Teknoloji Fakültesi

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

BİTİRME PROJESİ

Farklı Yük Dengeleme Algoritmaları ve Eş Zamanlı Kullanıcı Sayısının Ağ Performansına Etkisi

PROJE YAZARI

Aziz Eren SAĞANDA

DANIŞMAN

Dr. Öğr. Üyesi Eyüp Emre ÜLKÜ

İSTANBUL, 2024

MARMARA ÜNİVERSİTESİ

TEKNOLOJİ FAKÜLTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Marmara Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Öğrencisi Aziz Eren SAĞANDA'nın "Farklı Yük Dengeleme Algoritmaları ve Eş Zamanlı Kullanıcı Sayısının Ağ Performansına Etkisi" başlıklı bitirme projesi çalışması, 03/06/2024 tarihinde sunulmuş ve jüri üyeleri tarafından başarılı bulunmuştur.

Jüri Üyeleri

Dr. Öğr. Üyesi Eyüp Emre ÜLKÜ (Danışman)

Marmara Üniversitesi ..... (İMZA) .....

Prof. Dr. Serhat ÖZEKEŞ (Üye)

Marmara Üniversitesi ..... (İMZA) .....

Dr. Öğr. Üyesi Timur İNAN (Üye)

Marmara Üniversitesi ..... (İMZA) .....

İÇİNDEKİLER

Sayfa

KISALTMALAR/ABBREVIATIONS

TCP : Transmission Control Protocol

UDP : User Datagram Protocol

IP : Internet Protocol

HTTP : Hypertext Transfer Protocol

URI : Uniform Resource Identifier

DoS : Denial of Service

DDoS : Distributed Denial of Service

ŞEKİL LİSTESİ

Sayfa

TABLO LİSTESİ

Sayfa

ÖZET

Yük dengeleme, esnek, değişken iş yüklerine uyum sağlayabilecek ve geliştirmeye açık, uzun ömürlü ağ ve sistem altyapıları tasarlamaya yarar. Yük dengeleme, ağ trafiğini bir uygulamayı destekleyen bir kaynak havuzuna eşit olarak dağıtma yöntemidir. Yüksek miktarda trafiği işlemek için uygulamaların çalıştığı kaynak sunucuları arasındaki yükü eşit şekilde kullanılmasını sağlar. Bu çalışmada çeşitli yük dengeleme algoritmaları kıyaslanarak hangi algoritmanın hangi koşullar altında daha verimli olduğunun bulunması amaçlanmıştır. Taşıma katmanı protokolü olarak Transmission Control Protocol (TCP) ve uygulama katmanı protokolü olarak HTTP kullanılmıştır. Ağ topolojisinin simülasyonu için PNETLAB, simülasyonun çalışacağı hostlarda altyapı platformu olarak Docker kullanılması hedeflenmiştir. Ağ yapısının görselleştirilmesi için PNETLAB kullanılarak literatürdeki çalışmalarda önerilen ağ yapılarının modellemesi yapılacaktır. Simülasyon ortamları, gerçekçi ortamlar, beceri geliştirmek için pek çok avantaja sahiptir. Bu sebeple çalışmada sanallaştırma ortamlarından faydalanılması planlanmaktadır. Çalışmanın sunduğu katkılardan bir diğeri de sistem tasarımı yapacak kişilerin bulgulardan faydalanarak optimum performansı veren algoritmayı seçebileceklerdir.

ABSTRACT

Load balancing helps to design long-lasting network and system infrastructures that are flexible, adaptable to changing workloads and open to improvement. Load balancing is a method of distributing network traffic evenly across a pool of resources that support an application. It ensures that the load between the resource servers where applications run to handle large amounts of traffic is evenly utilized. In this study, we compare various load balancing algorithms to find out which algorithm is more efficient under which conditions. Transmission Control Protocol (TCP) is used as the transport layer protocol and HTTP as the application layer protocol. PNETLAB was used to simulate the network topology and Docker was used as the infrastructure platform on the hosts where the simulation will run. For the visualization of the network structure, PNETLAB will be used to model the network structures proposed in the studies in the literature. Simulation environments, realistic environments, have many advantages for developing skills. For this reason, virtualization environments are planned to be used in the study. Another contribution of the study is that system designers will be able to choose the algorithm that gives the optimum performance by utilizing the findings.

Bölüm 1 – GİRİŞ

Bilişim teknolojilerinin ilk günlerinden bu yana sürekli artan kapasite ihtiyacı, veri merkezlerinin kurulmasında ve küresel olarak yaygınlaşmasında önemli rol oynamıştır. Değişen teknolojik ihtiyaçlara ve artan veri yüklerine uyum sağlamak için, veri merkezlerinin tasarımları da değişmiştir. Bu merkezlerde etkin bir hizmet sunabilmek adına, trafiğin sunucular arasında dengeli bir şekilde dağıtılması zorunlu hale gelmiştir [1].

Ağ protokollerinin ilk ortaya çıkışları, küçük ölçekli ağlar için yeterli olmuşken, büyüyen ağ ihtiyaçları ile bu protokoller de dönüşüme uğramıştır. Bu evrim süreci, başlangıçta yüksek maliyetler getirse de günümüzdeki iletişim altyapılarının temelini oluşturmuştur. Günümüzde kullanılan çok sayıda iletişim protokolü, cihazlar arası etkileşimin temelini oluşturmaktadır. Etkili iletişim için temel gereklilik, protokolleri kullanan tarafların standartlara bağlı bir şekilde bunları yerine getirmesidir. Bu gereksinim, protokollerin standartlaştırmakla görevli merkezi otoritelerin kurulmasına yol açmıştır[1]

Günümüzde internet, çevrimiçi gazete okumaktan film izlemeye kadar sayısız uygulama alanında kullanılan en yaygın iletişim aracı olarak konumlanmıştır. Yönlendirme ve yük dengeleme algoritmaları tarafından yönetilen bu muazzam trafiğin her yere ulaşması, modern dijital varlığımızın birbirine bağlı dokusunun temelini oluşturarak vazgeçilmez hale gelmiştir[2]

Dağıtık hesaplama ortamlarını yönetmenin zorluklarından biri farklı türdeki hesaplama elemanlarını ve çok sayıdaki farklı türden kaynağı bir arada yönetmek olmasıdır[3]. Farklı türden yük durumlarını dağıtabilmek için farklı yük dağıtımı algoritmaları kullanılmaktadır. Bu algoritmaların hedefi, yükü sistemde müsait olan kaynaklara verimli bir şekilde dağıtmaktır. Jagdish Chandra Patni ve arkadaşları bu çalışmada her türlü grid yapısını idare edebilen dağıtık bir yük dengeleme algoritması önermişlerdir. Önerilen algoritma iki adıma ayrılmıştır: İşlerin yürütülmesi için geçen süreyi azaltmak veya bir grid mimarisinde işlerin bir hesaplama düğümünden diğerine aktarılması arasındaki yanıt süresini ve iletişim maliyetini azaltmak diyebiliriz. Önerilen algoritmada, iletişim maliyetini azaltmak için yükün öncelikle yerel düzeyde dengeleneceği gösterilmiştir [4].

Bu çalışmanın temel amacı, çeşitli yük dengeleme algoritmalarının etkinliğini değerlendirmek ve bu algoritmaların ağ performansı üzerindeki etkilerini analiz etmektir.

Yük dengeleme, özellikle büyük ölçekli ve dinamik veri merkezlerinde, sunucu kaynaklarının optimum kullanımı için kritik bir öneme sahiptir. Bu bağlamda çalışma kapsamında, algoritmaların performans ölçütleri, işlem süreçlerinin yanıt verme hızları ve sistemin genel kararlılığı üzerine yoğunlaşılmıştır. Ayrıca, modern ağ yapılarında karşılaşılan gerçek dünya senaryolarına dayalı simülasyonlar ve testler ile teorik bulguların pratik uygulamalara dönüşümü sağlanmaya çalışılmıştır. Çalışma, yük dengeleme tekniklerinin daha iyi anlaşılması ve bu tekniklerin daha etkili bir şekilde tasarlanıp uygulanması için gereken bilimsel verileri sunmayı amaçlamaktadır. Sonuç olarak, bu tez, yük dengeleme algoritmalarının iyileştirilmesine yönelik stratejik öneriler geliştirerek literatüre önemli bir katkı sağlayacak ve ağ yönetimi pratiklerinde yenilikçi çözümlerin önünü açacaktır.

1.1. Literatür Taraması ve İlişkili Çalışmalar

Literatür tarama sürecinde IEEE Xplore Digital Library, Springer, ScienceDirect, Elsevier ve ACM Digital Library veritabanları kullanılmıştır. Literatür taraması sırasında kullanılan anahtar kelimeler: “Load Balancing”, “Containerization”, “GNS3”, “IPerf3”, “Grid Computing” olarak seçilmiştir. Yayın yılı olarak 1995-2024 seçilip anahtar kelimeler güvenilir kaynakların ilgili portallarında tek tek veya kelimelerin kombinasyonu olarak aratılarak farklı kaynaklardan yayınlar seçildi. Bulunan yayınlar belli kriterlere göre değerlendirilerek tekrar elendi.

Malek Al-Zewairi ve arkadaşları gerçekleştirdikleri çalışmada, Yazılım Tanımlı Ağ (SDN) üzerinde IP ve MAC spoofing saldırılarını tespit etmeyi ve önlemeyi amaçlayan Open vSwitch Denetleyicisine dayalı Yazılım Tanımlı Güvenlik için deneysel bir denetleyici önermişlerdir [5]. Yazarlar, çalışmada sistem yönetimi ve ağ yönetimini kıyaslayarak; sistem yönetiminde ölçek büyüdükçe maliyetin azaldığını fakat ağ yönetiminde bunun tersine yapının karmaşıklaştığını ve maliyetin arttığını belirtmişlerdir. Bu sebepten ağ yönetiminde geleneksel modelin yerine SDN kullanılarak bu gibi dezavantajlardan kaçınılabileceğini öngörmektedirler[5].

Günümüz internet ağı hızla genişlemekte ve bu durum, tüketicileri daha fazla sunucu hizmetine bağlanmaya teşvik etmektedir. Bu koşulları yönetebilmek için güvenilir bir sunucu sistemi gereklidir. Birden fazla sunucu kullanmak, yanıt sürelerini etkileyebilir. Yük dengeleme yöntemiyle sunucu sistem hizmetlerini iyileştirmek için ağ mimarisi ve

algoritmaların analizi, testi ve değerlendirilmesi gereklidir. Wira Harjanti ve arkadaşları yaptıkları çalışmada, optimal yanıt süresi değerini belirlemek amacıyla, en az bağlantı (least-connection) ve round-robin algoritmaları karşılaştırılmıştır [6]. 500 bağlantı/saniye için 1000 istek ve 600 bağlantı/saniye için 1200 istek testlerinde round-robin algoritması daha iyi görünmektedir. Ancak, 700 bağlantı/saniye için 1400 istek, 800 bağlantı/saniye için 1600 istek ve 900 bağlantı/saniye için 1800 istek testlerinde en az bağlantı algoritması üstün görünmektedir. Artan bağlantı sayısında, en az bağlantı algoritmasının ortalama yanıt süresi daha küçüktür, bu da sunucu performansını artırır.

Bulut mimarisinin daha güvenilir bir hale gelmesi ve yaygınlaşmasıyla birlikte geleneksel uygulamaların buluta taşınması için konteynır teknolojilerinin önemi artmıştır. Application Containerization Advisory Framework for Modernizing Legacy Applications adlı çalışmada yazarlar, ACA adı verilen, geleneksel uygulamaları konteynır ortamına taşımaya yardımcı olan bir framework önermektedir.

Bu framework yüksek seviye dillerden faydalanarak veri tabanı, middleware ve uygulama sunucusu gibi elemanları konteynırize etmeyi amaçlamaktadır. Bu framework aynı zamanda makine öğrenmesi algoritmalarını da kullanarak insan eforunu minimize ederek migrasyonu en az çabayla gerçekleştirebilmektedir [7].

Amin Vahdat, Enter the Andromeda Zone adlı makalesinde Google Cloud Platform'da kullanılan Andromeda network stackten genel hatlarıyla bahsediyor. Andromeda'nın amacı yüksek seviyede performans ve erişilebilirlik sağlamaktır. Bunu yapabilmek için en alt seviyedeki donanımdan en üst katmandaki yazılım elemanlarındaki programlanabilirlikten faydalanmaktadır. Bu programlanabilir yapı sayesinde geleneksel yapıdaki ek noktalarından zafiyetli yapılar kurmak yerine uçtan uca güvenli ve yüksek performanslı yapılar kurmayı mümkün kılmaktadır.

Andromeda'nın amacı ağ fonksiyonu sanallaştırma kullanarak ağ altyapısından tam performans alabilmektir. Fonksiyon sanallaştırma sayesinde DdoS koruma, transparan yük dengeleme ve güvenlik duvarı işlevleri tek kalemde toplanabilmektedir.

Performans testi için isabetli ölçümler yapabilmek kadar az kaynak tüketen maliyet odaklı sistemler geliştirmek de önem arz etmektedir. IPERF: A Framework for Automatic Construction of Performance Prediction Models adlı makalede maliyet/performans odaklı bir yazılım olan iperf'in çalışma mantığından bahsedilmektedir [8].

Stateless Datacenter Load-balancing with Beamer makalesinde Vladimir Olteanu ve arkadaşları, stateless işletim sağlamak için tasarlanmış bir yük dengeleyici olan Beamer'ın tasarımından bahsetimişlerdir. Makaledeki temel fikir, bağlantı durumunu backend sunucularda halihazırda depolanan bağlantı durumunu kullanarak, bağlantıların churn altında asla düşürülmemesini sağlamaktır. Stateless yük dengeleme birçok avantaj sağlar: Beamer'ın yazılım uygulaması, Google'ın Maglev adlı en gelişmiş yazılım yük dengeleyicisinden iki kat daha hızlıdır ve 7 çekirdekte 40Gbps HTTP uplink trafiğini işleyebilir. Beamer, P4 uygulaması gösterdiği gibi hem yazılımda hem de donanımda basit bir şekilde dağıtılabilir. Son olarak, Beamer, bağlantıları koparmadan bağımsız ölçekleme olaylarına izin verir [9].

Packet Level TCP Performance of NAT44, NAT64 and IPv6 Using Iperf in the Context of IPv6 Migration adlı makalede, IPv4 adreslerinin tükenmek üzere olduğunu ele almakta ve Filipinler'de IPv6 geçişinin benimsenmesini savunmakta ve NAT44'ün arkasındaki ağlar için uygulanabilir bir çözüm olarak NAT IPv6'dan IPv4'e (NAT64) geçişi önermektedir. Daha önce UDP performansına odaklanan bu makale, iPerf kullanarak paket düzeyinde TCP performansını incelemektedir [10]. Bulgular, IPv6 ve NAT64 ağlarının NAT44'ten sürekli olarak daha iyi performans gösterdiğini, IPv6'nın NAT44'e kıyasla aktarım süresinde %26, NAT64'ün ise %27'lik bir azalma gösterdiğini ortaya koymaktadır. Ayrıca, IPv6 ve NAT64 ağları, NAT44'ü sırasıyla %50 ve %33 oranında aşarak üstün bant genişliği kullanımı sergilemektedir. Genel olarak, sonuçlar IPv6 ve NAT64'ün NAT44'e kıyasla TCP metriklerinde daha iyi performans sunduğunu göstermektedir.

Dağıtık hesaplama ortamlarını yönetmenin zorluklarından biri farklı türdeki hesaplama elemanlarını ve çok sayıdaki farklı türden kaynağı bir arada yönetmek olmasıdır. Farklı türden yük durumlarını dağıtabilmek için farklı yük dağıtımı algoritmaları kullanılmaktadır. Bu algoritmaların hedefi, yükü sistemde müsait olan kaynaklara verimli bir şekilde dağıtmaktır. Distributed Load Balancing Model for Grid Computing Environment adlı makalede bu algoritmaların 3 tanesinden genel hatlarıyla bahsedilmektedir [11]. Bu algoritmaların kullanılması sonucu işlemlerin tepki süresi ve dağıtık sistemdeki düğümler arasında haberleşme maliyeti azalacaktır.

Dağıtık hesaplama ortamlarını yönetmenin zorluklarından biri farklı türdeki hesaplama elemanlarını ve çok sayıdaki farklı türden kaynağı bir arada yönetmek olmasıdır. Farklı türden yük durumlarını dağıtabilmek için farklı yük dağıtımı algoritmaları kullanılmaktadır.

Bu algoritmaların hedefi, yükü sistemde müsait olan kaynaklara verimli bir şekilde dağıtmaktır. Jagdish Chandra Patni ve arkadaşları bu çalışmada her türlü grid yapısını idare edebilen dağıtık bir yük dengeleme algoritması önermişlerdir. Önerilen algoritma iki adıma ayrılmıştır: İşlerin yürütülmesi için geçen süreyi azaltmak veya bir grid mimarisinde işlerin bir hesaplama düğümünden diğerine aktarılması arasındaki yanıt süresini ve iletişim maliyetini azaltmak diyebiliriz. Önerilen algoritmada, iletişim maliyetini azaltmak için yükün öncelikle yerel düzeyde dengeleneceği gösterilmiştir.

SDN mimarisi iletim ve kontrol katmanlarını ayırarak dağınık durumda olan ağlara merkezi bir yönetim arayüzü sağlar. Bu yapının sağlanabilmesi için OpenFlow standardı kullanılmaktadır.

OpenFlow standardında çeşitli optimizasyon algoritmaları kullanılmaktadır. Bu algoritmaların bazıları doğadaki sistemlerden esinlenerek geliştirilmiştir. Bu algoritmalara NIOAs (nature-inspired optimization algorithms) denilmektedir. Bu algoritmalar evrimsel, bio-esinlenmiş, fizik ve kimya temelli algoritmalar ve diğerleri olarak kategorize edilmektedir. Makalede bahsedildiği üzere Assefa ve Özkasap bu algoritmalardan faydalanarak SDN altyapısında güç tasarrufu üzerine çalışmalar yapmıştır. [12]

Amin Vahdat, Enter the Andromeda zone: Google Cloud Platform's latest networking stack adlı makalesinde Google Cloud Platform'da kullanılan Andromeda network stackten genel hatlarıyla bahsediyor. Andromeda'nın amacı yüksek seviyede performans ve erişilebilirlik sağlamaktır. Bunu yapabilmek için en alt seviyedeki donanımdan en üst katmandaki yazılım elemanlarındaki programlanabilirlikten faydalanmaktadır. Bu programlanabilir yapı sayesinde geleneksel yapıdaki ek noktalarından zafiyetli yapılar kurmak yerine uçtan uca güvenli ve yüksek performanslı yapılar kurmayı mümkün kılmaktadır.

Andromeda'nın amacı ağ fonksiyonu sanallaştırma kullanarak ağ altyapısından tam performans alabilmektir. Fonksiyon sanallaştırma sayesinde DdoS koruma, transparan yük dengeleme ve güvenlik duvarı işlevleri tek kalemde toplanabilmektedir.

Cloudflare'ın kenar veri merkezleri, sunucuların yükünü yönetmek ve kenar ağının güvenilirliğini ve işletme verimliliğini artırmak için Unimog adlı bir Katman 4 Yük Dengeleyici kullanır. Unimog, Cloudflare'ın kenar ağına özgü olarak uyarlanmıştır ve diğer Katman 4 Yük Dengeleyicilerinde kullanılan tekniklere dayanmaktadır. Bağlantının hangi sunucuya gideceğini kontrol eder, sunucuların yalnızca işlemde oldukları zaman bağlantıları

aldığından emin olur ve sunucuların yükünü yönetir. Unimog, Cloudflare'ın altyapısının ve hizmetlerinin güvenilirliğini, performansını ve ölçeklenebilirliğini artırmıştır.

Bölüm 2 – MATERYAL VE YÖNTEM

Bu bölüm, bu çalışmanın temelini oluşturan metodoloji, kullanılan teknolojiler, ölçüm teknikleri ve karşılaşılan zorluklar ile bunların nasıl üstesinden gelindiği hakkında bilgi vermektedir. Araştırma sürecinde yük dengeleme algoritmalarının işleyişi ve ağ protokollerinin temel prensipleri detaylandırılmıştır. Ayrıca, simülasyonlar için kullanılan sanallaştırma platformları ve performansın nasıl ölçüldüğüne dair kapsamlı açıklamalar yapılmıştır. Bu bölüm, araştırmanın yapısal çerçevesini, kullanılan araç ve teknikleri ve elde edilen verilerin nasıl işlendiğini anlatarak, çalışmanın şeffaflığını ve tekrarlanabilirliğini artırmayı hedeflemektedir.

2.1. Yük Dengeleme (Load Balancing)

Yük dengeleme, ağ trafiğini birden fazla sunucu arasında eşit şekilde dağıtarak sistem performansını ve güvenilirliğini artıran bir tekniktir. Bu işlem, gelen isteklerin belirli algoritmalar veya kurallar çerçevesinde en uygun sunuculara yönlendirilmesiyle gerçekleştirilir. Bu işlemin gerçekleştirilmesi için yük dengeleyicilere ihtiyaç duyulur. Yük dengeleyiciler değişken performans ihtiyaçları nedeniyle hem bağımsız birer cihaz olarak hem de yazılım olarak sunucu üzerinde barındırılabilmektedir. Yük dengeleme, sunucu altyapısının daha verimli ve ölçeklenebilir olmasını sağlar [13]. Sunucu yükünü dağıtarak, tek bir sunucu üzerindeki iş yükünü azaltır ve genel performansı optimize eder.

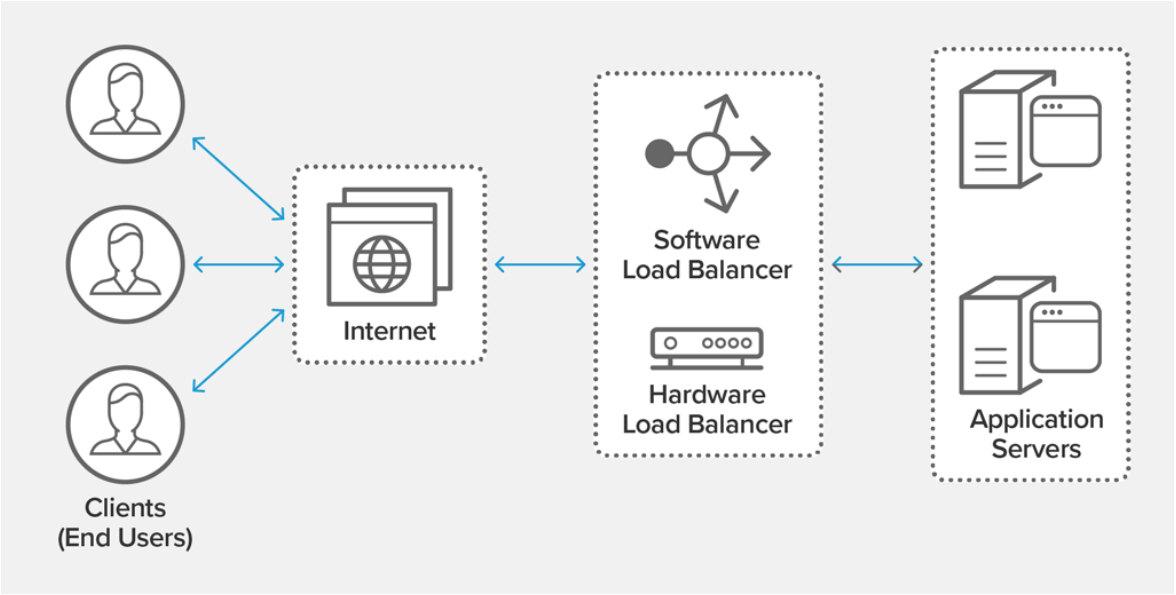
2.2. Yük Dengeleme Algoritmaları

İletişim protokolleri tarafından yönetilen birbirine bağlı cihazların topolojisi daha karmaşık bir hal aldıkça, bu protokollerin veri alışverişindekindeki rolünün önemi daha belirgin hale gelmektedir. Kesintisiz iletişime olan talep arttıkça, protokollerin rolünün yeri daha belirgin hale gelmektedir [14]. NAT ve IPv6 gibi yeniliklerin ortaya çıkması, IPv4 adres alanının tükenmesiyle birlikte güncellenen protokollere örnek olarak verilebilir.

Farklı ortamlar arasında kesintisiz iletişimin sağlanması için OSI modeli temel bir çerçeve olarak ortaya çıkmıştır [15]. Dünya çapında kullanılan OSI referans modeli, yönlendirme protokolleriyle birlikte Yerel Alan Ağları (LAN'lar) ve Geniş Alan Ağlarının (WAN'lar) birbirine bağlanmasını kolaylaştırmada önemli bir rol oynamaktadır. IPv4 adreslerinin sayıca kısıtlı olması, IP adreslerinin genel ve yerel olarak sınıflandırılmasına yol açmıştır [16]. Yerel IP’ler yerel ağlarla sınırlıdır ve erişimlerinin kısıtlanması için ağ cihazlarındaki kısıtlarla global genel ağ üzerinden trafiğe normal şartlar altında çıkamaz. Yerel ağda

bulunan cihazlar NAT adı verilen, yerel ağ adresini genel ağ adresine çeviren bir işlemden geçerek internete erişebilirler.

Bu tür gelişmelerle birlikte, yük dengeleme algoritmaları ağ performansını optimize etmek için gerekli bir hal almıştır. Yük dengeleme, yük altındaki sunucuların; çökmeden trafiği karşılayabilmesi için ağ trafiğinin veya işlem yükünün sunucular arasında dağıtılmasıdır [17]. Bu yaklaşım, tek bir cihazın gereğinden fazla yük taşımasının önüne geçerek verimliliği arttırır. Şekil 1'de bir yük dengeleyicinin çalışma prensibi görülebilir.



Şekil 1 *Load Balancer Çalışma Prensibi*

2.3. Round Robin

Ağ protokollerinin karmaşık mimarisinde, yük dengeleme algoritmaları, trafiğin kesintisiz akışını sağlar ve sunucuların darboğaza girmesini engellemekte kilit rol alır. Bu algoritmalar arasında Round Robin, OSI protokol yığınının yapılandırılmış sıralamasıyla paralellik gösteren temel bir yaklaşım olarak ortaya çıkmaktadır [18].

Round Robin mekanizmasında server ve client, geleneksel bir posta sistemindeki gibi iletişime geçmek için bir dizi kurala ihtiyaç duyar [18]. Algoritma gelen taleplerin bir grup sunucu arasında sırayla döngüsel olarak dağıtılması prensibine göre çalışır ve işlem yükünü adil olarak paylaştırmayı hedefler.

Eşit Dağıtım: Round Robin algoritması, gelen istekleri mevcut sunucular arasında eşit dağıtarak basitliğiyle öne çıkar. Bu tahsis metodu, her sunucunun isteklerden adil pay almasını sağlayarak ağdaki herhangi bir sunucuya aşırı yük binmesini önler [19].

Round Robin hem L4'te TCP-UDP hem de L7'de http yük dengelemede kullanılabilir. TCP bir aktarım protokolü olarak, bilgi akışının karşıya aktarımının kontrolünü yapar. [20], [21] UDP ise bir datagram protokolü olarak hız odaklıdır, paketlerin karşıya ulaştığını kontrol etmez [22]. HTTP ise uygulama katmanında çalışan bir protokol olup TCP'nin üzerinde çalışır ve modern web sayfalarında gezinebilmemiz için gerekli olan altyapıyı sunar [23]

Round Robin algoritmasının kullandığı basit yaklaşım, sistem yönetimini kolaylaştırmakla birlikte sunucu kapasitelerindeki veya iş yüklerindeki değişiklikleri hesaba katmayabilir. Bu sebeple dinamik sunucu performansı ihtiyaçları olan ortamlar için en uyarlanabilir çözüm olmayabilir.

Round Robin algoritması yük dengelemek için basit bir yaklaşım sunar ve böylelikle kolay bir şekilde isteklerin ilgili sunuculara aktarılması sağlar. Böylelikle birbirine bağlı sistemlerin verimliliğine ve esnekliğine katkıda bulunur [19]. Örnek bir Round Robin yük dengeleme mekanizması Şekil 2'de gösterilmiştir.

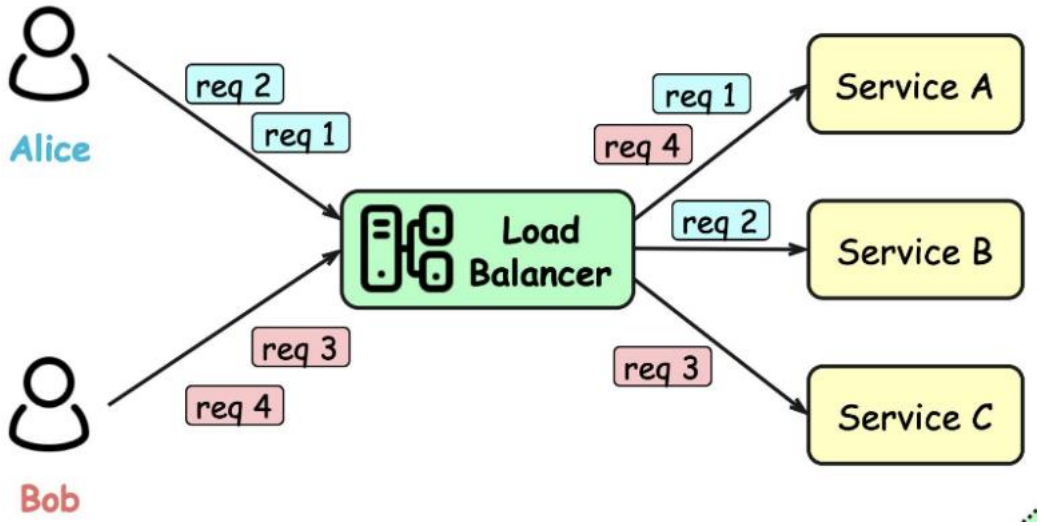
Ağ protokollerinin karmaşık çerçevesinde, yük dengeleme algoritmaları optimum kaynak kullanımını sağlamada ve sunucuların aşırı yüklenmesini önlemede etkilidir [24]. Bu algoritmalar arasında En Az Bağlantı yaklaşımı, yeni gelen isteği o sırada üzerinde en az bağlantı olan sunucuya aktararak sunucuların darboğaza girmesini önlemeyi hedefler [24], [25].

Least Connection algoritması, Round Robin'den farklı olarak sunucu durumundan bağımsız, sıralı bir paylaşım yapmak yerine sunucuların anlık durumunu göz önünde bulundurarak yük paylaşımı yaptığı için bazı durumlarda round robin algoritmasından daha avantajlı olabilir [26].

En az bağlantı mekanizmasında geleneksel posta sistemine benzer şekilde cihazlar sistematik bir dizi kuralla iletişime geçer. Algoritmanın temel mantığında gelen isteği en az aktif bağlantıya sahip olan sunucuya aktararak darboğazların önüne geçmek vardır.

Dinamik Dağıtım: En az bağlantı metodu, gerçek zamanlı trafik durumunda talepleri dinamik olarak dağıtmada diğer algoritmalara göre üstündür. En az bağlantı metodu değişen trafik yüklerine daha etkili bir şekilde tepki vererek diğer algoritmalardan farkını gösterir [27].

En az bağlantı algoritması kullanılarak hem TCP hem UDP bağlantılarda yük dengeleme yapılabilir [28]. Bir aktarım protokolü olarak TCP, paket aktarımının yönetilmesinde çok önemli bir rol oynar ve tıkanıklığı önlemek ve verimliliği en üst düzeye çıkarmak için sunucular arasında en az bağlantının sürdürülmesini sağlar.



Şekil 2 Round Robin Algoritması

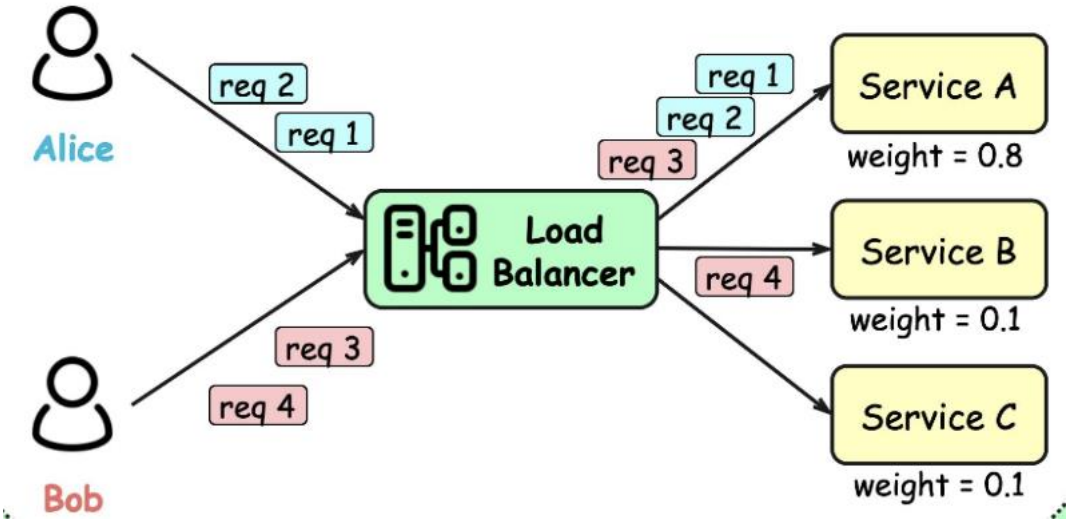
2.4. Ağırlıklı Round Robin

Ağırlıklı Round Robin algoritması, klasik olan denginden farklı olarak yükü sabit sırayla dağıtmak yerine belirli bir düzende önceliklendirerek dağıtım sağlar. Ağırlıklı Round Robin bir ağdaki tüm sunucuların aynı kapasitede olmadığı durumlarda diğer algoritmalara göre daha avantajlı olmaktadır. Bu yaklaşımda, sunuculara ağırlıklar atanarak, sunucu kapasitelerindeki farklılıkları hesaba katılarak gelen taleplerden orantılı bir pay ayrılır [29]. Şekil 3'te örnek bir Ağırlıklı Round Robin mekanizması gösterilmiştir.

Diğer algoritmalara benzer şekilde ağırlıklı round robin algoritması da yönlendirmeleri yaparken belli standartlardan faydalanır. Kaynak ve hedef adresler gibi bilgiler ilgili paketlere eklenerek cihazlar arasında yönlendirme yapılmasına katkı sağlamaktadır.

Ağırlıklı Round Robin'in, atanan ağırlıklar aracılığıyla sunucu kapasitelerini dikkate alan incelikli yaklaşımı, onu heterojen hesaplama kaynaklarına sahip ortamlar için uygun hale getirir [30]. Bu algoritma, basit Round Robin'e kıyasla daha uyarlanabilir bir çözüm sunarak değişen iş yüklerinin verimli bir şekilde ele alınmasını sağlar ve olası darboğazları önler.

Özetle, sunucu kapasitelerine dayalı ağırlık atama ilkelerine dayanan Ağırlıklı Round Robin yük dengeleme algoritması, görevlerin sunucular arasında dengeli bir şekilde dağıtılmasını teşvik etmede önemli bir rol oynar ve birbirine bağlı sistemlerin verimliliğine ve esnekliğine katkıda bulunur.



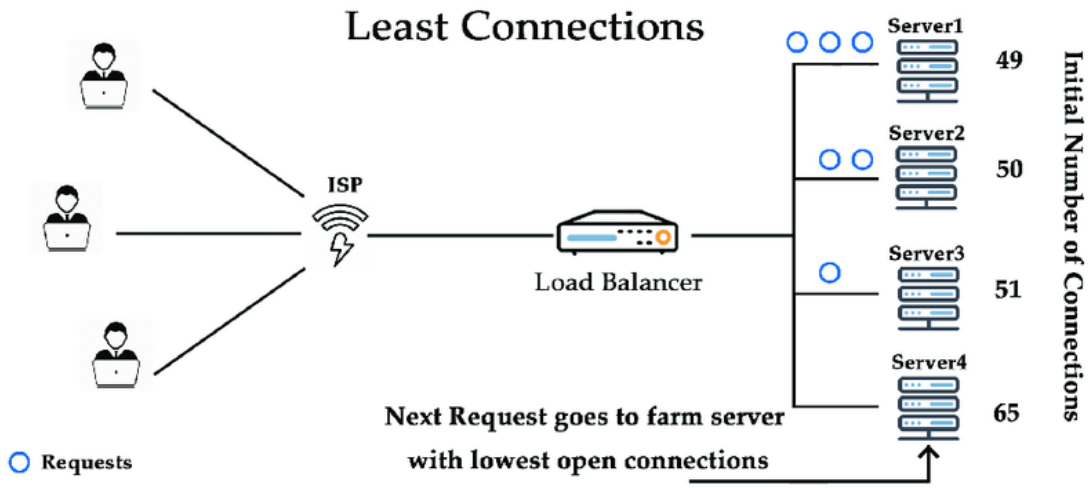
Şekil 3 Ağırlıklı Round Robin Algoritması

2.5. Rastgele Yük Dağıtım Algoritması

Bu algoritmada, gelen istekler rastgele bir şekilde sunuculara atanır. Bu, her sunucunun eşit sayıda istek alması ve eşit kapasitedeki sistemlerin uzun bir süre zarfında aynı yükü alması prensibine dayanır [31]. Algoritma son derece basittir ve herhangi bir özel konfigürasyon gerektirmez. Son derece basit ve kolayca uygulanabilir olması küçük sistemler için tercih edilebilir kılmaktadır. Kolaylığının yanında algoritmanın belli başlı dezavantajları vardır. Bu algoritma sunucuların işlem kapasitesi veya bellek gibi özelliklerini dikkate almaz. Bu sebeple verimliliği garanti etmez. Sunucu kapasiteleri benzer olduğunda, hızlı ve basit çözüm gereken durumlarda ve küçük topolojiler haricinde kullanılması önerilmez.

2.5.1. Least Connections (En Az Bağlantı) Algoritması

Least connection algoritması yeni gelen bağlantıları, o an en az sayıda aktif bağlantıya sahip olan sunucuya yönlendirir [32]. Bu sayede, sunucular arasında yükün eşit bir şekilde dağıtılması ve aşırı yüklenmelerin önlenmesi amaçlanır. Bunun sağlanabilmesi için algoritma her sunucu için aktif bağlantı sayısını takip eder ve yeni bir bağlantı geldiğinde algoritma trafiğin yönlendirilmesi için aktif bağlantı sayısı en düşük olan sunucuyu seçer. Bu işlem her yeni bağlantı için tekrarlanır. Bu algoritmanın kullanıldığı alanlara örnek olarak web sunucuları, veri tabanı sunucuları, uygulama sunucuları ve oyun sunucuları verilebilir [33]. Şekil 4'te örnek bir Least Connections yük dağıtım modeli gösterilmiştir. Sunucu arızalarına karşı dayanıklı olması bu algoritmanın tercih edilmesinde büyük bir etkendir. Anlık yük dalgalanmalarına karşı hassastır bu sebeple hızlı değişen trafik yoğunluklarında pek tercih edilmez [6].



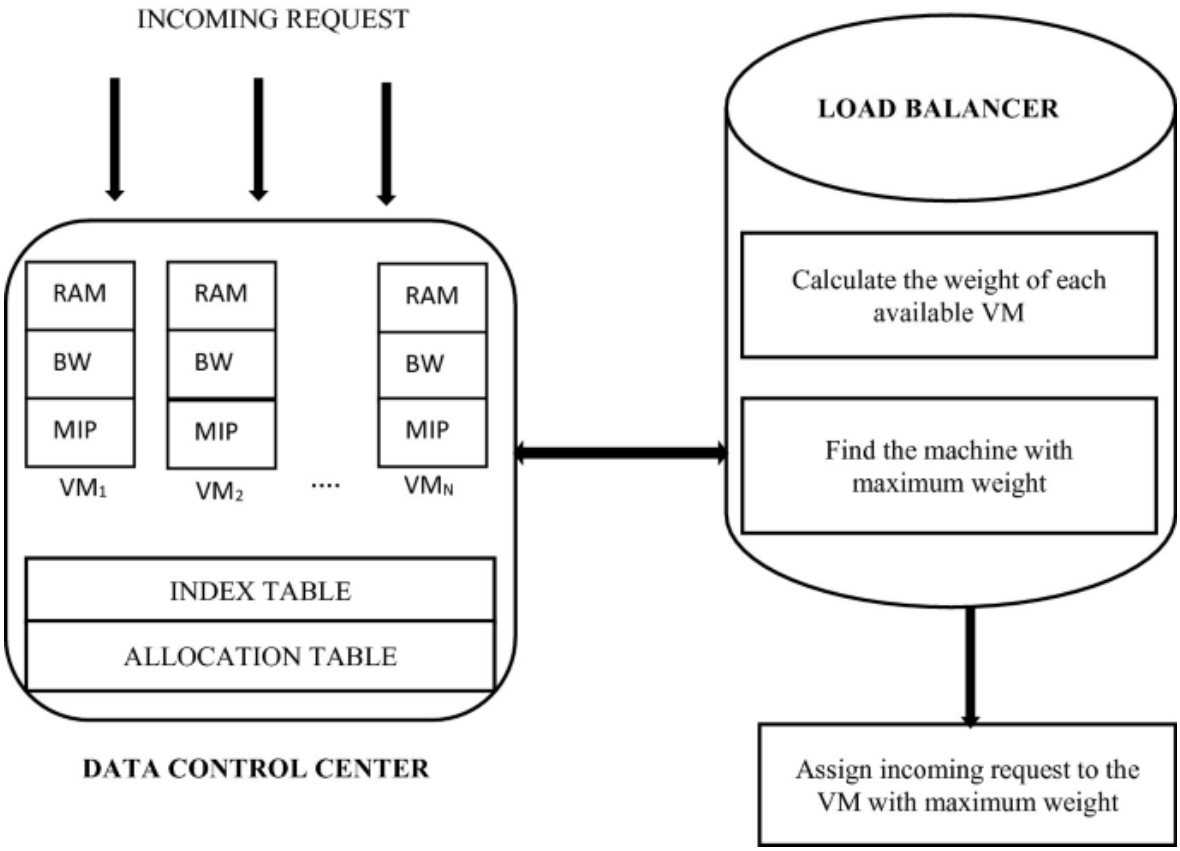
Şekil 4 Least Connection Algoritması

2.6.Yük Dengeleyicinin Anatomisi

Donanımsal bir yük dengeleyici insan vücudu gibi düşünülebilir. Farklı organlar, sistemin düzgün çalışması için birlikte çalışır. Sistemde genel amaçlı kullanılan bir sunucu gibi işlemci, bellek, ağ arabirimleri, güç kaynağı ve bazı modellerde ağ işlemleri için özelleştirilmiş ASIC üniteleri (NPU) bulunmaktadır [34]. NPU'lar, geleneksel CPU'lara ve GPU'lara kıyasla daha yüksek performans sunmaktadır. Bu da ağ işlemlerinin daha hızlı ve daha verimli bir şekilde yapılmasını sağlar [35]. NPU'ların ağ işleme birimi olarak kullanımı, ağların artan karmaşıklığı ve bant genişliği gereksinimleri ile gelecekte daha da yaygınlaşması beklenen bir teknolojidir. NPU'nun sunduğu performans, gecikme süresi ve güç tüketimi açısından avantajlar, onu ağ cihazları için ideal bir seçim haline getirmektedir

[36]. Kullanılan işlemciler ise sunucularda kullanılan işlemcilere benzer olarak genellikle Xeon ailesinden seçilmektedir. Yük dengeleyicilerde değişken ağ trafiğini kaldırabilmesi için 512gb'ye kadar çıkabilen DDR5 bellekler kullanılmaktadır. Ağ birimi olarak 100 Gbit'e kadar kapasitesi olan ara yüzler kullanılmaktadır. Şekil 5'te bir yük dengeleyicinin anatomisi gösterilmiştir.

Fiziksel yük dengeleme cihazları, asıl görevleri olan yük dengelemenin yanında güvenlik, veri optimizasyonu, ön bellekleme gibi görevleri de icra edebilirler. Bir yük dengeleyici konfigüre edilmesi durumunda güvenlik duvarı gibi çalışarak belirli protokol ve portlara göre gelen ve giden trafiği filtreleyebilir. Oturum durumunu tutarak bağlantıların durumunu izleyebilir ve izinsiz girişleri engelleyebilir. Ayrıca bazı yük dengeleyicilerin Yeni Nesil Güvenlik Duvarı (NGFW) özellikleri bulunmaktadır ve bu sayede sadece ağ katmanında (L4) olan DDoS saldırılarını engellemekle kalmayıp ayrıca uygulama katmanındaki (L7) saldırıları da engelleyebilir.[37] Aynı zamanda SQL enjeksiyonları ve siteler arası komut dosyası çalıştırma (XSS) gibi web uygulama saldırılarına karşı koruma sağlar [38]. Ayrıca belirli coğrafi bölgelerden gelen trafiği engelleme ve izin verme imanı da tanır.



***Şekil 5** Load Balancer Anatomisi*

2.7. Üretim Ortamında Kullanılan Çeşitli Yük Dengeleyiciler

Üretim ortamlarında, yük dengeleyiciler işletmelerin sürekli artan veri trafiğini yönetmede önemli rol oynar. Bu cihazlar ve yazılımlar, yüksek erişilebilirlik, mükemmel performans ve güvenlik sağlayarak, web uygulamaları, veritabanı sistemleri ve bulut tabanlı hizmetler gibi kritik iş uygulamalarının kesintisiz çalışmasını destekler. Piyasada F5 BIG-IP, Citrix NetScaler ve KEMP LoadMaster gibi çeşitli ürünler bulunmaktadır. F5 BIG-IP, karmaşık yük dengeleme ihtiyaçlarını karşılayabilen gelişmiş özelliklere sahiptir ve yüksek trafikli web sitelerinde ve global uygulamalarda tercih edilir. Citrix NetScaler, özellikle sanallaştırma ve bulut teknolojileriyle entegrasyon konusunda güçlüdür ve hem L4 hem de L7 yük dengeleme kabiliyetine sahiptir. KEMP LoadMaster ise maliyet etkin çözümleriyle öne çıkar ve özellikle KOBİ'ler tarafından tercih edilen bir çözümdür. Bu sistemler, modern iş yüklerinin gereksinimleri doğrultusunda ölçeklenebilir ve yönetilebilir yapılarıyla dikkat çeker ve bu sayede işletmelerin sürekli değişen teknoloji ortamına uyum sağlamasına olanak tanır. Her bir yük dengeleyici model, belirli senaryolara ve ihtiyaçlara göre özelleştirilebilir yapıda tasarlanmıştır, bu da kullanıcıların çeşitli uygulama senaryolarında maksimum fayda sağlamasına yardımcı olur.

2.7.1. F5 BIG-IP

F5 BIG-IP uygulamaların ve ağ altyapısının güvenliğini, performansını ve kullanılabilirliğini optimize etmeye yardımcı olan bir uygulama teslim denetleyicisidir. Çeşitli donanımlarda çalışmasının yanında sanal platformlarda da çalışabilir. BIG-IP yük dengeleme özelliğinin yanında yüksek ulaşılabilirlik (High Availability) ve güvenlik özellikleri de sağlar. Güvenlik duvarı, saldırı tespiti, tehdit engelleme ve uygulama katmanı güvenliği (WAF) özellikleri sağlayan bu cihazlar çeşitli sektörler ve kuruluşlarda oldukça geniş bir yelpazede kullanılmaktadır. Şekil 6'da F5 BIG-IP ile private cloud bir mimarinin orkestrasyonu gösterilmiştir.

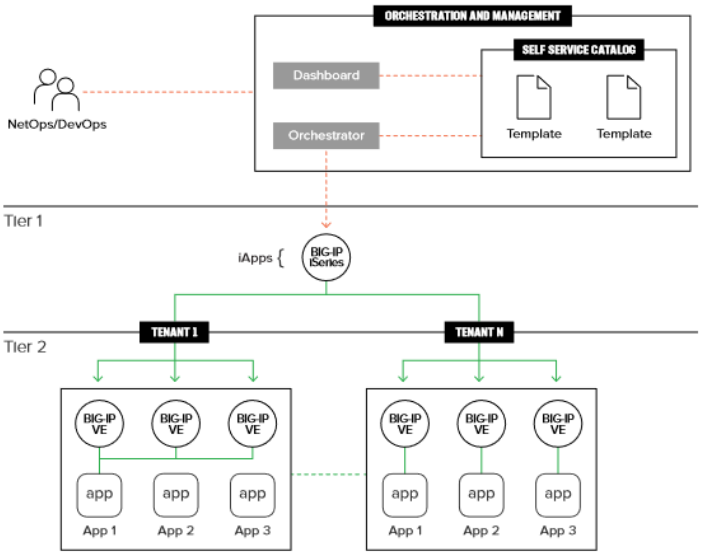
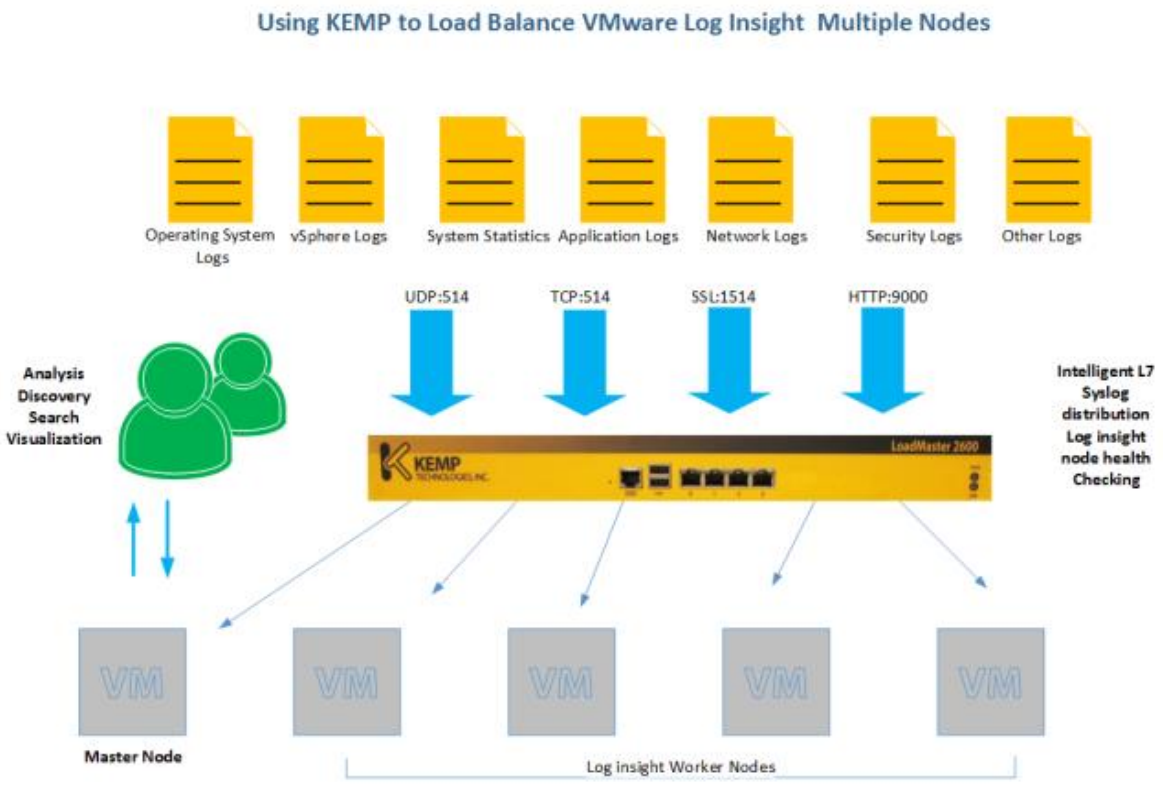


Figure 1: Orchestrated and automated deployment of app services in a two-tier private cloud architecture.

Şekil 6 *BIG-IP LB ile Orkestrasyon*

2.7.2. KEMP LoadMaster

Kemp LoadMaster ücretsiz bir sürüm sunmasıyla diğer kurumsal yük dengeleyiciler arasında öne çıkan bir çözümdür. Test ve kişisel kullanım için 20Mbit hız destekleyen bir sürümü ve işletmelere hitap eden ücretli bir sürüm bulunmaktadır. KEMP LoadMaster'ın teknik özellikleri ve avantajları oldukça çeşitlidir. Platformun temel özellikleri, akıllı trafik yönlendirme algoritmaları, güvenlik duvarları, SSL şifreleme ve saldırı tespit sistemleri entegrasyonu gibi özelliklerle donatılmıştır. Bu özellikler, işletmelerin verimliliğini artırırken, güvenlik ve kesintisizlik sağlar. KEMP LoadMaster'in kullanımı işletmelere çeşitli faydalar sağlamaktadır. E-ticaret siteleri, CDN platformları, yük dengeleyicilerin sıklıkla kullanıldığı senaryolara örnek verilebilir [39]. Bu gibi senaryolarda Kemp LoadMaster gibi yük dengeleyicilerin sağladığı güvenlik, hız ve esneklik işletmelerin çevrimiçi varlıklarını etkili bir şekilde yönetmelerini sağlar. Platformun teknik özellikleri ve sağladığı güvenlik önlemleri işletmeler için kritik bir bileşen olmasında öne çıkan sebeplerden biridir. Şekil 7'de Kemp LoadMaster ile logların yönetimi gösterilmiştir.



Şekil 7 *Kemp LoadMaster ile VMware Loglarının Yönetimi*

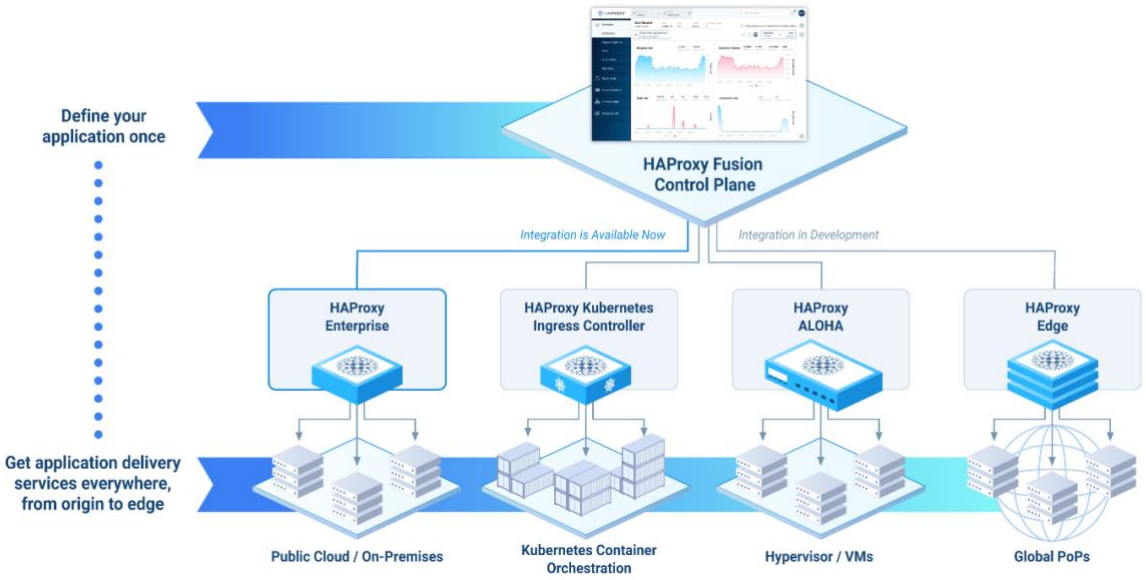
2.7.3. HAProxy

HAProxy, yüksek kullanılabilirlik, yük dengeleme ve TCP ve HTTP tabanlı uygulamalar için proxy hizmetleri sunar. TCP ve HTTP tabanlı uygulamalar için yük dengelemesi ve yüksek kullanılabilirlik sağlar [40]. HA (High availability) ve L7 katmanında çalışabilme gibi özelliklerinden ötürü tercih edilmektedir.

2.7.3.1. HAProxy Kullanım Senaryosu

Bir e-ticaret platformumuz olduğunu düşünelim ve bu platform, kullanıcı arayüzü, ürün katalogu, sepet hizmeti ve ödeme hizmeti gibi çeşitli mikro hizmetlerden oluşsun. Bu mimaride her bir mikroservis, kendi içinde yatay olarak ölçeklendirilebilir ve her biri kendi veritabanına sahip olabilir. Bu durumda gelen istekleri uygun mikroservise yönlendirmeyi HAProxy yapacaktır. Ayrıca HAProxy, her bir servisin sağlığını düzenli olarak kontrol eder ve bir hizmet kullanılamaz hale gelirse, trafiği otomatik olarak diğer hizmetlere yönlendirir. Bu şekilde yüksek erişilebilirlik sağlanmış olur. HAProxy'nin sağladığı istatistiklerle

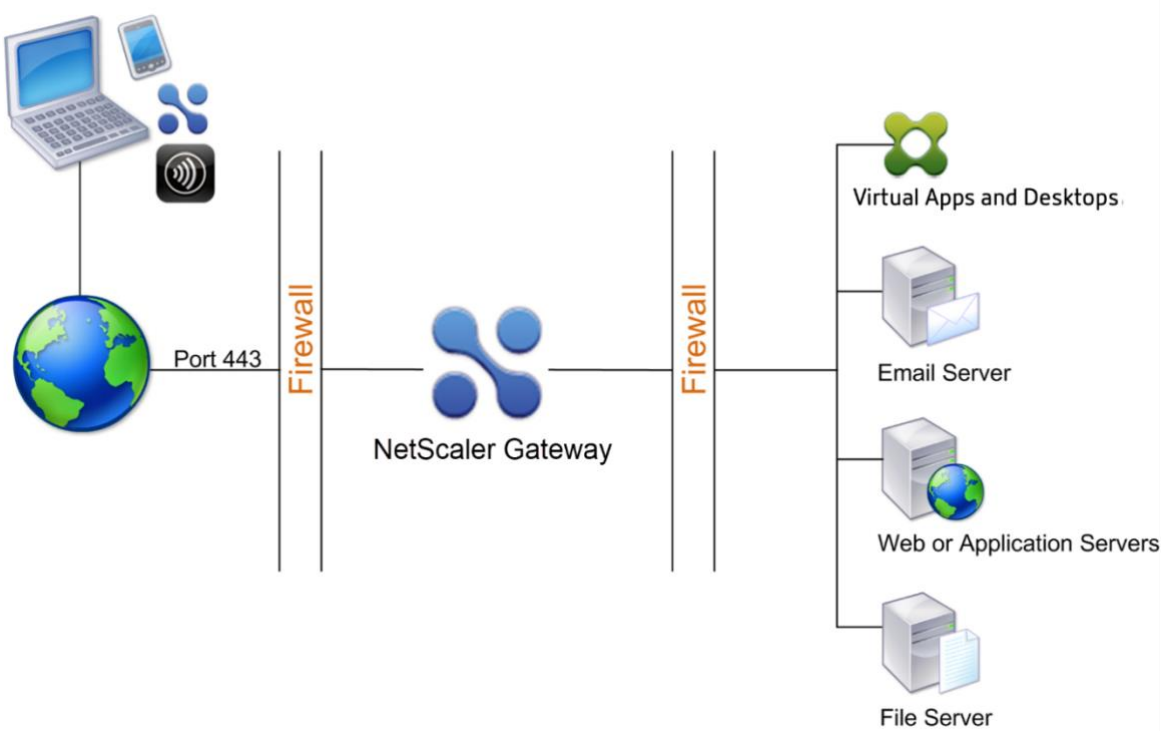
performans optimize edilebilir ve bu istatistikler ölçeklendirme için planlama aşamasında oldukça bilgi sağlayıcıdır [41]. Şekil 8'de HAProxy Fusion ile kurulan bir orkestrasyon altyapısının şeması gösterilmiştir.



Şekil 8 *HAProxy Fusion'un Control Plane Yapısı*

2.7.4. Citrix NetScaler

Citrix NetScaler Load Balancer, web uygulamalarınızın en üst düzeyde performans ve güvenlikle çalışmasını sağlayan bütünleşmiş bir çözümdür. L4-7 katmanlarında trafik yönetimi ve uygulama sürekliliği sunarak, web ve veri tabanı sunucularınızın yükünü optimize eder ve kesintisiz çalışma imkânı sağlar [34]. Ayrıca SYN Flood, TCP Flood, HTTP Flood ve DNS Flood gibi DDoS saldırılarını etkin bir şekilde engeller. Şekil 9'da Netscaler Gateway'in ağ topolojisinde nasıl kullanıldığı gösterilmiştir.



Şekil 9 Citrix Netscaler Gateway Yapısı

2.8.Çeşitli DDoS Saldırıları

DDoS (Distributed Denial of Service) saldırıları, hedeflenen sistemin, hizmetin veya ağın kaynaklarını aşırı yükleyerek erişilemez hale getirmek amacıyla gerçekleştirilir. Bu saldırılar volumetrik, protokol ve uygulama katmanı olmak üzere üç ana kategoriye ayrılır. Volumetrik saldırılar büyük miktarda trafik göndererek bant genişliğini tüketirken, protokol saldırıları TCP/IP zafiyetlerini kullanır ve SYN flood gibi teknikler içerir. Uygulama katmanı saldırıları ise web uygulamalarını hedef alır ve genellikle HTTP GET/POST flood yöntemlerini kullanır.

2.8.1. SYN Flood Saldırıları

DoS saldırılarının en ilkel yolu olan ping floodun gelişmiş bir versiyon SYN Flood mevcut sunucu kaynaklarını tüketerek trafiği engellemeyi amaçlamaktadır [42].

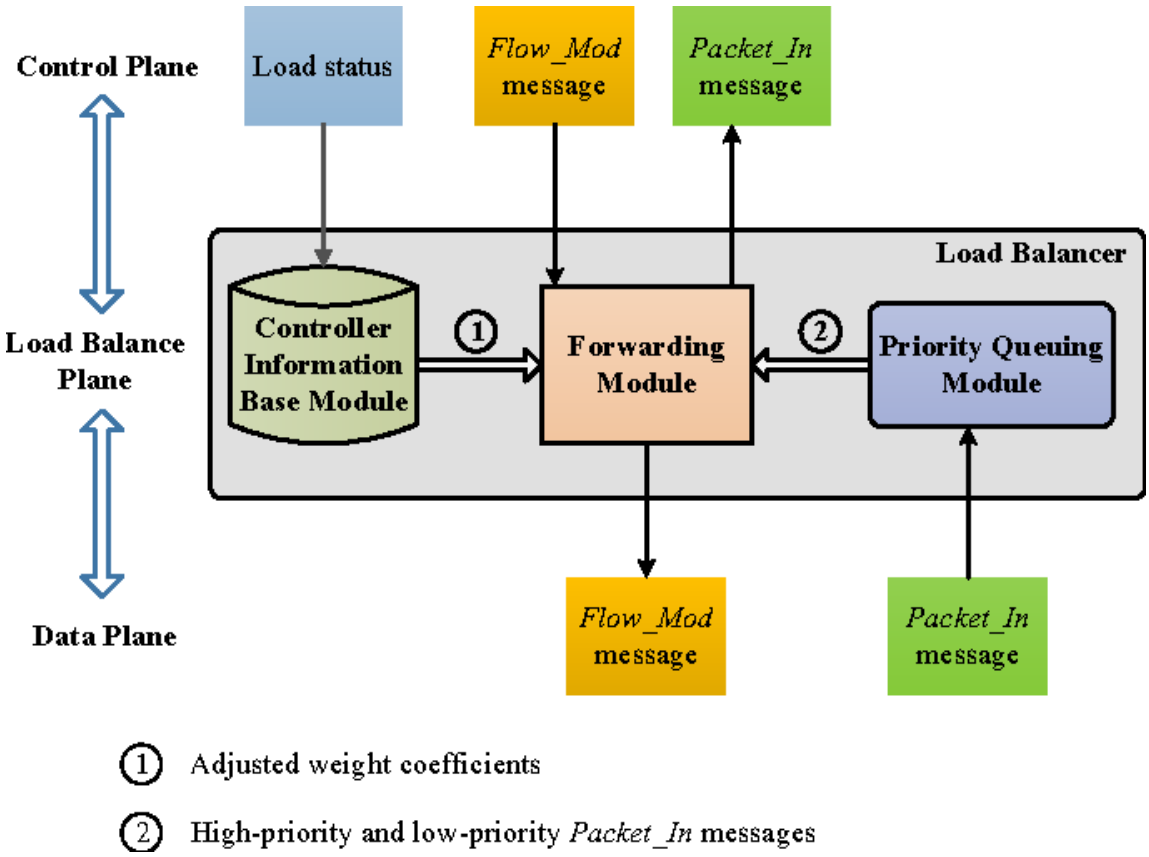
TCP protokolü kullanılarak istemci ve sunucu arasında bir oturum başlatıldığında, oturumu oluşturan “üçlü el sıkışması” iletileri sunucu belleğinde küçük bir arabellek alanı ayrılır. Oturumu oluşturan paketler, ileti alışverişindeki diziyi tanımlayan bir SYN alanı içerir. SYN Flood saldırıları ise bu süreci bozarak trafiği aksatır [43].

2.8.2. HTTP Flood Saldırıları

HTTP Flood L7'de gerçekleştirilen DDoS saldırılarının özel bir biçimidir. Bu saldırı istemcinin GET veya POST isteğine dayanır. Bir istemci – diğer bir deyişle, web sitesini çağıran tarayıcı – bu isteklerden birini gönderdiğinde, sunucu isteği işler ve sonucu istemciye geri gönderir. Sunucu kapasitesini aşan bir seviyede istek geldiği durumda, kaynaklar tükenerek trafiği engeller [44].

2.9.Load-Balancer Blok Yapısı

Yük dengeleyicisi blok diyagramı, bir yük dengeleyicinin farklı bileşenlerini ve bunların nasıl etkileşime girdiğini gösteren görsel bir temsilidir. Şekil 10'da temel bir yük dengeleyicisi blok diyagramı ve bileşenlerinin işlevleri yer almaktadır.



Şekil 10 Load Balancer Blok Yapısı

Control plane, ağdaki routerlar ve switchler gibi ağ cihazlarının yapılandırılması ve yönetilmesinden sorumludur. Control plane, OpenFlow protokolü gibi bir protokol aracılığıyla ağ cihazlarıyla iletişim kurar [45]. Data plane ise paketlerin ağ üzerinden yönlendirilmesinden sorumludur. Sunucu ise istekleri işler ve kullanıcılara yanıtlar gönderir.

Akış diyagramında bir switch veya routerin hangi paketleri nereye yönlendireceğine dair karar verme mekanizmasını görebiliriz.

Flow_mod mesajı, OpenFlow protokolünde bir switch veya routerin akış tablosunu güncellemek için kullanılır [46]. Akış tablosu, bir anahtar veya yönlendiricinin hangi paketleri nereye yönlendireceğine dair kuralların bir listesidir.

2.9.1. Load-Balancer Katmanları

Load-balancer katmanları, ağ trafiğini verimli yönetmek için control plane, data plane ve load-balancing plane olarak üçe ayrılır. Control plane, yük dengeleyicinin trafiği nasıl yönlendireceğini belirler ve yapılandırma, yönlendirme kararları gibi yönetim işlevlerini içerir. Data plane, gelen ve giden trafiğin gerçek zamanlı olarak işlenmesi ve yönlendirilmesi ile ilgilenir, yani trafiğin akışını sağlar. Load balancer plane'in temel görevi ise üzerindeki trafik yükünü optimize etmek ve dağıtmaktır.

2.9.1.1. Control Plane

Kontrol katmanı bir ağ veya iletişim sistemi içindeki cihazların ve kaynakların yönetiminden sorumlu olan bileşendir. Bu katman, ağdaki trafiği yönlendirme, ağdaki cihazların durumunu izleme, yönetme ve high-available olan cihazları failover etme gibi işlevleri gerçekleştirir. Kontrol düzlemi, ağdaki diğer cihazlarla iletişim kurarak ağ durumunu izler, ağ trafiğini yönetir ve gerektiğinde değişiklikler yapar [47]. Örneğin, bir yönlendirici kontrol düzlemi, ağdaki diğer yönlendiricilerle iletişim kurarak en kısa rotaları hesaplar ve trafiği yönlendirir. Ayrıca, bir routerdaki kontrol düzlemi, ağdaki diğer routerlarla iletişim kurarak en kısa rotaları hesaplar ve trafiği yönlendirir. Kontrol düzlemi, ağdaki cihazların yönetiminden sorumlu iken, veri düzlemi ise gerçek veri iletiminden sorumludur. Veri düzlemi, ağ üzerinde gerçekleşen veri aktarımlarını sağlar ve kontrol düzleminden aldığı talimatlar doğrultusunda trafiği yönlendirir. Bu iki düzlem, birbirinden bağımsız olarak çalışır ve farklı işlevlere sahiptir. Kontrol düzlemi, ağın etkin ve güvenli bir şekilde çalışmasını sağlar. Sonuç olarak kontrol düzlemi ağ ve iletişim sistemlerinin temel bir bileşeni olarak görev aldığından oldukça önemlidir.

2.9.1.2.Load Balancer Plane

Load balancer katmanı, modern internet altyapısında vazgeçilmez bir rol üstlenmektedir. Temel olarak görevi ağ üzerindeki trafik yükünü optimize etmek ve dağıtmaktır. Bu tek bir sunucunun veya cihazın üzerindeki yükü azaltıp sistem genelinde verimliliği arttırır.

2.9.1.3. Data Plane

Ağ mimarisinde başka bir önemli katman da veri katmanıdır. Veri katmanı kullanıcıların gönderdiği ve aldığı verilerin gönderilmesinden sorumludur ve iletimin gerçekleştiği katmandır.

Veri katmanı, kullanıcıların aldığı ve gönderdiği verileri yönlendirir ve iletimini sağlar [48]. Her veri paketi, hedefine ulaşmak için en uygun rotayı bulmak için routing table'lar ve başka kaynaklar kullanır. Veri düzlemi, ağdaki veri paketlerini hedeflerine doğru yönlendirmek için çeşitli yönlendirme protokolleri kullanır. Örneğin, Internet Protokolü (IP) adreslerine dayalı yönlendirme, en yaygın kullanılan yönlendirme protokollerinden biridir.

Anahtarlamalı ağlar, veri paketlerini hedeflerine doğrudan iletmek için anahtarlamalı anahtarlar kullanır. Ethernet ve MPLS gibi protokoller, anahtarlamalı ağların temelini oluşturur. Paketler küçük parçalara bölünür ve ağ üzerinde en uygun rota kullanarak iletilir.

2.10. DNS Yük Dengeleme

TCP/UDP gibi L4 katmanında yük dengelemesinin ve L7'de yapılan HTTP yük dengelemesinden bahsettik. Bunların yanında DNS için de yük dengeleme yapılabilmektedir.

DNS (Domain Name System), internet üzerindeki alan adlarını IP adreslerine çeviren bir sistemdir. DNS yük dengelemesi, bu sistemi kullanarak gelen talepleri yönlendirerek, bir web sitesinin veya uygulamanın farklı sunucular arasında dengeli bir şekilde dağıtılmasını sağlar [49]. Bu, sunucu yükünü dengeleyerek, tek bir sunucunun aşırı yük altında ezilmesini engeller ve kullanıcıların hızlı yanıt almasını sağlar. Ayrıca, DNS yük dengelemesi, yedekleme sunucularını devreye alarak yüksek kullanılabilirlik ve kesintisiz erişim sağlar.

2.10.1. DNS Yük Dengelemenin Endüstrideki Kullanımı

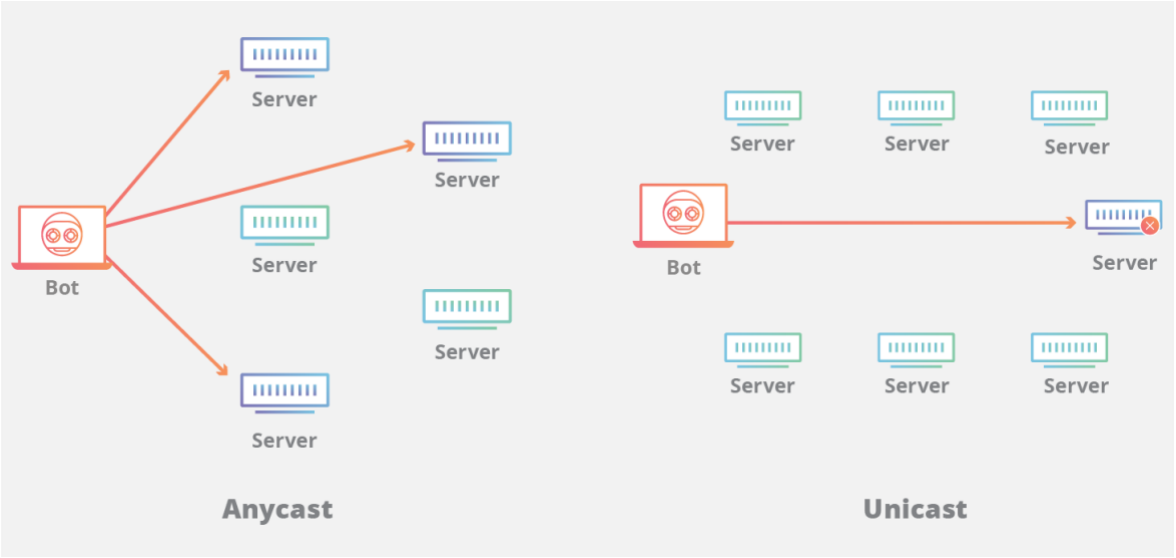
DNS yük dengeleme, endüstride ağ trafiğini optimize etmek ve hizmet sürekliliğini sağlamak için yaygın olarak kullanılır. Bu yöntemle, birden fazla sunucu arasındaki istekler çeşitli algoritmalar kullanılarak dağıtılıp sistem performansının arttırılması hedeflenir. Küresel ölçekte hizmet veren servislerde, kullanıcılar coğrafi olarak en yakın veya en az yoğun sunucuya yönlendirilerek gecikmenin en aza indirilmesi hedeflenir. Bu servislere örnek olarak içerik dağıtım ağları (CDN'ler), büyük web siteleri ve bulut hizmetleri verilebilir.

2.10.2. CDN Servisleri

Günümüzde CDN'ler (içerik dağıtım ağları) web sitelerinin ve uygulamaların performansını arttırmak için kurulmuş özel veri merkezleridir. Dünya üzerinde coğrafi olarak stratejik lokasyonlara kurulmuş olup veri gecikmesini minimuma düşürmeyi hedefleyen bu veri merkezleri, bağlantıları optimize etmek için çeşitli stratejiler kullanmaktadır [50]. Bunlardan biri de DNS yük dengelemedir.

2.10.3. Anycast İle Dağıtılmış Sunucular

Cloudflare ve benzeri CDN'ler, dünya genelinde birçok noktada bulunan sunucu ağlarına sahiptir. Bu sunucular, aynı IP adresiyle hizmet verirler ve Anycast protokolü kullanılarak trafiği optimize ederler. Kullanıcı bir DNS sorgusu yaparken, Anycast ile, bu sorgu en yakın ve en uygun sunucuya yönlendirilir [51]. Şekil 11'de sunucularda anycast adreslerin kullanımıyla ilgili bir diyagrama yer verilmiştir.



Şekil 11 Sunucularda Anycast Adreslerin Kullanımı

2.10.4. Akıllı Yük Dengeleme Algoritmaları

Cloudflare, kullanıcı taleplerini yönlendirmek için akıllı yük dengeleme algoritmaları kullanır. Bu algoritmalar, kullanıcının coğrafi konumunu, sunucu yükünü ve ağ koşullarını dikkate alarak, en uygun sunucuya yönlendirme yaparlar [52]. Böylece, kullanıcılar her zaman hızlı yanıt alır ve sunucu kaynakları verimli bir şekilde kullanılır.

2.10.5. Otomatik Hata Tespiti ve Yedekleme

Cloudflare gibi CDN servisleri, sunucu hatalarını otomatik olarak tespit eder ve etkilenen sunucuları geçici olarak devre dışı bırakır. Bu süreçte, yedek sunucular devreye alınarak kesintisiz erişim sağlanır. Bu, yüksek kullanılabilirlik ve hizmet kesintisi olmadan sürekli bir hizmet sunmayı mümkün kılar.

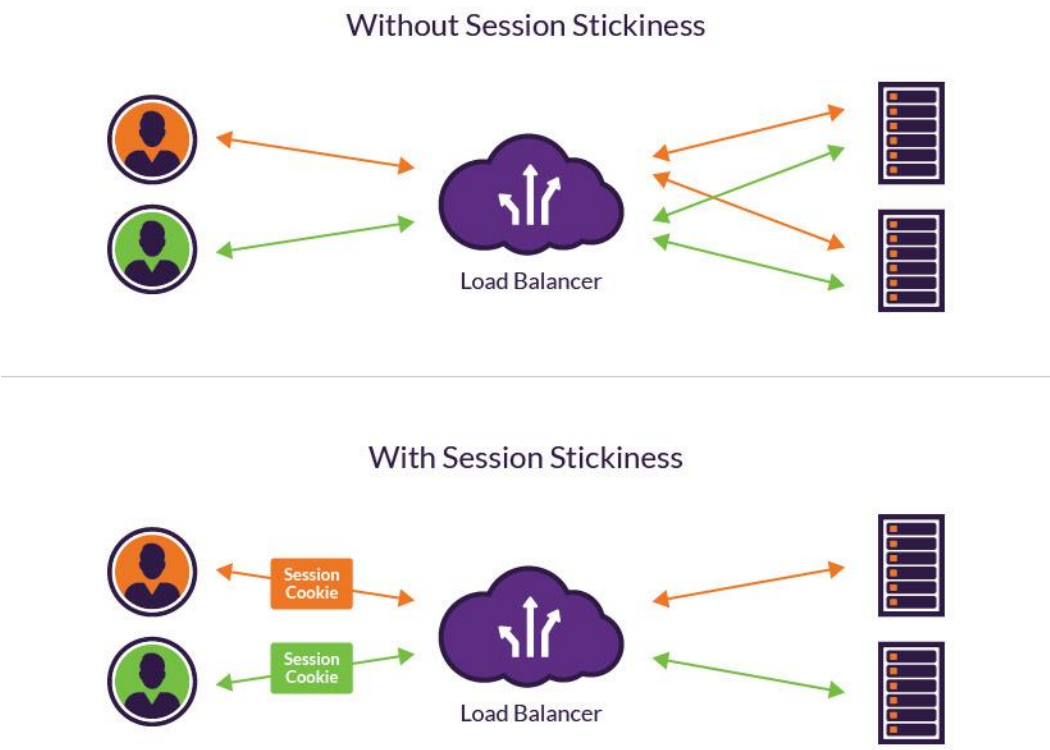
2.10.6. En Yakın Sunucu Seçimi

Anycast protokolü, kullanıcının talebine en yakın olan sunucuyu seçmek için BGP (Border Gateway Protocol) gibi yönlendirme protokollerini kullanır [53]. Bu, kullanıcıya en iyi erişim hızını sağlamak için önemlidir.

2.10.7. Session Affinity / Session Stickiness

Session affinity (oturum bağlılığı), ağ yük dengelemesinde kilit noktalardan biridir ve kullanıcının açtığı oturum boyunca aynı sunucuya yönlendirilmesini ifade eden bir

kavramdır [54]. Session affinity, genellikle yük dengeleyici (load balancer) gibi ağ altyapı bileşenleri tarafından uygulanır. Şekil 12'de session affinity'nin şematik gösterimine yer verilmiştir.



Şekil 12 Session Stickiness'in Şematik Gösterimi

Session affinity'nin uygulanması için farklı yöntemler vardır. En sık kullanılanları ise IP bazlı session affinity ve Cookie bazlı session affinitydir.

2.10.8. IP bazlı session affinity

Kullanıcının IP adresine dayalı olarak bir oturuma yönlendirilmesini sağlar. Bu, kullanıcının aynı cihazdan gelen tüm isteklerinin aynı sunucuya yönlendirilmesini sağlar. Ancak, aynı ağ arkasında birden fazla kullanıcı varsa veya IP adresi değişirse bu yöntem etkisiz olabilir.

2.10.9. Cookie bazlı session affinity

Bu yöntemde, kullanıcının tarayıcısına özel bir oturum kimliği (session ID) oluşturulur ve bu kimlik sunucuda saklanır veya tarayıcıya bir çerez olarak gönderilir. Kullanıcının

tarayıcısı bu çerezi saklar ve her istekte sunucuya gönderir [55]. Böylece, kullanıcının oturumu boyunca aynı sunucuya yönlendirilmesi sağlanır.

Session affinity kullanımı, web uygulamalarının gereksinimlerine göre değişir. Bazı senaryolarda, session affinity önemli bir gereksinimken bazı senaryolarda uygulamanın dağıtık doğası ve yüksek ölçeklebilirlik (HA) gereksinimleri sebebiyle session affinity kullanılmaz.

2.10.10. Load Balancer'da Health Checking

Yük dengeleyicilerin etkin çalışabilmesi için arkalarındaki sunucuların sağlığını sürekli olarak izlemek ve yönetmek gerekmektedir.

Health check, bir load balancerın, arkasındaki sunucuların sağlık durumunu izlemek için gerçekleştirdiği düzenli testlerdir. Bu testler, sunucuların çalışma durumunu, yanıt sürelerini ve erişilebilirliklerini değerlendirir [56]. Eğer bir sunucu sağlık kontrolünden geçemezse, load balancer, o sunucuya yönlendirilen trafiği durdurabilir ve sağlıklı sunuculara yönlendirebilir. Bu sayede, kullanıcılar kesinti yaşamadan hizmet almaya devam edebilirler.

2.10.11. TCP Health Check

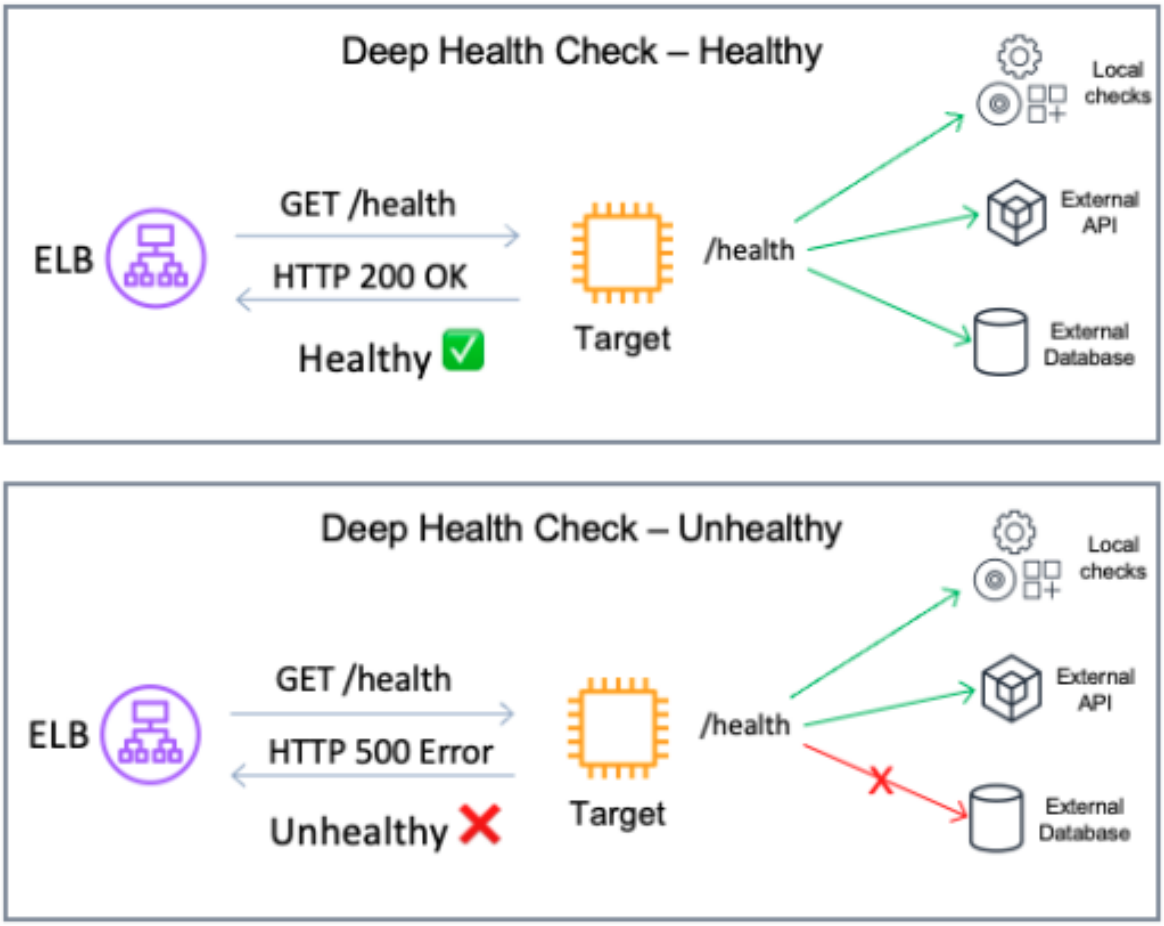
Bu sağlık kontrolünde, load balancer sunucularla TCP protokolüyle bağlantı kurarak, sunucunun erişilebilir olup olmadığını kontrol eder. Eğer sunucu belirli bir süre içinde yanıt vermezse, sağlık kontrolü başarısız olur.

2.10.12. UDP Health Check

TCP'nin aksine, UDP sağlık kontrolü, bağlantı kurmadan sunucuların erişilebilirliğini kontrol eder. Bu, özellikle UDP tabanlı servisler için önemlidir.

2.10.13. HTTP Health Check

Bu sağlık kontrolü, HTTP veya HTTPS protokollerini kullanarak sunuculara belirli bir URL'yi çağırır ve beklenen yanıt kodunu alıp alamadığını kontrol eder. Bu sayede, sunucunun hizmet verip vermediği ve beklenen içeriğin alınıp alınmadığı belirlenebilir. Şekil 13’te yük dengeleyici üzerinde health check uygulamasının şematik gösterimine yer verilmiştir.



Şekil 13 LB'de Health Check'in Şematik Gösterimi

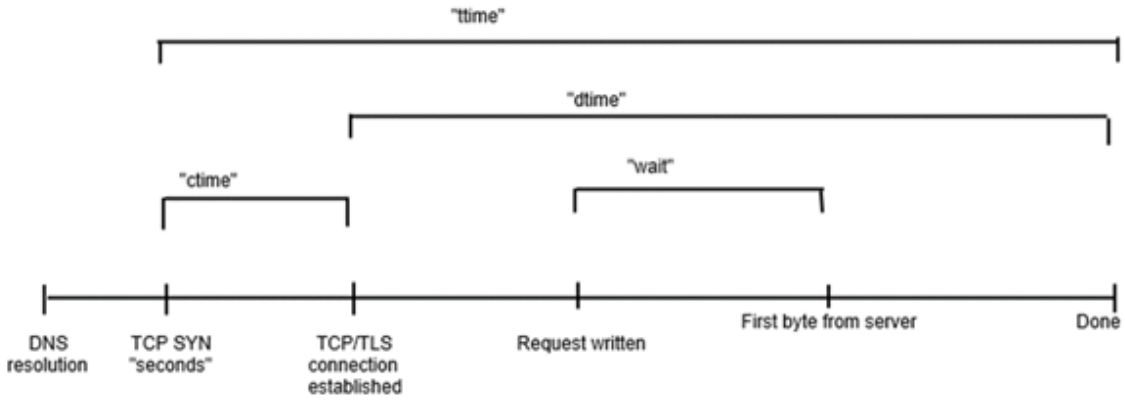
2.11. Apache Benchmark

Web sunucularının performansını değerlendirmek, kullanıcı deneyimini iyileştirmek için önemli hususlardan biridir. Bu bağlamda Apache Benchmark (ab) sıklıkla kullanılan araçlardan biridir. Komut satırında çalışan bir araç olan Apache Benchmark, belirli bir sunucuya paralel istekler göndererek sunucunun yanıt süresini, istek başına geçen ortalama süreyi vb. performans metriklerini ölçmeyi mümkün kılar.[57] Şekil 15'te bahsedilen performans metriklerinin şematik gösterimi verilmiştir. Kullanıcılar, istenen URL'yi ve istek sayısını belirterek bir test senaryosu tanımlarlar. Ardından, Apache Benchmark belirtilen istek sayısını belirtilen URL'ye gönderir ve sunucudan aldığı yanıtları ölçer. Sonuçlar, kullanıcıya sunucunun performansı hakkında değerli bilgiler sağlar.

Apache Benchmark, çoklu iş parçacıklarından (threads) faydalanarak web sunucusuna bağlantılar kurar ve yanıtları analiz eder. Şekil 14'te bahsedilen thread yapsının şeması

gösterilmiştir. E-ticaret siteleri gibi oluşumlar indirim dönemindeki yüksek kullanıcı senaryosunu test etmek için yük testlerine sıklıkla başvurmaktadırlar. Elde edilen veriler ışığında yük yönetimi optimize edilebilir ve kaynaklardan tasarruf edilebilir.

Şekil 14 Apache Benchmark Thread Yapısı



Şekil 15 Apache Benchmark Parametreleri

- **starttime** – İsteğin yapıldığı tarih ve saat
- **seconds** − İsteğin yapıldığı tarih ve saatin Unix timestamp olarak formatı
- **ctime** – Bağlantı süresi
- **dtime** – İşleme süresi
- **ttime** – Toplam süre (Bağlantı süresi ve işleme süresinin toplamı)
- **wait** – Bekleme süresi

2.12. Gnuplot

Veri görselleştirilmesi, elde edilen verinin anlamlandırılması için önemli bir adımdır. Doğru araçlar kullanılarak yapıldığında verilerin anlaşılmasını kolaylaştırır ve bilgi keşfini hızlandırır. Apachebench'te elde edilen verilerin yorumlanması için Gnuplot kullanılacaktır.

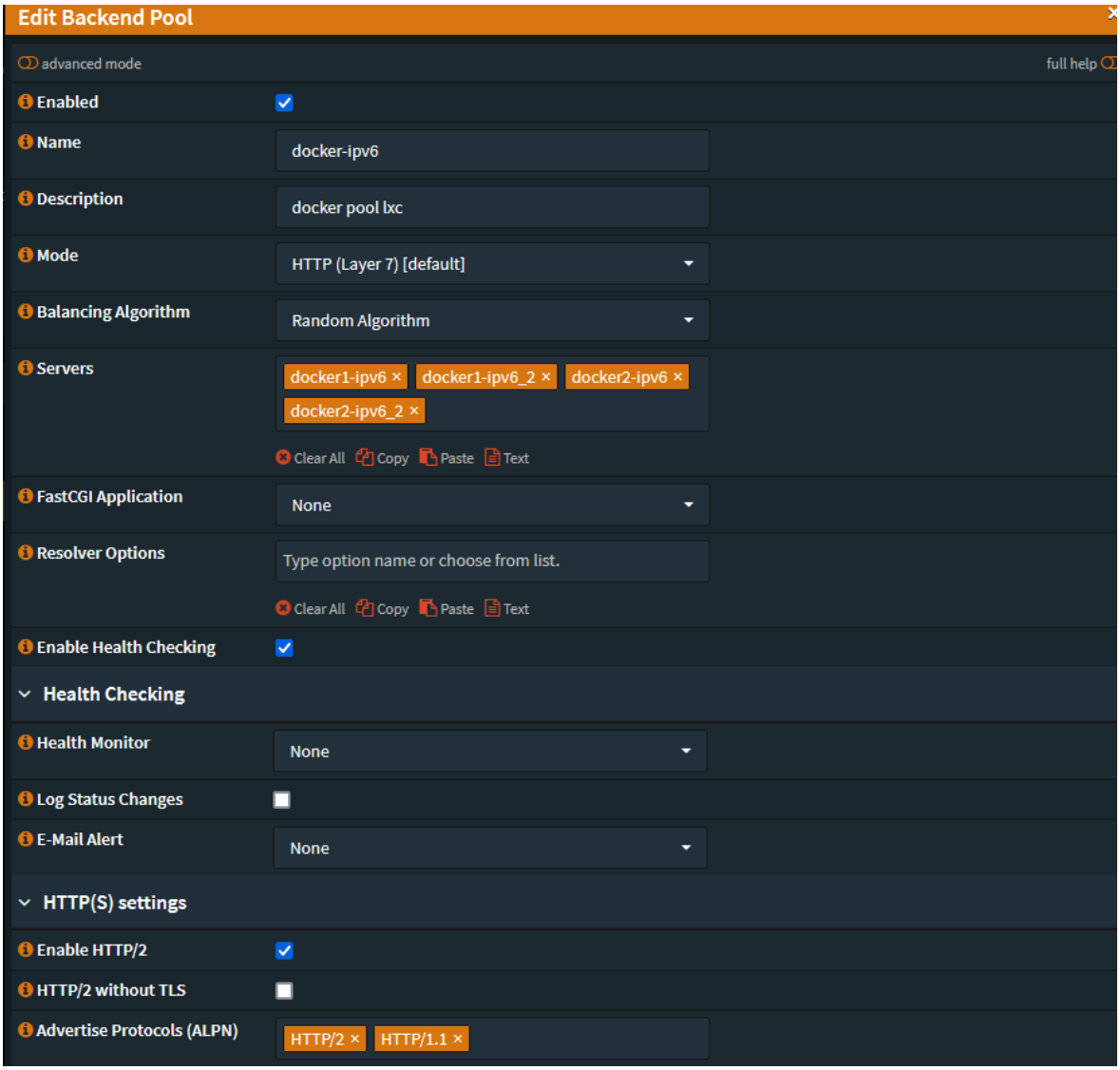
Komut satırı tabanlı bir araç olan Gnuplot, veri setlerini grafikler, grafikler, 3B yüzeyler ve daha fazlası gibi çeşitli grafiksel biçimlerde görselleştirmek için kullanılmaktadır. Ayrıca Gnuplot istatiksel verilerin analizi için de kullanılabilir [58]. Bu çalışmada Gnuplot, Apache Benchmark'tan elde edilen .csv formatındaki dosyaların grafiğe dökülmesinde kullanılmıştır.

Bölüm 3 – BULGULAR VE TARTIŞMA

Bu bölümde, daha önceki bölümlerde anlatılan yük dengeleme algoritmalarının etkinlikleri test edilerek değerlendirilmiştir. Testler, Proxmox Hypervisor altında kurulu HAproxy load balancer ve özellikle Proof of Concept (POC) amaçlı hazırlanan Docker konteynerleri üzerinde yürütüldü. Yük testleri için Apache Benchmark (ab) aracı kullanıldı ve bu süreçte ölçüm parametreleri olarak bağlantı süresi (connection time), işleme süresi (processing time) ve bekleme süresi (waiting time) belirlendi.

Bu testler sırasında, farklı yük dengeleme algoritmalarının trafik performansı ve verimlilik üzerine etkileri kapsamlı bir şekilde incelendi. HAproxy arayüzü, kullanılan algoritmaların yanı sıra L4 ve L7 ağ protokol katmanlarında, SSL/TLS şifreleme suitleri ve HTTP protokolü versiyonlarının seçimine imkân tanıdı. Ayrıca, algoritmaların trafiği yönetme kapasitesi ve verimlilik oranları, sistemin genel performansı üzerindeki etkileri ile analiz edildi. Bahsedilen parametrelerin bulunduğu ekranın görüntüsü Şekil 16'da paylaşılmıştır.

Sonuçlar, belirlenen algoritmaların farklı ağ yapıları ve trafik yoğunlukları altında nasıl performans gösterdiğini açıkça ortaya koymuştur. Özellikle, yüksek trafikli ortamlarda ve çeşitli uygulama senaryolarında algoritmaların dayanıklılığı ve yönetim kabiliyetleri değerlendirilmiştir. Bu bulgular, yük dengeleme teknolojilerinin ve stratejilerinin, modern ağ altyapılarında nasıl optimize edilebileceği konusunda derinlemesine bir inceleme sağlar.



Şekil 16 *HAproxy Backend Seçim Ekranı*

Bu bölümde, yük dengeleme sisteminde kullanılan çeşitli algoritmaların ve konfigürasyon seçeneklerinin nasıl performans gösterdiği üzerine yoğunlaşılacaktır. Deney düzeni, yük dengeleyicilerde mevcut olan farklı algoritma seçenekleri (Source-IP Hash, Round Robin, Static Round Robin, Least Connections, URI Hash ve Random Algorithm) arasından seçim yaparak, bu algoritmaların çeşitli senaryolarda nasıl bir davranış sergilediğini gözlemlemeyi amaçlamaktadır.

Bu algoritmaların performansı, farklı trafik yükleri, istek dağılımları ve sunucu kapasiteleri gibi değişkenlerle test edilerek değerlendirilmiştir. Her bir algoritmanın verimliliği, yanıt süresi ve kaynak kullanımı gibi kriterler üzerinden analiz edilmiştir. Bu incelemelerin

sonucunda, belirli durumlar için en uygun algoritmanın hangisi olduğunu belirlemeye yardımcı olmak ve yük dengeleme sistemlerinin performansını optimize etmek için veri sağlaması amaçlanmıştır.

- Balancing algorithm, Source-IP Hash, Round Robin, Static Round Robin, Least Connections, URI Hash ve Random Algorithm olarak seçilebilmektedir. Algoritmaların çeşitli senaryolar çerçevesinde nasıl performans gösterdiğini anlamak için hepsi denenecektir.
- Mode HTTP – Layer 7 veya TCP – Layer 4 olarak seçilebilmektedir.
- Prefer IP family kısmında IPv4 veya IPv6 protokolleri arasında seçim yapılabilmektedir.
- Advertise Protocols ise TLS (Transport Layer Security) protokolüyle ilgilidir. ALPN, TLS el sıkışması sırasında istemci ve sunucu arasında kullanılacak uygulama katmanı protokolünü belirlemek için kullanılır. ALPN, HTTP/2 ve HTTP/3 gibi modern HTTP protokollerinin yanı sıra diğer uygulama katmanı protokollerinin desteklenmesinde önemli bir rol oynar.

Yukarıda bahsedilen parametrelerden balancing algorithm, bu çalışmada sistemin farklı algoritmalarda nasıl davrandığını göstermek için kontrollü olarak değişmekteyken IP family seçeneği sistemi değerlendirmek için gerekli olmadığından sabit tutulmuştur.

3.1. Yük Dengeleme Algoritmalarının Değerlendirilmesi

Yüksek trafik alan mimarilerde açılan bağlantıların, sunuculara koordineli ve verimli bir şekilde iletilmesi için doğru algoritmaların seçilmesi operasyonun kesintisiz ilerlemesi için son derece önemlidir. Bu algoritmalar sunucular arasındaki trafiği belirli kriterlere göre yönlendirir. Örneğin sunucunun üzerindeki yük, erişim süresi, sunucunun coğrafi konumu ve ağ üzerindeki başka faktörler göz önünde bulundurulabilir.

Yük dengeleme algoritmaları sunucular arasında iş yükünün dengelenmesini sağlayan gelişmiş iletişim sistemlerine dayanır. Bu sistemler genellikle dağıtık bir mimariye sahiptir ve sunucular arasındaki veri iletişimini optimize etmek için çeşitli teknikler kullanır. Bu sebeple yük dengeleme sistemlerinde algoritmaların trafik performansına etkisi test edilmiştir.

Bölüm 4 – SONUÇLAR

Bu çalışma, çeşitli yük dengeleme algoritmalarının kapsamlı bir değerlendirmesini sunarak, bu algoritmaların ağ performansı üzerindeki etkilerini ortaya koymuştur. Testler, Proxmox hypervisor ve HAproxy load balancer kullanılarak gerçekleştirildi ve Apache Benchmark aracılığıyla stres testleri yapıldı. Bulgular, Round Robin ve Least Connections algoritmalarının belirli trafik yoğunluklarında önemli performans avantajları sağladığını gösterdi.

Bu çalışmada yük dengeleme performansının kıyaslanması için bahsedilen algoritmaların uygulaması yapıldı, gerekli test ortamı Proxmox hypervisor üzerinde çalışan OPNsense Firewall ve HAproxy Loadbalancer üzerinde geliştirildi. Literatürde geçen çalışmalara, ilgili platformlara ve teorik bilgilere atıfta bulunuldu. Host web sunucularının barındırıldığı platform Docker olarak seçildi. Altyapıda kullanılan kablo cinsi olarak yüksek hızları desteklediği için CAT6 Ethernet seçilmiştir. Anahtarlama işlemleri için 5 portlı PoE özelliği olan ve 1 Gbit hız destekleyen switch kullanılmıştır. Test ortamı oluşturulurken web sunucularının public internete çıkabilmesi için Hurricane Electric IPv6 Tunnelbroker kullanılmıştır. Bu sayede NAT arkasındaki sunucular, yerel ağ dışındaki hostlar tarafından da erişilebilir kılınmıştır. Çalışma ortamı hazırlanırken sanal sunucularda UDP bağlantısı olumsuz etkilendiğinden donanım hızlandırma devre dışı bırakılmıştır. Yük dengeleme platformu için HAproxy platformu, sunduğu çeşitli opsiyonlarla istenilen gereksinimleri karşıladığı için tercih edildi. Stres testi için Apache Benchmark aracı kullanıldı ve elde edilen tablo formundaki verilerin grafiğe dökülmesi için “Gnuplot” yazılımından faydalanıldı. Sistem son haliyle L7 HTTP katmanında yük dengeleme performansı odaklı test edildi, geliştirme sırasında izlenen adımlardan ve ileri çalışmalardan bahsedildi.

Sonuç olarak, yük dengeleme algoritmalarının seçimi ve konfigürasyonu, ağın performansı üzerinde belirleyici bir rol oynamaktadır. Bu çalışmanın bulguları, ağ yöneticilerine ve sistem mimarlarına, veri merkezleri ve bulut altyapıları için daha etkin yük dengeleme stratejileri geliştirmede rehberlik edebilir. Gelecekte, farklı ağ protokolleri ve karmaşık trafik desenleri üzerine yapılan araştırmalar, bu alanda daha derinlemesine bilgiler sunarak, yük dengeleme tekniklerinin daha da iyileştirilmesine olanak tanıyacaktır.

Yük dengeleme algoritmalarının performanslarının kıyaslanması, ağ sistemlerinde verimliliğin sağlanması ve erişilebilirliğin arttırılması için önemlidir. Özellikle günümüzde

internetin geldiği noktada yoğun trafik alan ağ ortamlarında, doğru bir yük dengeleyici algoritması seçimi sistem performansı ve kullanıcı deneyimi açısından kritik bir rol oynamaktadır.

Çeşitli yük dengeleyici algoritmaları, farklı özelliklere ve avantajlara sahiptir. Örneğin, Round Robin gibi basit algoritmalar, gelen istekleri sırayla sunuculara dağıtırken, Weighted Round Robin gibi gelişmiş versiyonları, sunuculara farklı ağırlıklar atayarak daha dengeli bir dağılım sağlar. Diğer yandan, Least Connections algoritması, en az bağlantıya sahip sunucuya yönlendirme yaparak mevcut yükü göz önünde bulundurur. Bununla birlikte, daha karmaşık algoritmalar da mevcuttur, örneğin, Least Response Time veya Adaptive algoritmalar, sunucuların yanıt sürelerini veya mevcut yükü dinamik olarak değerlendirerek yönlendirme yapmaktadır. [6], [26]

Bu algoritmaların performanslarını kıyaslamak için çeşitli metrikler kullanılmaktadır. Örneğin, sistemdeki sunucuların kullanım oranları, yanıt süreleri, hizmet kesintileri ve yük dengeleyici tarafından sağlanan verimlilik gibi metrikler, algoritmaların performanslarını değerlendirmek için kullanılır.

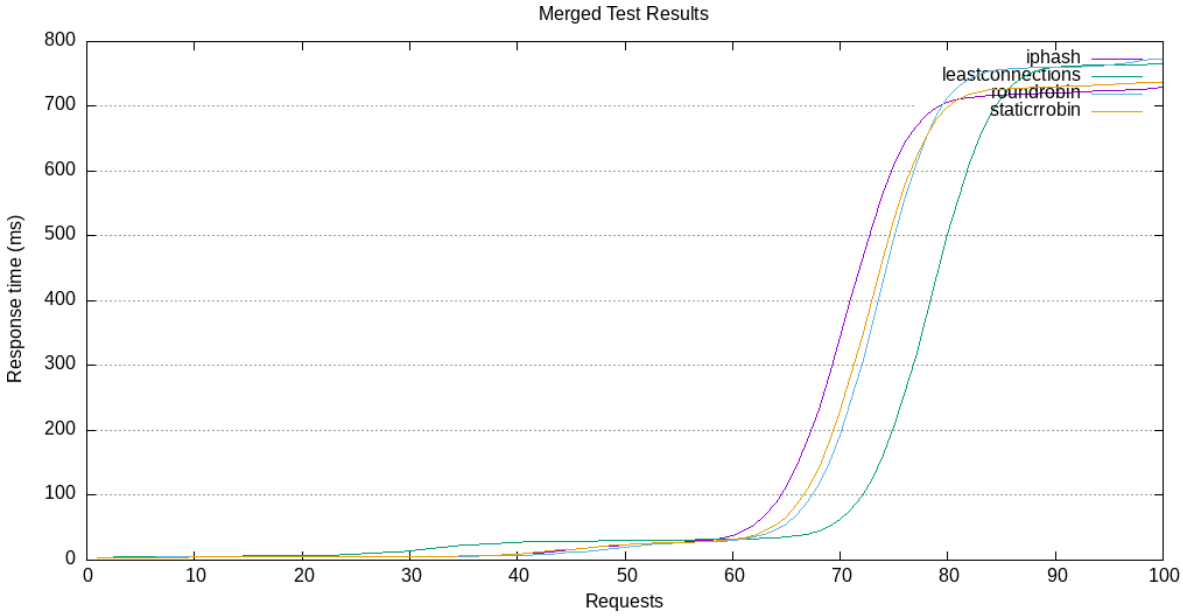
Gerçek dünya senaryolarını simüle etmek ve farklı trafik modellerini temsil etmek zor olduğu için çalışmalarının sonuçlarının genelleştirilmesi bu makalede önceliklendirilmiştir.

Bununla birlikte, algoritmaların performansını kıyaslayan bu çalışmanın, endüstriyel uygulamalar için rehberlik sağlaması hedeflenmiştir. Ayrıca, bu çalışma hazırlanırken yeni algoritmaların geliştirilmesi veya mevcut algoritmaların iyileştirilmesi için de temel oluşturması amaçlanmıştır.

4.1.Seçilen Algoritmanın Verimliliğe Etkisi

Bir yük dengeleme sisteminde alınan verim, sistemde kullanılan algoritmanın uygunluğuyla doğrudan ilgilidir. Kullanılan algoritmanın senaryoya uygunluğu arttıkça elde edilen verim de artar. Yapılan çalışmanın bağlamında verim, belirli bir süre içerisinde gelen trafiğin sunucular tarafından verilen tepki süresi olarak ifade etmektedir. Ağ iletişimi oldukça hızlı gerçekleştiği için tepki süresi ms cinsinden ifade edilmektedir. Yüksek trafik alan bir yük dengeleme sisteminde daha düşük bir tepki süresi verimli ve uygun bir algoritma kullanıldığına işaret eder. Çalışmada kullanılan performans metriklerinden biri de dengeli

yük dağılımıdır. İyi bir yük dengeleme algoritması, kaynakların eşit bir şekilde dağıtılmasını sağlar. Bu, her sunucunun benzer bir yük altında olması anlamına gelir. Düşük tepki süresi, kullanıcı taleplerine cevap verme süresi bakımından önemlidir. Doğru seçilmiş yük dengeleme algoritması, taleplere hızlı bir şekilde cevap verilmesini sağlar. Kullanılan ağ protokolü, sisteme bağlantı açan kullanıcı sayısı ayrıca tepki süresini etkileyebilir. Bu çalışmanın amacı, farklı sayıda eş zamanlı kullanıcının olduğu senaryolarda yük dengeleme algoritmalarının değiştirerek tepki sürelerini karakterize etmektir. Tepki süresi farklı algoritmalara göre değiştiğinden 5 farklı yük dengeleme algoritması değişen kullanıcı sayısı (100 – 1000) ile test edilmiştir. Şekil 17 ve Şekil 18'de farklı algoritmaların tepki süresi ile olan ilişkisi gösterilmektedir.

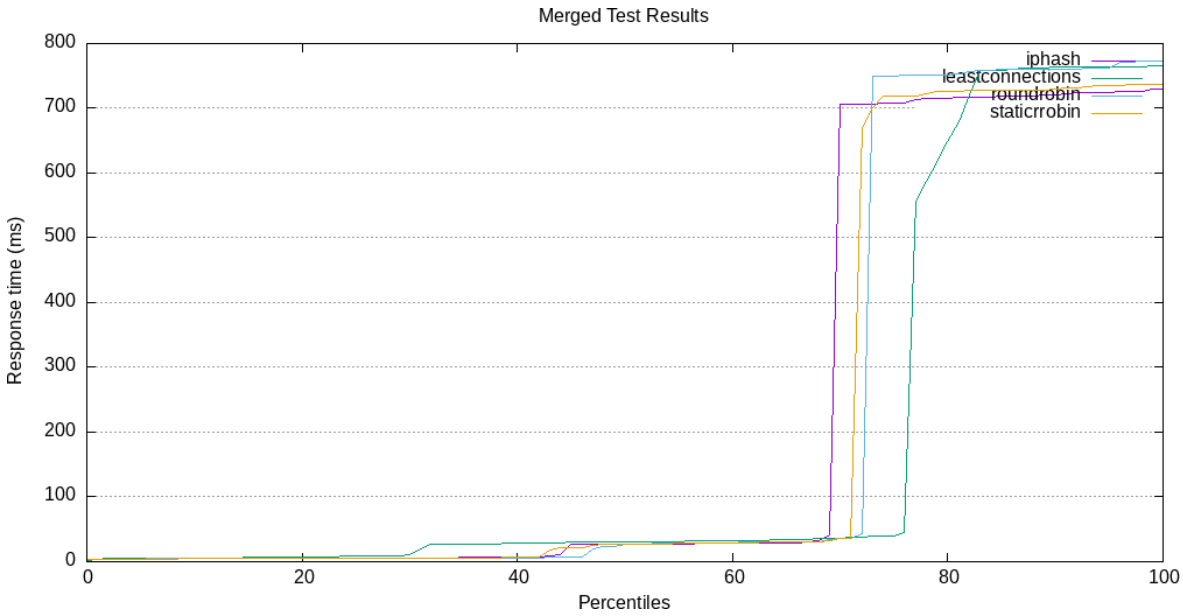


Şekil 17 *Algoritma ve Tepki Süresi İlişkisi*

4.2. Eş Zamanlı Kullanıcı Sayısının Verimliliğe Etkisi

Bir ağda yük dengeleme sisteminin verimi ve eş zamanlı bağlantı oluşturan kullanıcı sayısı arasındaki ilişki, ağ tasarımında kullanılan topoloji, iletişim katmanında kullanılan altyapı elemanları ve ağda kullanılan protokoller ve kontrol mekanizmaları gibi belirli faktörlerden etkilenebilir. Yüksek miktarda eş zamanlı kullanıcı paylaşılan kaynaklar arasında rekabet ettiğinde veya bant genişliğinin yetersiz olduğu durumlarda ağ performansında etkilere sebep olabilir. Eş zamanlı kullanıcı sayısı ve verim arasındaki ilişki, kullanılan yük dengeleme algoritmasına ve altyapıda tercih edilen donanımın özelliklerine göre değişebilir.

Eş zamanlı kullanıcı sayısındaki artış, sınırlı kapasiteli ve paylaşılan kaynaklara sahip ağlarda tıkanıklık ve kaynak çekişmesi gibi sorunlara yol açarak aktarım hızını ve dolayısıyla verimi düşürebilir. Ancak uygun yük dengeleme algoritmalarının kullanılması ve doğru bir ağ topolojisi planlanmasıyla birlikte artan eş zamanlı kullanıcı sayısının negatif etkileri minimize edilebilir.



Şekil 18 *Eş Zamanlı Bağlantı Sayısı ve Tepki Süresi İlişkisi*

Yapılan test çalışmasında eş zamanlı kullanıcı sayısı ve çeşitli yük dengeme algoritmalarının değişkenleri göz önüne alınarak sistemin performansı analiz edilmiştir. Yapılan testte verim, belirlenen zaman içerisinde yük dengeleyicinin bağlantıyı düşürmeden eş zamanlı olarak kabul edebileceği kullanıcı sayısını ifade eder. Her bir yük dengeleme algoritması için toplam 100 eşzamanlı kullanıcının 100 istek yapacağı bir senaryo hazırlanmıştır ve elde edilen sonuçlar Apache Benchmark'tan tablo formatında alınmıştır. Ardından elde edilen tablodaki veriler Gnuplot aracıyla grafiğe dökülerek görselleştirilmiştir. Şekilde bu ilişki bağlamında çizilen grafik gösterilmiştir. Eş zamanlı kullanıcı sayısının artışı ile yanıt sürelerinin artışı arasında bir korelasyon görülmektedir. Ancak eş zamanlı kullanıcı sayısının oldukça yüksek rakamlara çıkmasıyla birlikte bir noktadan itibaren tıkanıklık ve bant genişliği yetmemesi gibi sebeplerden ötürü korelasyon azalmaya başlamaktadır.

Tablo 1 'de, algoritmaların bağlantı süreleri arasındaki ilişki gösterilmiştir. En düşük ortalama bağlantı süresi sahip algoritma Least Connections olup, ortalama 165 ms bağlantı

süresi ile en iyi performansı sergilemiştir. En yüksek ortalama bağlantı süresi ise IP Hash algoritmasına aittir (214 ms).

Algorithm	Min Connect (ms)	Mean Connect (ms)	Median Connect (ms)	Max Connect (ms)
IP Hash	0	214	0	725
Least Connections	0	165	0	761
Random	0	206	0	747
Round Robin	0	203	0	768
Static Robin	0	201	0	732

Tablo 1 *Algoritma ve Bağlantı Süresi İlişkisi*

Tablo 2'de, işlem süreleri ile algoritmalar arasındaki ilişki sunulmuştur. Least Connections algoritması, 19 ms ile en yüksek ortalama işlem süresine sahipken, IP Hash ve Random algoritmaları 11 ms ile en düşük ortalama işlem süresine sahiptir. Static Robin algoritması, 12 ms ile işlem süreleri açısından ortalama bir performans sergilemiştir.

Algorithm	Min Processing (ms)	Mean Processing (ms)	Median Processing (ms)	Max Processing (ms)
IP Hash	3	11	6	41
Least Connections	3	19	26	43
Random	3	11	5	37
Round Robin	3	12	6	42
Static Robin	4	12	6	36

Tablo 2 *Algoritma ve İşlem Süresi İlişkisi*

Tablo 3, algoritmaların bekleme süreleri üzerindeki etkisini göstermektedir. **Random** algoritma, 11 ms ile en düşük ortalama bekleme süresine sahipken, **Least Connections** algoritması 19 ms ile en yüksek ortalama bekleme süresine sahiptir. Diğer algoritmalar ise bu iki değer arasında yer almaktadır.

Algorithm	Min Waiting (ms)	Mean Waiting (ms)	Median Waiting (ms)	Max Waiting (ms)
IP Hash	3	11	6	41
Least Connections	3	19	26	43
Random	2	11	5	37
Round Robin	3	12	6	42
Static Robin	3	12	6	36

Tablo 3 *Algoritma ve Bekleme Süresi İlişkisi*

Tablo 4'te, algoritmaların toplam süre üzerindeki etkisi gösterilmiştir. **Least Connections** algoritması, ortalama 184 ms toplam süre ile en iyi performansı sergilemiştir. **IP Hash** algoritması ise 225 ms ile en yüksek ortalama toplam süreye sahiptir. **Random** ve **Round Robin** algoritmaları, sırasıyla 217 ms ve 215 ms ortalama toplam süre ile benzer performanslar göstermektedir.

Algorithm	Min Total (ms)	Mean Total (ms)	Median Total (ms)	Max Total (ms)
IP Hash	3	225	27	729
Least Connections	3	184	30	765
Random	3	217	28	752
Round Robin	3	215	27	773
Static Robin	4	213	27	737

Tablo 4 *Algoritma ve Toplam Süre İlişkisi*

KAYNAKLAR

[1] M. Mueller, “Detaching Internet Governance from the State: Globalizing the IANA,” 2014.

[2] J. M. Sanjay Ignatius Ma Regina Hechanova, “Internet Usage from a Generational Perspective,” 2014.

[3] M. Hajibaba and S. Gorgin, “A Review on Modern Distributed Computing Paradigms: Cloud Computing, Jungle Computing and Fog Computing,” *Journal of Computing and Information Technology*, vol. 22, no. 2, p. 69, 2014, doi: 10.2498/cit.1002381.

[4] J. C. Patni and M. S. Aswal, “Distributed load balancing model for grid computing environment,” in *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, IEEE, Sep. 2015, pp. 123–126. doi: 10.1109/NGCT.2015.7375096.

[5] M. Al-Zewairi, D. Suleiman, and S. Almajali, “An experimental Software Defined Security controller for Software Defined Network,” in *2017 Fourth International Conference on Software Defined Systems (SDS)*, IEEE, May 2017, pp. 32–36. doi: 10.1109/SDS.2017.7939137.

[6] T. Wira Harjanti, H. Setiyani, and J. Trianto, “Load Balancing Analysis Using Round-Robin and Least-Connection Algorithms for Server Service Response Time,” *Applied Technology and Computing Science Journal*, vol. 5, no. 2, pp. 40–49, Dec. 2022, doi: 10.33086/atcsj.v5i2.3743.

[7] A. Kalia, R. Batta, J. Xiao, M. Choudhury, and M. Vukovic, “ACA: Application Containerization Advisory Framework for Modernizing Legacy Applications,” in *2021 IEEE 14th International Conference on Cloud Computing (CLOUD)*, IEEE, Sep. 2021, pp. 708–710. doi: 10.1109/CLOUD53861.2021.00090.

[8] C.-H. Hsu, U. Kremer, and C.-H. Hsu, “IPERF : A Framework for Automatic Construction of Performance Prediction Models.”

[9] V. Olteanu, A. Agache, A. Voinescu, and C. Raiciu, *Open access to the Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation is sponsored by USENIX. Stateless Datacenter Load-balancing with Beamer Stateless Datacenter Load-balancing with Beamer*.

[10] V. J. D. Barayuga and W. E. S. Yu, “Packet Level TCP Performance of NAT44, NAT64 and IPv6 Using Iperf in the Context of IPv6 Migration,” in *2015 5th International Conference on IT Convergence and Security (ICITCS)*, IEEE, Aug. 2015, pp. 1–3. doi: 10.1109/ICITCS.2015.7293006.

[11] J. C. Patni and M. S. Aswal, “Distributed load balancing model for grid computing environment,” in *2015 1st International Conference on Next Generation Computing Technologies (NGCT)*, IEEE, Sep. 2015, pp. 123–126. doi: 10.1109/NGCT.2015.7375096.

[12] H. Liu, X. Liao, and B. Du, “The applications of nature-inspired meta-heuristic algorithms for decreasing the energy consumption of software-defined networks: A comprehensive and systematic literature review,” *Sustainable Computing: Informatics and Systems*, vol. 39, p. 100895, Sep. 2023, doi: 10.1016/j.suscom.2023.100895.

[13] N. Kruber, M. Högqvist, and T. Schütt, “The Benefits of Estimated Global Information in DHT Load Balancing,” in *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, IEEE, May 2011, pp. 382–391. doi: 10.1109/CCGrid.2011.11.

[14] P. Francis and R. Gummadi, “IPNL,” in *Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, New York, NY, USA: ACM, Aug. 2001, pp. 69–80. doi: 10.1145/383059.383065.

[15] H. Zimmermann, “OSI Reference Model--The ISO Model of Architecture for Open Systems Interconnection,” *IEEE Transactions on Communications*, vol. 28, no. 4, pp. 425–432, Apr. 1980, doi: 10.1109/TCOM.1980.1094702.

[16] L. E. Hughes, “The Depletion of the IPv4 Address Space,” in *Third Generation Internet Revealed*, Berkeley, CA: Apress, 2022, pp. 119–146. doi: 10.1007/978-1-4842-8603-6_4.

[17] E. Jafarnejad Ghomi, A. Masoud Rahmani, and N. Nasih Qader, “Load-balancing algorithms in cloud computing: A survey,” *Journal of Network and Computer Applications*, vol. 88, pp. 50–71, Jun. 2017, doi: 10.1016/j.jnca.2017.04.007.

[18] M. N. Hoda, I. Bharati Vidyapeeth’s Institute of Computer Applications and Management (New Delhi, Institute of Electrical and Electronics Engineers. Delhi Section, and I. INDIAcom (Conference) (9th : 2015 : New Delhi, *2015 International Conference on Computing for Sustainable Global Development (INDIACom) : 11th to 13th March, 2015, Bharati Vidyapeeth’s Institute of Computers, Applications and Management (BVICAM)*.

[19] T. Hidayat, Y. Azzery, and R. Mahardiko, “Load Balancing Network by using Round Robin Algorithm: A Systematic Literature Review,” *Jurnal Online Informatika*, vol. 4, no. 2, p. 85, Feb. 2020, doi: 10.15575/join.v4i2.446.

[20] K. Leung, V. O. k. Li, and D. Yang, “An Overview of Packet Reordering in Transmission Control Protocol (TCP): Problems, Solutions, and Challenges,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, pp. 522–535, Apr. 2007, doi: 10.1109/TPDS.2007.1011.

[21] J. Postel, “Transmission Control Protocol,” Sep. 1981. doi: 10.17487/rfc0793.

[22] D. Clark, “The design philosophy of the DARPA internet protocols,” in *Symposium proceedings on Communications architectures and protocols*, New York, NY, USA: ACM, Aug. 1988, pp. 106–114. doi: 10.1145/52324.52336.

[23] R. Fielding *et al.*, “Hypertext Transfer Protocol -- HTTP/1.1,” Jun. 1999. doi: 10.17487/rfc2616.

[24] S. Aslam and M. A. Shah, “Load balancing algorithms in cloud computing: A survey of modern techniques,” in *2015 National Software Engineering Conference (NSEC)*, IEEE, Dec. 2015, pp. 30–35. doi: 10.1109/NSEC.2015.7396341.

[25] M. N. Hoda, I. Bharati Vidyapeeth’s Institute of Computer Applications and Management (New Delhi, Institute of Electrical and Electronics Engineers. Delhi Section, and I. INDIAcom (Conference) (9th : 2015 : New Delhi, *2015 International Conference on Computing for Sustainable Global Development (INDIACom) : 11th to 13th March, 2015, Bharati Vidyapeeth’s Institute of Computers, Applications and Management (BVICAM)*.

[26] L. Zhu, J. Cui, and G. Xiong, “Improved dynamic load balancing algorithm based on Least-Connection Scheduling,” in *2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC)*, IEEE, Dec. 2018, pp. 1858–1862. doi: 10.1109/ITOEC.2018.8740642.

[27] T. You, W. Li, Z. Fang, H. Wang, and G. Qu, “Performance Evaluation of Dynamic Load Balancing Algorithms,” *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 12, no. 4, Apr. 2014, doi: 10.11591/telkomnika.v12i4.4731.

[28] H. Gasmelseed and R. Ramar, “Traffic pattern–based load-balancing algorithm in software-defined network using distributed controllers,” *International Journal of Communication Systems*, vol. 32, no. 17, Nov. 2019, doi: 10.1002/dac.3841.

[29] W. Wang and G. Casale, “Evaluating Weighted Round Robin Load Balancing for Cloud Web Services,” in *2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, IEEE, Sep. 2014, pp. 393–400. doi: 10.1109/SYNASC.2014.59.

[30] D.-C. Li and F. M. Chang, “An In Out Combined Dynamic Weighted Round-Robin Method for Network Load Balancing,” *Comput J*, vol. 50, no. 5, pp. 555–566, Jun. 2007, doi: 10.1093/comjnl/bxm020.

[31] M. Adler, S. Chakrabarti, M. Mitzenmacher, and L. Rasmussen, “Parallel randomized load balancing,” in *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing - STOC ’95*, New York, New York, USA: ACM Press, 1995, pp. 238–247. doi: 10.1145/225058.225131.

[32] D. Choi, K. S. Chung, and J. Shon, “An Improvement on the Weighted Least-Connection Scheduling Algorithm for Load Balancing in Web Cluster Systems,” 2010, pp. 127–134. doi: 10.1007/978-3-642-17625-8_13.

[33] V. Nae, R. Prodan, and T. Fahringer, “Cost-efficient hosting and load balancing of Massively Multiplayer Online Games,” in *2010 11th IEEE/ACM International Conference on Grid Computing*, IEEE, Oct. 2010, pp. 9–16. doi: 10.1109/GRID.2010.5697956.

[34] X. Huang, Z. Guo, and M. Song, “FGLB: A fine-grained hardware intra-server load balancer based on 100 G FPGA SmartNIC,” *International Journal of Network Management*, vol. 32, no. 6, Nov. 2022, doi: 10.1002/nem.2211.

[35] S. Kadwadkar, “Latency Aware SmartNIC based Load Balancer (LASLB),” 2021.

[36] C. Zeng *et al.*, *This paper is included in the Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation. Open access to the Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation is sponsored by Tiara: A Scalable and Efficient Hardware Acceleration Architecture for Stateful Layer-4 Load Balancing Tiara: A Scalable and Efficient Hardware Acceleration Architecture for Stateful Layer-4 Load Balancing*. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/zeng

[37] J. Liang and Y. Kim, “Evolution of Firewalls: Toward Securer Network Using Next Generation Firewall,” in *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, Jan. 2022, pp. 0752–0759. doi: 10.1109/CCWC54503.2022.9720435.

[38] Md. S. Islam, M. A. Uddin, Dr. Md. S. Ahmed, and G. Moazzam, “Analysis and Evaluation of Network and Application Security Based on Next Generation Firewall,” *International Journal of Computing and Digital Systems*, vol. 13, no. 1, pp. 193–202, Jan. 2023, doi: 10.12785/ijcds/130116.

[39] Y. Bai, B. Jia, J. Zhang, and Q. Pu, “An Efficient Load Balancing Technology in CDN,” in *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery*, IEEE, 2009, pp. 510–514. doi: 10.1109/FSKD.2009.130.

[40] L. H. Pramono, R. C. Buwono, and Y. G. Waskito, “Round-robin Algorithm in HAProxy and Nginx Load Balancing Performance Evaluation: a Review,” in *2018 International Seminar on Research of Information Technology and Intelligent Systems (ISRITI)*, IEEE, Nov. 2018, pp. 367–372. doi: 10.1109/ISRITI.2018.8864455.

[41] C. Rawls and M. A. Salehi, “Load Balancer Tuning: Comparative Analysis of HAProxy Load Balancing Methods,” Dec. 2022.

[42] M. Masdari and M. Jalali, “A survey and taxonomy of DoS attacks in cloud computing,” *Security and Communication Networks*, vol. 9, no. 16, pp. 3724–3751, Nov. 2016, doi: 10.1002/sec.1539.

[43] M. Bogdanoski, T. Shuminoski, and A. Risteski, “Analysis of the SYN Flood DoS Attack,” *International Journal of Computer Network and Information Security*, vol. 5, no. 8, pp. 15–11, Jun. 2013, doi: 10.5815/ijcnis.2013.08.01.

[44] I. Sreeram and V. P. K. Vuppala, “HTTP flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm,” *Applied Computing and Informatics*, vol. 15, no. 1, pp. 59–66, Jan. 2019, doi: 10.1016/j.aci.2017.10.003.

[45] A. Tootoonchian and Y. Ganjali, “HyperFlow: A Distributed Control Plane for OpenFlow.”

[46] A. Nguyen-Ngoc, S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, and M. Jarschel, “Performance evaluation mechanisms for FlowMod message processing in OpenFlow switches,” in *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, IEEE, Jul. 2016, pp. 40–45. doi: 10.1109/CCE.2016.7562610.

[47] J. Cui, Q. Lu, H. Zhong, M. Tian, and L. Liu, “A Load-Balancing Mechanism for Distributed SDN Control Plane Using Response Time,” *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1197–1206, Dec. 2018, doi: 10.1109/TNSM.2018.2876369.

[48] F. Hauser *et al.*, “A survey on data plane programming with P4: Fundamentals, advances, and applied research,” *Journal of Network and Computer Applications*, vol. 212, p. 103561, Mar. 2023, doi: 10.1016/j.jnca.2022.103561.

[49] Y. S. Hong, J. H. No, and S. Y. Kim, “DNS-Based Load Balancing in Distributed Web-server Systems,” in *The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, and the Second International Workshop on Collaborative Computing, Integration, and Assurance (SEUS-WCCIA’06)*, IEEE, pp. 251–254. doi: 10.1109/SEUS-WCCIA.2006.23.

[50] A. Vakali and G. Pallis, “Content delivery networks: Status and trends,” *IEEE Internet Comput*, vol. 7, no. 6, pp. 68–74, Nov. 2003, doi: 10.1109/MIC.2003.1250586.

[51] M. Calder, A. Flavel, E. Katz-Bassett, R. Mahajan, and J. Padhye, “Analyzing the Performance of an Anycast CDN,” in *Proceedings of the 2015 Internet Measurement Conference*, New York, NY, USA: ACM, Oct. 2015, pp. 531–537. doi: 10.1145/2815675.2815717.

[52] A. Parihar and S. Sharma, “Smart Load Balancing in Cloud Using Deep Learning,” in *Deep Learning Approaches to Cloud Security*, Wiley, 2022, pp. 145–166. doi: 10.1002/9781119760542.ch10.

[53] T. Arnold *et al.*, “Beating BGP is Harder than we Thought,” in *Proceedings of the 18th ACM Workshop on Hot Topics in Networks*, New York, NY, USA: ACM, Nov. 2019, pp. 9–16. doi: 10.1145/3365609.3365865.

[54] M. Stecca, L. Bazzucco, and M. Maresca, “Sticky Session Support in Auto Scaling IaaS Systems,” in *2011 IEEE World Congress on Services*, IEEE, Jul. 2011, pp. 232–239. doi: 10.1109/SERVICES.2011.27.

[55] S. M. Hosseini, A. H. Jahangir, and S. Daraby, “Session-persistent Load Balancing for Clustered Web Servers without Acting as a Reverse-proxy,” in *2021 17th International Conference on Network and Service Management (CNSM)*, IEEE, Oct. 2021, pp. 360–364. doi: 10.23919/CNSM52442.2021.9615592.

[56] M. A. Saifullah and M. A. Maluk Mohamed Ph D, “Server Load Balancing Through Enhanced Server Health Report,” 2014. [Online]. Available: http://www.ripublication.com

[57] A. M. Potdar, N. D G, S. Kengond, and M. M. Mulla, “Performance Evaluation of Docker Container and Virtual Machine,” *Procedia Comput Sci*, vol. 171, pp. 1419–1428, 2020, doi: 10.1016/j.procs.2020.04.152.

[58] B. G. Moore, “Orbital Plots Using Gnuplot,” *J Chem Educ*, vol. 77, no. 6, p. 785, Jun. 2000, doi: 10.1021/ed077p785.