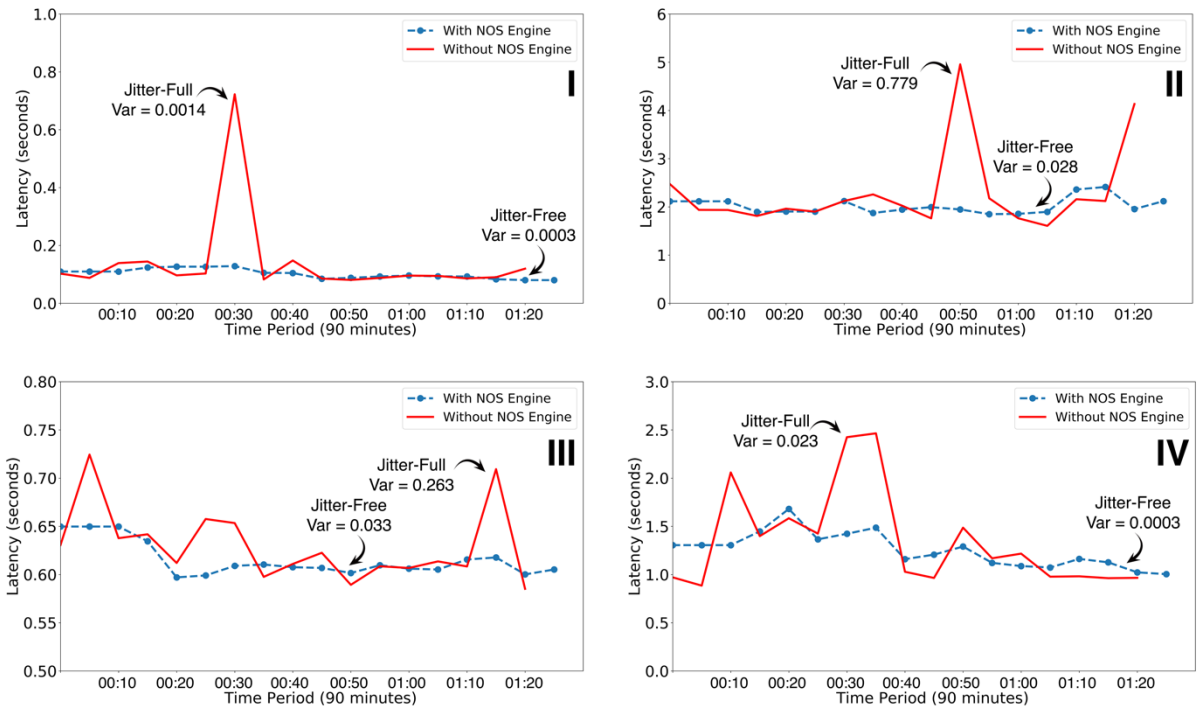
NOS olmadan seçeneği, her OriginShield alttaki ağ tarafından sağlanan yönlendirme yollarını kabul ettiği ve doğrudan Origin sunucularına ulaşmaya çalıştığı durumu göstermektedir ve *NOS İle* seçeneği merkezi modül tarafından hesaplanan ve sunulan bindirmeli ağ üzerinden geçen durumu gösteriyor.

Şekil 4.4 ve 4.5'te de gösterildiği gibi, teklif edilen çerçevemiz üç ana avantaja sahiptir.

İlk olarak, uçtan uca yönlendirme yolu gecikmeleri açısından, NOS yönlendirme yolu gecikmesini azaltır ve varsayılan yaklaşıma kıyasla daha kısa bir yol sağlar. Gecikmeye duyarlı ve dalgasız bindirmeli ağ yönlendirme algoritmamızın doğası gereği grafiklerde örtüşmeler görebiliriz. Bu noktalarda ya T_{Delay} gereksinimleri karşılamamış ya da T_{Loadl} tatmin olmamış ve bu nedenle bu örtüşmeler meydana gelmiştir.



Şekil 4.4 Hit Oranı NOS ile ve NOS Olmadan.



Şekil 4.5 Uçtan uca yönlendirme yolu gecikmeleri NOS ile ve NOS Olmadan.

İkincisi, NOS daha kararlı ve dalgasız bir yönlendirme rotası sağlar ve bu rota keskin yönlendirme yolu gecikme salınımlarına sahip değildir. Biz varyans değerini (Variance)

dalgalanma seviyesi göstergesi olarak kabul ediyoruz. Eğer bir rota yüksek varyansa sahipse, bu rotayı dalgalı bir rota olarak sayabiliriz. Şekil 4.5'te gösterilen *NOS ile* ve *NOS olmadan* yolların varyansının karşılaştırarak, NOS yönteminin daha düşük varyanslı yollar sağladığını görebiliriz. Ancak, NOS olmayan yolların tümünde daha yüksek varyans gözlemlenmektedir. NOS, sırasıyla varyansı Şekil 4.5 -I'de% 78, Şekil 4.5 -II'de% 96, Şekil 4.5 -III'de% 87 ve Şekil 4.5 IV'de % 98 azaltmıştır. Böylece NOS her durumda dalgasız bir rota sağlamıştır.

Üçüncüsü, NOS, Şekil 6 (b) 'de gösterildiği gibi Hit oranını % 0.74 oranında başarıyla arttırmıştır. CDN'ler için Hit / Miss oranının önemi göz önüne alındığında, bu iyileştirme buyuk bir önem arz etmektedir. Zira CDN'ler genel olarak müşterilerini memnun etmek için yüksek bir Hit oranına sahip olmaları gerekiyor. Ancak CDN firmaları bu oranı arttırmak için genellikle Önbellek yenileme (Cache Replacement) yöntemlerine başvururlar. Bu yöntemleri uygulama CDN'leriçin masraflı ve komplike bir işlemdir.

5. TARTIŞMA VE SONUÇ

Yapılan bu çalışmada CDN'leri İnternet'in dinamikliğine uyarlamayı amaçlayan yeni bir bindirme yönlendirme çerçevesi önerdik. ISP'ler ve ağ koşulları nedeniyle yönlendirme yollarındaki beklenmeyen değişiklikler meydana gelir. Bu değişiklikler ile başa çıkmak için, NOS, alttaki ağdaki yol gecikmelerini düzenli aralıklarla ölçmektedir, bu ölçümleri bindirmeli ağdaki bileşenlerin yüküyle (yani CPU, disk G / Ç ve ağ kullanım yüzdesi) birleştirir ve CDN sunucularından Origins'e en hızlı yönlendirme yollarını hesaplayıp bu yolları OriginShield'lere sağlar.

Gerçek bir CDN test ortamındaki gerçekleştirdiğimiz testler, NOS'un Origins'e daha kısa, daha istikrarlı ve dalgasız bir yönlendirme yolu sağladığını ve Hit oranını iyileştirdiğini göstermektedir.

Geliştirmiş olduğumuz yeni Ağ Yönetim Sistemi sayesinde ISP'ler nedeniyle içerik servisi performansında düşüşler yaşanmayacaktır. Her zaman gecikmesi daha düşük yollar seçileceğinden içerik servis hızı daha yüksek olacaktır. Sistemimiz en kısa yolu belirlemede sunucu yoğunluklarına da bakacağı için sunucular satürasyona uğramayacaklardır.

Tüm bunlar içerik servis kalitesinin artmasını sağlayacaktır. Dünya çapında CDN hızlarını ölçen kuruluşlar vardır. Kurum ve kuruluşlar bu ölçümlere göre CDN firmalarını seçmektedirler. CDN firmaları da NOS çerçevesini kullanarak artan içerik servis hızıyla sıralamalarda en üstlerde yer alacaklardır.

Gelecekteki bir çalışma olarak Gecikmeye duyarlı ve dalgasız bindirmeli ağ yönlendirme algoritmasını geliştirmeyi planlıyoruz. NOS'un çalışmasını optimize etmek için eşik ve ağırlık değerlerini analitik olarak belirleyeceğiz. Bu ağırlıkları belirlemek için makine öğrenme yöntemlerini kullanmayı ön görüyoruz.

KAYNAKLAR

Akamai. (tarih yok). *Akami Facts & Figures* . May 2019 tarihinde https://www.akamai.com/uk/en/about/facts-figures.jsp adresinden alındı

Apache JMeter,". (2019, May). https://jmeter.apache.org/ adresinden alındı

B. Molina, V. R. (2004). A closer look at a content delivery network implementation. *12th IEEE Mediterranean Conference.* Valencia, Spain.

C. Huang, A. W. (2008). Measuring and evaluating large-scale CDNs. *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement.* Vouliagmeni, Greece.

C. Lumezanu, R. B. (2009). Triangle inequality variations in the internet. *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement.* New York, USA.

CDN net. (tarih yok). https://cdn.net/push-vs-pull-cdn/ adresinden alındı

CDNPlanet. (2017, March 27). https://www.cdnplanet.com/guides/origin-shield/ adresinden alındı

Cereghetti, A. N. (November 9th, 2012). *MASTER THESIS : Global evaluation of CDNs performance using PlanetLab.* Universitat Politècnica de Catalunya.

Cloudflare - What is Anycast DNS? | How Anycast Works With DNS . (tarih yok). (Cloudflare) May 20, 2019 tarihinde https://www.cloudflare.com/learning/dns/what-is-anycast-dns/ adresinden alındı

Cloudflare - What is DNS ? . (tarih yok). May 20, 2019 tarihinde https://www.cloudflare.com/learning/dns/what-is-dns/ adresinden alındı

D. S. Berger, R. K.-B. (2016). Achieving high cache hit ratios for cdn memory caches with size-aware admission.

D. Wang, S. Z. (2018). Identifying influential factors of cdn performance with large-scale data analysis. *2018 International Conference on Computing, Networking and Communications (ICNC).* Hawaii.

DigitalOcean - What is Load Balancing? . (2017, February). May 20, 2019 tarihinde https://www.digitalocean.com/community/tutorials/what-is-load-balancing adresinden alındı

F. Ahmed, Z. S. (2016). Optimizing internet transit routing for content delivery networks. *2016 IEEE 24th International Conference on Network Protocols (ICNP).*

Group, 1. -N.-G. (2018). IEEE Standart for ContentDelivery Protocols of NextGeneration Service OverlayNetworks.

H. H. Zheng, E. K. (2005). Internet Routing Policies and Round-Trip-Times . *Passive and Active Network Measurement.*

Huberman, L. A. (2002). Zipf's law and the internet. *Geometrics* (s. 143-150). içinde

J. Wang, B. B. (2019). Maximizing page-level cache hit ratios in largeweb services. *ACM SIGMETRICS Perform. Eval. Rev, 46*(2), 91-92.

John Dilley, B. M. (07 November 2002). Globally Distributed Content Delivery. *IEEE Internet Computing, 6*(5), 50 - 58.

Limelight Networks. (tarih yok). https://www.limelight.com/origin-shield/ adresinden alındı

M. Day, B. C. (March 2001). A model for content internetworking. Internet-Draft.

M. Green, B. C. (November 2000). CDN peering architectural overview. Internet-Draft.

Max CDN. (tarih yok). https://www.maxcdn.com/one/faq/#push-zone-questions adresinden alındı

MaxCDN FAQ. (tarih yok). https://www.maxcdn.com/one/faq/#pull-zone-questions adresinden alındı

N. B. Hassine, D. M. (2015). Popularity prediction in content delivery networks. *IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC).*

NGINX - What Is DNS Load Balancing? . (tarih yok). May 20, 2019 tarihinde https://www.nginx.com/resources/glossary/dns-load-balancing/ adresinden alındı

NGINX - What Is Load Balancing? . (tarih yok). https://www.nginx.com/resources/glossary/load-balancing/ adresinden alındı

P. Medagliani, S. P. (2017). Overlay routing for fast video transfers in CDN. *IFIP/IEEE Symposium on Integrated Network and Service Management.*

S. J. Jafari, H. N. (2018). Investigating Performance Metrics for Evaluation of Content Delivery Networks . *Australian Symposium on Service Research and Innovation*, (s. 131–145).

S. J. Jafari, H. N. (2018). Investigating Performance Metrics for Evaluation of Content Delivery Networks . *Service Research and Innovation* (s. 131–145). Springer International Publishing.

Stephanos Androutsellis-Theotokis, D. S. (December 2004). A survey of peer-to-peer content distribution technologies . *ACM Computing Surveys (CSUR) , 36*(4), 335-371.

T. Oh-ishi, K. S. (2003). Study of the Relationship between Peer-to-Peer Systems and IP Multicasting. *IEEE Communications Magazine, 41*, 80-84.

What is Proxy Caching? . (2016, May 13). https://www.maxcdn.com/one/visual-glossary/proxy-caching/ adresinden alındı

What is Reverse Proxy Server. (tarih yok). May 22, 2019 tarihinde https://www.imperva.com/learn/performance/reverse-proxy/ adresinden alındı

Wikipedia - Proxy server . (tarih yok). http://www.wikizero.biz/wiki/en/Proxy_server adresinden alındı

Y. Zhu, Y. C. (New York, USA). Taming the triangle inequality violations with network coordinate system on real internet. *Proceedings of the Re-Architecting the Internet Workshop.* 2010.

EKLER

ÖZGEÇMİŞ

Kişisel Bilgiler	
Adı Soyadı	NİMA NAJAFLOU
Doğum Yeri	IRAN
Doğum Tarihi	07.09.1986
Uyruğu	❒ T.C. ☑ Diğer: IRAN
Telefon	05443167244
E-Posta Adresi	najaflou@gmail.com
Web Adresi	



Eğitim Bilgileri	
Lisans	
Üniversite	Nabi Akram University Tabriz
Fakülte	Mühendislik Fakültesi
Bölümü	Information Technology
Mezuniyet Yılı	2011

Yüksek Lisans	
Üniversite	İstanbul Üniversitesi-Cerrahpaşa
Enstitü Adı	Lisansüstü Eğitim Enstitüsü
Anabilim Dalı	Bilgisayar Mühendisliği Anabilim Dalı
Programı	Bilgisayar Mühendisliği

Makale ve Bildiriler
N. Najaflou, A. Arıs, B. Canberk, and Z. G. Aydın, “The Nearest Origin-Shield (NOS): A Jitter-Free Overlay Routing Framework for Content Delivery Networks ” in 2019IEEE International Symposium on Networks, Computers and Communications (ISNCC), June 2019