KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DİNAMİK WEB ÖNBELLEK GÜNCELLEME ALGORİTMASI VE WEB ÖNBELLEK SUNUCULARIN EN UYGUN ŞEKİLDE YERLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Bilgisayar Müh. Mohammad Edris RAJABY

OCAK 2018
TRABZON

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

DİNAMİK WEB ÖNBELLEK GÜNCELLEME ALGORİTMASI VE WEB ÖNBELLEK SUNUCULARIN EN UYGUN ŞEKİLDE YERLEŞTİRİLMESİ

Mohammad Edris RAJABY

Karadeniz Teknik Üniversitesi Fen Bilimleri Enstitüsünce
"BİLGİSAYAR YÜKSEK MÜHENDİSİ"
Unvanı Verilmesi İçin Kabul Edilen Tezdir.

Tezin Enstitüye Verildiği Tarih : 19 / 12 /2017
Tezin Savunma Tarihi : 08 / 01 /2018

Tez Danışmanı : Yrd. Doç. Dr. Tuğrul ÇAVDAR

Trabzon 2018

KARADENİZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Bilgisayar Mühendisliği Anabilim Dalında
Mohammad Edris RAJABY Tarafından Hazırlanan

DİNAMİK WEB ÖNBELLEK GÜNCELLEME ALGORİTMASI VE WEB ÖNBELLEK SUNUCULARIN EN UYGUN ŞEKİLDE YERLEŞTİRİLMESİ

başlıklı bu çalışma, Enstitü Yönetim Kurulunun 19 / 12 / 2017 gün ve 1732 sayılı kararıyla oluşturulan jüri tarafından yapılan sınavda
YÜKSEK LİSANS TEZİ
olarak kabul edilmiştir.

Jüri Üyeleri

Başkan : Doç. Dr. Burhan ERGEN .....

Üye : Yrd. Doç. Dr. Tuğrul ÇAVDAR .....

Üye : Yrd. Doç. Dr. İbrahim SAVRAN .....

Prof. Dr. Sadettin KORKMAZ
Enstitü Müdürü

ÖNSÖZ

Web önbellekler çok uzun zamandan beri ağların yükünü azaltmak ve müşterilerin internet erişimini yükseltmek amacıyla kullanılan bir bilgi teknolojisidir. Web önbelleklerin evrimine göre son zamanlarda CDN ve ICN gibi yeni teknolojiler önerilmiştir. Önerilen tüm teknolojiler, geleneksel web önbellek mantığına dayanmaktadır. Tüm teknolojilerin ana bileşeni, hafıza güncelleme algoritmalar olarak tanınmaktadır. Bu teknolojilerin uygulamasında önbellek sunucuların uygun noktalarda yerleştirilmesi günümüze kadar ortak bir problem yaşanmaktadır. Bu tez de önbelleklerin performansını arttırmak için yeni dinamik web önbellek güncelleme algoritması ve web önbellek sunucuların en uygun yerleştirilmesi için yeni yöntem önerilmiştir.

Böyle güncel ve çok önemli bir konuyu seçmemde yardım eden ve çalışma boyunca desteklerini esirgemeyen sayın danışman hocam Yrd. Doç. Dr. Tuğrul ÇAVDAR'a ve değerli dostum Abdullah YUSEFİ'ye şükranlarımı sunarım. Ayrıca, bu tezin yazmasında emeği geçen saygıdeğer hocam Yrd. Doç. Dr. Muzaffer UZUN'a teşekkür ederim. Son olarak hayat arkadaşım sevgili eşim Ahdia QAEYM ve hayatım boyunca her an yanımda olan dünyanın bulunmaz nimeti, başta sevgili annem ve değerli babam olmak üzere tüm aileme teşekkürü bir borç bilirim.

Mohammad Edris RAJABY
Trabzon 2018

TEZ ETİK BEYANNAMESİ

Yüksek Lisans Tezi olarak sunduğum "Dinamik Web Önbellek Güncelleme Algoritması ve Web Önbellek Sunucuların En Uygun Şekilde Yerleştirilmesi" başlıklı bu çalışmayı baştan sona kadar danışmanım Yrd. Doç. Dr. Tuğrul Çavdar'ın sorumluluğunda tamamladığımı, verileri kendim topladığımı, analizleri ilgili laboratuvarlarda yaptığımı, başka kaynaklardan aldığım bilgileri metinde ve kaynakçada eksiksiz olarak gösterdiğimi, çalışma sürecinde bilimsel araştırma ve etik kurallara uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim. 08/01/2018

Mohammad Edris RAJABY

İÇİNDEKİLER

Sayfa No

Yüksek Lisans Tezi

ÖZET

DİNAMİK WEB ÖNBELLEK GÜNCELLEME ALGORİTMASI VE WEB ÖNBELLEK SUNUCULARIN EN UYGUN ŞEKİLDE YERLEŞTİRİLMESİ

Mohammad Edris RAJABY

Karadeniz Teknik Üniversitesi
Fen Bilimleri Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı
Danışman: Yrd. Dr. Tuğrul ÇAVDAR
2018, 65 Sayfa

Uzun zamandan günümüze kadar web önbellekler ağ yükünü azaltmak ve kullanıcıların erişim hızını iyileştirmek için uygun bir çözüm olarak kullanılmaktadır. Ancak HTTPS trafiğin artması ve bazı ağlarda daha güvenli bir şekilde internete bağlanmak için VPN teknolojisi gibi yöntemlerin kullanılması, geniş alan ağlar ve İSS'lerde web önbelleklerin daha uygun yerlerde yerleştirilmesini ciddi problem olarak ortaya çıkartmaktadır. Web önbelleklerin ana bileşeni önbellek güncelleme algoritmalarıdır. Önbelleğin hafızası dolduğunda, bu algoritmalar eski veya sık kullanılmayan nesneleri, yeni veya gelecekte kullanıcılar tarafından daha yüksek kullanım olasılığına sahip olabilecek nesneler ile değiştirir. Bu makale iki çalışmayı içermektedir. Birinci çalışmada veri madenciliği ve ağ özelliklerine dayalı, nesnelerin yenilik, frekans, boyut ve gecikme metriklerini dikkate alarak çalışan, her ağda iyi performans gösteren önbellek güncelleme algoritması önerilmiştir. İkinci çalışmada ise bağlantı yükü, bant genişliği ve trafik özellikleri dikkate alınarak, EIGRP maliyet fonksiyon mantığına dayalı web önbelleklerin optimal yerleştirilmesi için bir yöntem önerilmiştir. Simülasyon sonuçlarına göre önerilen önbellek güncelleme algoritma, farklı ağlarda başka algoritmalardan daha iyi performans göstermektedir. Önerilen web önbellek yerleştirme yöntemi ile sıralanan önbelleklerin simülasyon sonuçları, ileri sıralarda yer alan önbelleklerin tasarruf eden bant genişliği oranı kendilerinden sonraki sıralarda yer alan önbelleklerin oranından yüksek çıkmaktadır. Önbelleklerin tasarruf eden bant genişliği oranı ve sıra rakamlarının normal logaritmalarının grafiği Zipf yasasına uyduğunu göstermektedir.

Anahtar Kelimeler: Dinamik web önbellek güncelleme algoritması, veri madenciliği, metrik puanlama, web önbellek yerleştirme, web önbellek ve Zipf

Master Thesis

SUMMARY

DYNAMIC WEB CACHE REPLACEMENT ALGORITHM AND OPTIMAL PLACEMENT OF WEB CACHE SERVERS

Mohammad Edris RAJABY

Karadeniz Technical University
The Graduate School of Science
Computer Engineering Graduate Program
Supervisor: Assistant Prof. Dr. Tuğrul ÇAVDAR
2018, 65 Pages

Web caches are used as a convenient solution for network load reduction and user access speed improvement. With the rise of modern technologies like VPN to secure connections, placement of web cache servers in optimal locations has become a serious concern in WANs and ISPs. Web cache replacement policies are the main part of web cache servers. In case the memory is full, these algorithms replace old or infrequently used objects with those that have higher probability of usage by new or future users. This article consists of two studies. First study proposes a dynamic web cache replacement algorithm which trains itself utilizing previous network logs and by exploiting the data mining clustering algorithm. It shows the best performance on different networks by scoring each metric of the enquiries including recency, frequency, size and delay. Considering the connection load, bandwidth and traffic characteristics, second study proposes a method for optimal placement of web cache servers. According to simulation results, proposed algorithm has better performance compared to other traditional algorithms in different network according to HR, BHR and DR. The simulation results of cache placement show that the web caches with higher priorities save more bandwidth compared to caches in next queues. By graphing the normal logarithms of the saved bandwidth ratio and priority numbers it has been shown that the prioritized caches comply the Zipf's law.

Key Words: Dynamic web cache replacement policy, data mining, metric scoring, cache server placement, web cache servers and Zipf's law

ŞEKİLLER DİZİNİ

Sayfa No

TABLOLAR DİZİNİ

Sayfa No

SEMBOLLER DİZİNİ

ALFUR : Average Least Frequency Used Removal
ARC : Adaptive Replacement Cache
BHR : Byte Hit Rate
CAIDA : Center for Applied Internet Data Analysis
CDN : Content Delivery Networks
DR : Delay Ratio
EIGRP : Enhanced Interior Gateway Routing Protocol
G_Puan : General Puan
GDSF : Greedy Dual Size Frequency
HR : Hit Rate
HTML : Hypertext Markup Language
HTTP : Hyper Text Transfer Protocol
HTTPS : Hyper Text Transfer Protocol Secure
ICN : Information Centric Networks
ID : Identity Document
IOT : Internet of Things
ISP : Internet Service Provider
ISS : İnternet Servis Sağlayıcılar
IT : Information Technology
Kb : Kilo Bit
KDD : Knowledge Discovery from Data
LAN : Local Area Network
LFU : Least Frequently Used
LRU : Least Recently Used
MATLAB : Matrix Laboratory
Mb : Mega Bit
MS-DOS : Microsoft Disk Operating System
MTU : Maximum Transmission Unite
NOC : Network Operation Center
POP : Point of Presence

PRTG : Paessler Router Traffic Grapher

RFSD : Recency, Frequency, Size, Delay

SSE : Sum of Squared Errors

SSL : Secure Sockets Layer

TLS : Transport Layer Security

URL : Uniform Resource Locator

VPN : Virtual Private Network

WAN : Wide Area Network

WRP : Weighting and Replacement Policy

1. GENEL BİLGİLER

1.1. İletişim Ağları

İletişim ağların temel amacı iki cihaz arasında bilgi alışverişi, kaynak ve servis paylaşımıdır [1]. Bu cihazlar birbiri ile kablolu ya kablosuz olarak bağlanabilir. Kablo ile bağlanan cihazlar kablolu iletişim ağları ve kablosuz bağlanan cihazlar kablosuz iletişim ağlarını oluşturmaktadır. Cihazlar uzak mesafelere ve aynı anda birden fazla cihazla iletişim kurabilmesi için bir merkezi sisteme bağlamaları gerekmektedir. İletişim ağları coğrafi kapsamlarına göre üç kategoriye ayrılmaktadır [2].

1.1.1. Yerel Alan Ağları (LAN'lar)

Kapsam alanı küçük olan bir ağ kategorisidir. Genel olarak kapsam alanı bir bina ya da bir bina kümesindedir ve bu ağların kontrolü, cihazların sahip olduğu kişi ve ya kurumun kontrolü altındadır [3].

1.1.2. Metropoliten Alan Ağıları (MAN'lar)

Eski kullanımında, bazen kampüs ağları olarak da adlandırılmıştır. LAN'lardan daha büyük ve WAN'lardan daha küçük kapsam alanda sahip olan ağlardır. Bir şehirdeki cihazların tek bir merkezi sistemle bir biri ile bağlamaktadır.

1.1.3. Geniş Alan Ağları (WAN'lar)

En büyük kapsam alana sahip olan ağ tipidir. Şehirleri, ülkeleri ve hatta kıtaları bir birisi ile bağlamakta olan ağlara denir. WAN'lar, kurumsal geniş alan ağlar ve küresel geniş alan ağlar olarak ayrılmaktadır. Kurumsal WAN ismi ifade ettiği gibi, bir şirket veya bir kuruma ait olan WAN'ları içermektedir. Küresel WAN'lar herhangi bir şirket veya kuruma ait değildir ve ulusal sınırları aşabilir. Küresel WAN'ların bilinen en iyi örneği olarak günümüzdeki interneti örnek verebiliriz [2].

Günümüzde Geniş alan ağlar, bir ülkeyi, bir kıtayı veya hatta tüm dünyayı kapsayabilecek büyük coğrafi alanlar üzerinden veri, resim, ses ve video gibi bilgilerin uzun mesafeli iletilmesini sağlamaktadır [4]. Bu tip ağlara her an bir şekilde bağlanmaktayız. Kısaca, WAN'lar birçok MAN'ların ve LAN'ların bir birisine bağlanması ile oluşur. MAN'lar ve LAN'lar arası veya cihazlar ile LAN'ların bağlantısı kablo veya radyo dalgası olabilir. Radyo dalgası ile bağlantı koran ve hizmet veren geniş alan ağlara kablosuz geniş alan ağları (WWAN) denir. Hücresel *(Cellular)* iletişim ağlar WWAN'ların bilinen en iyi örneği olarak tanınmaktadır. Genel olarak WAN'lar bilgisayar ve hücresel ağların hepsini içermektedir.

1.2. İnternet Servis Sağlayıcılar

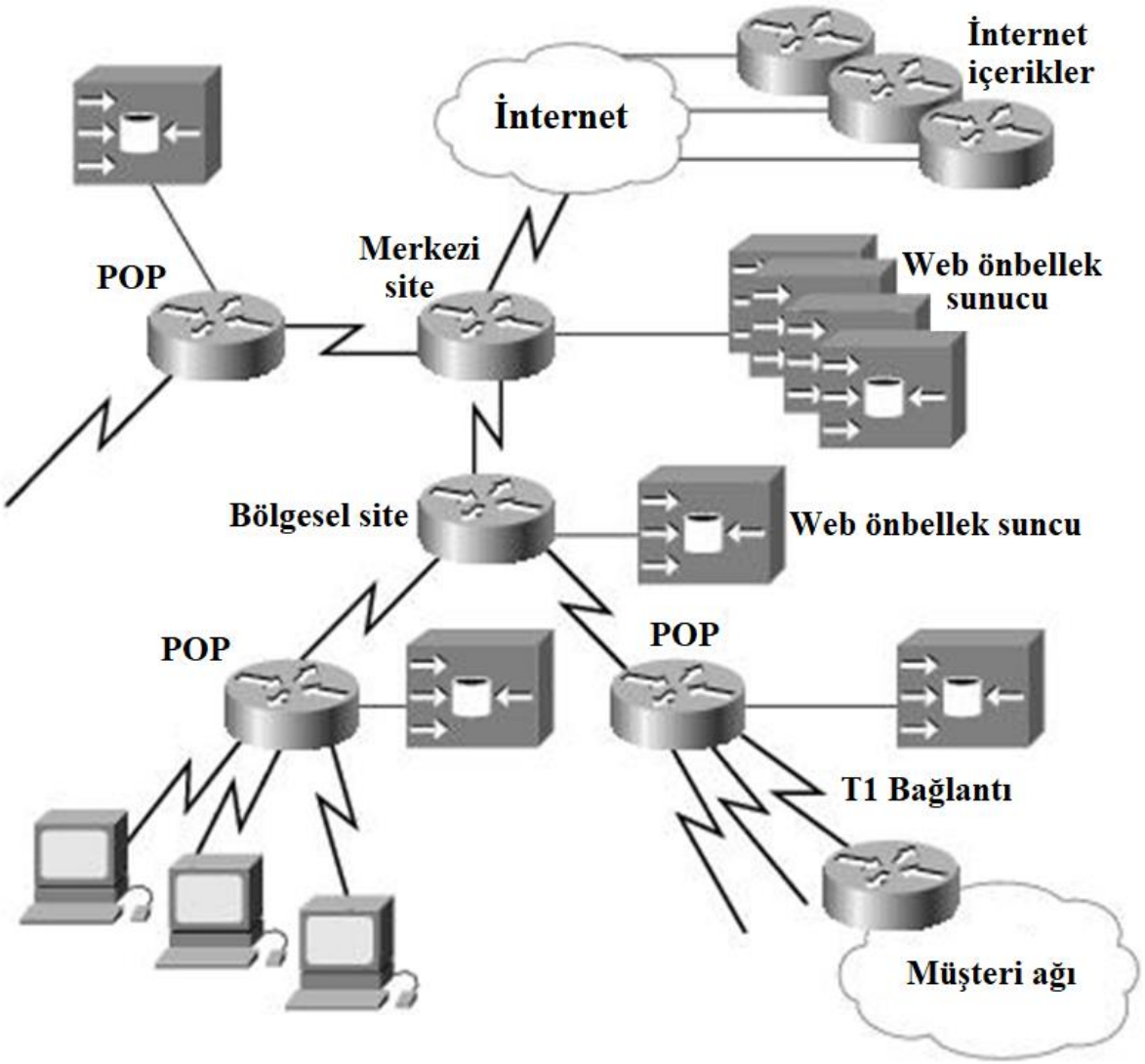
İnternet Servis Sağlayıcılar (İSS) ağları, geniş alan ağların bir örneği olarak tanınmaktadır. İSS, bireyler ve diğer şirketlere internet erişimi, web oluşturma ve sanal barındırma gibi diğer ilgili hizmetleri sağlayan bir şirkettir. Bir ISS ağ topolojisi üç tabakadan oluşur [5]. 1- Merkezi site *(central site)* tüm bölgesel sitelere hizmet vermek için bir merkeze bağlanan sitedir. 2- Bölgesel site *(regional site)* bir bölgede yer alan POP *(Point of Presence)*'lere hizmet verir. 3- POP'lar, İSS'nin kenar noktalarıdır ve müşterilere hizmet verir. İSS'lerin boyutu doğrudan POP'larının sayısı ile ilgilidir. Bir İSS ne kadar çok POP sayısına sahipse o kadar büyük anlamına gelir. İSS'lerin, bölgesel ve kenar siteler ile bağlanması için telekomünikasyon bağlantılarına ihtiyaç vardır. Büyük İSS'lerin kendi yüksek hızlı kiralık hatları vardır, böylece telekomünikasyon sağlayıcılarına daha az bağımlıdır ve müşterilerine daha iyi hizmet verebilirler [6].

1.3. Geniş Alan Ağ/İSS Topolojisi

İSS'ler geniş alan ağların bir örneği olduğundan burada Cisco tarafından sunulan genel bir İSS topolojisi örnek olarak gösterilir. Şekil 1'de gösterilen İSS topolojisinde gerekli bolümler olarak, merkezi site, bölgesel site, POP'lar ve web önbellek sunucuları bulunmaktadır. Bir İSS'in büyük veya küçük olması mevcut olan POP'lar sayısı üzerinden tespit edilir. Şekilde görüldüğü gibi web önbellek sunucuları hiyerarşik olarak tüm bölümlerde (merkezi site, bölgesel site ve POP'larda) kurulması mümkündür. Eğer bir İSS

yöneticisi topolojide gösterilen merkezi sitede bir web önbellek sunucu yerleştirirse, tüm POP'lar ve bölgesel siteler o önbellek hizmetinden faydalanırlar. Tüm müşterilerin istekleri POP'lara geldiğinde merkezi sitede yer alan web önbellek sunucusuna iletilir. Merkezi sitedeki web önbellek sunucusu istenilen belgelerin hafızasında mevcut olup olmadığını kontrol eder. Eğer istenilen belge hafızasında mevcutsa doğrudan müşteri isteğine yanıt verir. Müşteriler tarafından istenilen belgelerin hafızada mevcut olmadığı takdirde web önbellek sunucusu müşteri isteklerini orijin sunucuya iletilir ve orijin sunucudan belgeyi aldıktan sonra bir kopyasını kendi hafızasında gelecek istekler için kaydedip müşteri isteğine yanıt verir.

Müşterilere sunulan hizmetlerin kalitesini daha da iyileştirmek için, İSS yöneticileri her bir POP'ta ve bölgesel sitelerde web önbellek sunucusunu yerleştirebilir. Böylece müşterilerin istekleri, ilk önce POP'lara geldiği için, ilgili POP önce kendi önbellek hafızasını istenilen belgenin olup olmadığı için kontrol eder. Müşteri tarafından istenilen belge POP'taki web önbellekte mevcutsa ve geçerlilik süresi kalkmamış ise hemen müşterinin isteğine yanıt verilir. POP'ta yerleştirilen önbellek sunucusu, isteği kendi hafızasından temin edemiyorsa, isteği bölgesel siteye gönderir. Bölgesel site de kendi hafızasını kontrol edip hafızasında bulunan istekleri kendi hafızasından yanıt verir ve mevcut olmayan istekleri merkezi siteye iletir. Müşterilerin istekleri ne kadar yakın önbellek sunucusundan temin edilirse o kadar çok kullanıcıların verilere erişim zamanı düşer, internet bağlantısının trafiği azalır ve orijin sunucular daha düşük talep görür [5]. Genel olarak müşterilerin istekler herhangi bir web önbellek sunucusundan temin edildiğinde kullanıcıların verilere erişim zamanı düşer, internet bağlantısının trafiği azalır ve orijin sunucular daha düşük talep görür.



Şekil 1. İnternet servis sağlayıcının mimarisi [5].

1.4. Web Önbellekler

İnternet kullanıcılarının ve cihazların hızlı bir şekilde artması internet trafiği ve statik verilerin üssel olarak artmasına neden olmuştur. Büyük miktardaki kullanıcı taleplerine etkili bir şekilde yanıt vermek için, yüksek bant genişliği olan bağlantılar, yüksek hızlı sunucular ve güçlü ekipmanlara ihtiyaç duyulur; bu gereksinimlerin mevcut olmasına rağmen tam kullanıcı memnuniyetini elde etmek son derece zor bir mücadele olarak tanımlanmıştır. Bahsi geçen, sorunların üstesinden gelmek için ağlarda web önbellek sunucuları uzun zamandan beri uygun bir çözüm olarak kullanılmaktadır [7]. Web Önbellekleme veya HTTP *(Hyper Text Transfer Protocol)* önbellekleme mekanizması, HTML *(Hypertext Markup Language)* sayfaları, resimleri, videoları ve benzer statik verileri geçici bir şekilde

depolayarak bant genişliği kullanımını, sunucu yükünü ve algılanan gecikmeyi azaltan bir bilgi teknolojisidir [8]. Diğer bir deyişle web önbellekleme sadece HTTP statik veri trafiği üzerinde yapılmaktadır. Örneği resimler, videolar ve HTML kodları gibi statik veriler web önbellekte kaydedilir ve tekrar bir kullanıcı tarafından istendiğinde web önbellekten temin edilir.

Web önbellek sunucuların kullanılması, internette web performansını önemli ölçüde arttırdığı görülmüştür [9] [10] [11]. Web önbelleklerin ağlarda sağladığı avantajlar şunlardan ibarettir [12].

1- Web önbellekleme teknolojisi bant genişliği tüketimini ve ağ trafiğini azaltarak ağ tıkanıklığını ortadan kaldırır.
2- Orijin web sunuculardaki verilerin kopyalarının, kullanıcılar ve web sunucusu arasında yer alan web önbelleklerde depolanması web sunucuların iş yükünü azaltır.
3- Web önbellekleme, aşağıda verilen iki nedene bağlı olarak erişim gecikmesini azaltır:
   a) Kullanıcılar tarafından sık istenilen belgeler uzaktaki web sunucuların yerine yakındaki önbellek sunucusundan temin edilir ve iletim gecikmesi en aza indirilir.
   b) Bağlantıların yoğunluğu ve web sunucuların yükü azalınca önbellekten temin edilemeyen belgeler orijin web sunucularından daha hızlı bir şekilde kullanıcılara iletilir.
4- Farklı nedenlerden dolayı web sunucusu kısa süre devre dışı kaldığında web önbellek sunucuları, kullanıcılara hizmet verir. Böylece, web hizmetinin sağlamlığı artırılır.
5- Web önbellekler, ağ trafiğinin analizi için ağ yöneticilere yardımcı olur.

Web önbellek mekanizması internette üç fraklı seviyede uygulanmaktadır. 1-Kullanıcı seviyesi *(client level)* 2-Proxy seviyesi *(proxy level)* 3-Orijinal sunucu seviyesi *(original server level)* [12].

1.4.1. Tarayıcı Önbelleği

İstemci cihazında bulunur. Kullanıcı, Internet Explorer, Safari, Mozilla Firefox, Netscape ve Google Chrome gibi modern web tarayıcıları önbellek ayarlarını görebilir. Bu

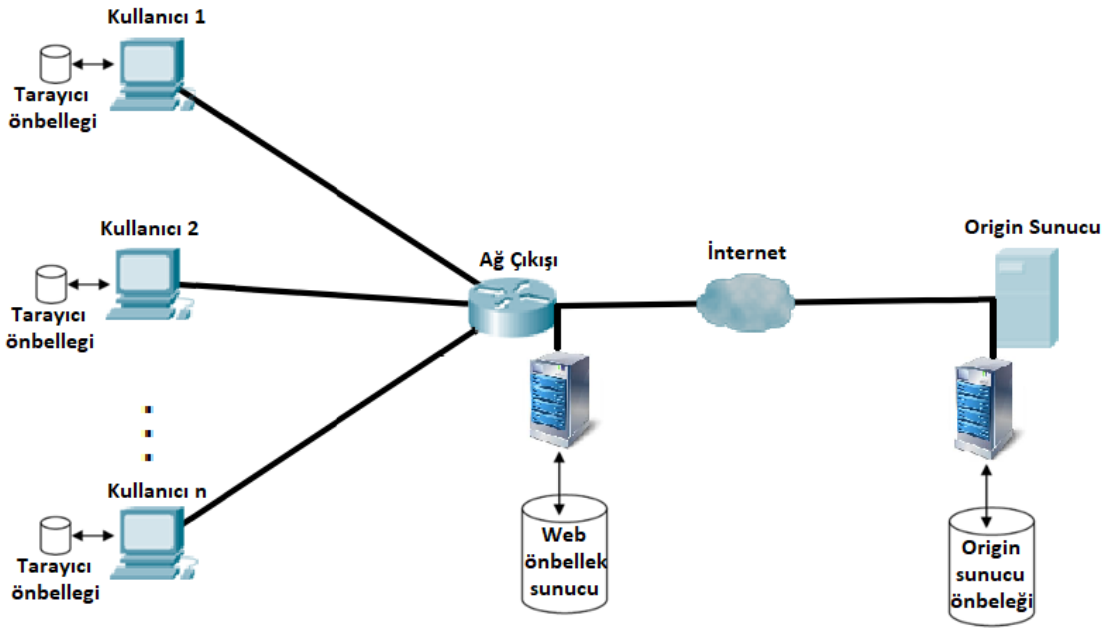
önbellek, özellikle, kullanıcılar geri (back) düğmesine bastığında veya ziyaret ettikleri bir sayfaya yakın zaman aralığı ile tekrardan baktığında yararlıdır.

1.4.2. Proxy Önbellek/Web Önbellek Sunucu

İstemci cihazı ve orijin web sunucular arasında bulunur. Tarayıcı önbelleği ile aynı prensibe dayalıdır, ancak tarayıcı önbelleği bir istemciye hizmet verirken web önbellek sunucular daha çok istemciye hizmet verir. Web önbellek sunucusu Bir istek alındığında, ilk önce kendi hafızasını kontrol eder. İstenilen belge hafızasında mevcutsa ve geçerliliği kalkmamışsa, belgeyi hemen istemciye gönderir. Eğer istenilen belge hafızasında mevcut değilse veya geçerliliği kalkmış ise web önbellek sunucusu isteği orijin sunucuya iletir ve belgeyi aldıktan sonra istemciye gönderir. Belgenin bir kopyası önbellek hafızasına gelecek istekler için kaydedilir. Proxy sunucular veya web önbellek sunucular, kullanıcılar ve web sunucuları arasındaki gecikmeyi azaltmak, bant genişliğini daha verimli kullanılması konusunda kilit rol oynamaktadır.

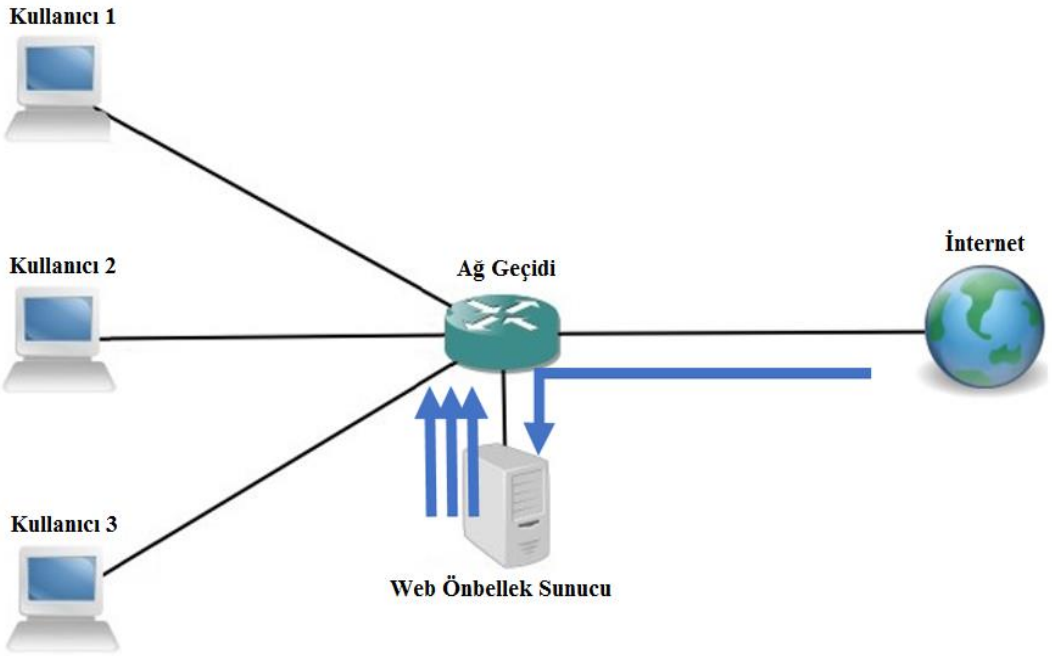
1.4.3. Orijin Sunucu Önbelleği

Orijin web sunucusunda yakın bir noktada, web sunucusunun yükünü azaltmak amacıyla önbellek sunucusu kullanılır. Bu tür önbellekler orijin sunucu önbellek ismini taşımaktadır. Bu tür önbelleklerde, orijin sunuculardaki yüksek frekansa sahip veriler önbellek hafızasına kaydedilir ve kullanıcılar tarafından tekrar istendiğinde orijin sunucunun yerine önbellek hafızasından temin edilir. Tarayıcı önbellekler, web önbellek sunucular ve orijin sunucu önbellekler şekil 2'de gösterilmiştir.



Şekil 2. Web önbelleklerin türleri

Proxy sunucular veya web önbellek sunucular, kullanıcılar ve web sayfalar arasında gecikmeyi azaltmak ve bant genişliğini daha verimli kullanılması konusunda kilit rol oynamaktadır [12]. Bu sunucular HTTP protokolü kullanan statik verileri depolayarak erişim performansını arttırır ve trafikler için İnternet bant genişliğini boşaltır. Bu nedenle, algılanan gecikmeyi azaltmak ve genel olarak kullanıcılara internet hızını arttırmak için proxy seviyesindeki web önbellek sonuncularda daha verimli yaklaşımlar oluşturulmalıdır. Proxy seviyesinde web önbellek sunucusunun yerleşimi ve çalışma prensibi şekil 3'te gösterilmiştir.



Şekil 3. Proxy önbellek/Web önbellek sunucusu

Web önbellek sistemi üzerinden geçen ve belirli koşulları yerine getiren statik veriler hafızaya kaydedilir ve sonraki istekler önbellek sunucusundan temin edilir. Bu çalışmada adı geçen en sık kullanılan web önbellek türü süz konusudur.

1.5. Web Önbelleklerin Çalışma Mekanizması

Tüm önbellekler bazı kurallara göre verilerin önbelleklenebilir olup olmamasını teşhis etmektedir. Bu kuralların bazıları protokollerde (HTTP 1.0 ve http 1.1) belirlenir ve bazıları önbellek yöneticisi (tarayıcı önbellek kullanıcısı veya web önbellek sunucusunun yöneticisi) tarafından ayarlanır. Genel olarak önbelleklerin önbelleğe alma ve sunma kuralları şunlardır [13] [14].

1- Eğer HTTP yanıt başlıkları önbellekte saklanmamasını söylerse, önbelleğe alınmaz.
2- İstek doğrulanmış veya güvenli ise (örneği, HTTPS protokolü kullanmışsa), önbellekler tarafından önbelleğe alınmaz.
3- Önbellekte bulunan bir belge kullanıcılara göndermek için geçerli sayılır, eğer:
   - Geçerlilik süresi geçmemiş ise, veya

- Yakın zamanda verinin geçerlilik süresi web önbellek tarafından güncellenmiş ise.

Geçerli veriler, orijin sunucusuna bakılmadan doğrudan önbellekten sunulmaktadır.

4- Geçerlilik suresi geçmiş bir veri kullanıcı tarafından istenildiğinde, verinin hala değişip değişmediği orijin sunucuya sorulacaktır. Bu işlem verilerin onaylanması olarak tanınmaktadır.

5- Belirli koşullar altında - örneğin bir ağ bağlantısı kesildiğinde - web önbellekler, orijinal sunucuyla iletişime geçmeden hafızada mevcut onaylanmayan verileri kullanıcılara sunabilir.

Geçerlilik ve onaylama önbelleklerin çalışması için çok önemlidir. Geçerlilik suresi geçmemiş veriler hemen önbellek hafızasından temin edilir. Geçerlilik suresi geçmiş ve orijin sunucu tarafından onaylanması gerekken bilgiler kullanıcılar tarafından istenildiğinde web önbellek hafızasındaki veriyi hala güncel olduğunu orijin sunucu ile iletişime geçerek tespit eder ve orijin sunucunun yanıtına göre veriye yeni geçerlilik süre belirler. Bir HTTP yanıtının başlığı aşağıdaki şekil 4'te gösterilmiştir.

```
HTTP/1.1 200 OK
Date: Fri, 30 Oct 1998 13:19:41 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 30 Oct 1998 14:19:41 GMT
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT
ETag: "3e86-410-3596fbbc"
Content-Length: 1040
Content-Type: text/html
```

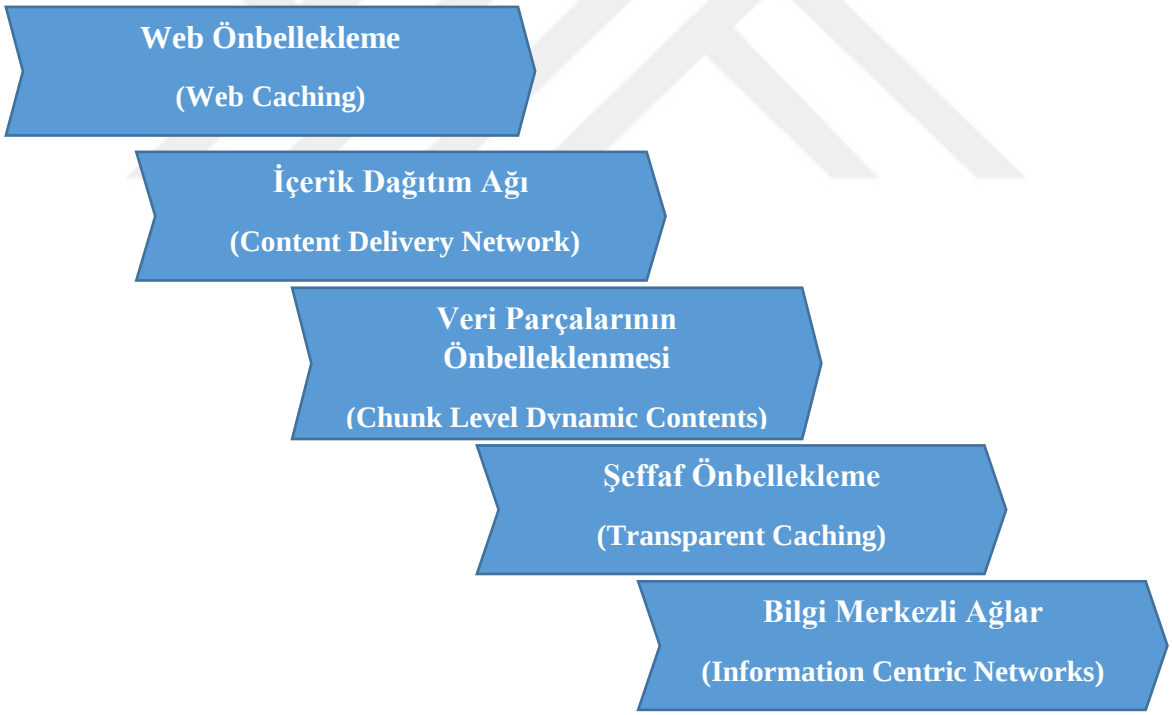
Şekil 4. HTTP yanıtın Başlığı [13].

HTTP başlıkları hem tarayıcı önbelleklere ve hem de proxy önbelleklere veriler üzerinde işlem yapabilmeleri için çok kolaylık sağlamaktadır. Genellikle HTTP başlıklarında yer alan bilgiler orijin web sunucuları tarafından otomatik olarak üretilir. HTTP başlıklarında, *Expires* ile başlayan satır, web önbellekteki bilgilerin güncel tutulması ve önbellek hafızasında mevcut olan, orijin sunucusunda da değişmeyen verilerin tekrar

indirilmesini engellemek için en önemli bilgidir. Bu bilgiler bir verinin ne zamana kadar geçerli olduğunu gösterir. Çoğu web sunucular, yanıt başlıkların geçerlilik süresini değiştirmesini farklı yollar ile izin vermektedir. Geçerlilik suresinin değiştirilmesi ve daha uzun sure olarak ayarlanması bir web sayfasının başlığında yer alan statik resimler için daha verimlidir. Çünkü bu tür veriler çok değişmez ve geçerlilik süresini daha süre olarak belirlendiğinde, web sitedeki statik veriler kullanıcılara en yankın web önbellekten temin edilir. Böylece web sitelerin daha duyarlı olması sağlanır.

1.6. Web Önbelleklerin Evrimi

Daha önce belirtildiği gibi, web önbellekleme, ağ trafiği ve sunucu yükünü azaltırken son kullanıcı erişimini iyileştirir; Ayrıca, hizmet kalitesinin artmasını sağlar. Şekil 5'te web önbelleklerin evrimi grafiksel olarak gösterilmiştir [15].



Şekil 5. Web önbelleklerin evrimi

1.6.1. Web Önbellekleme

İnternette algılanan gecikmeyi azaltmak amacıyla ilk kez yaygın olarak kullanılan ve kabul edilen çözümdür [16].

1.6.2. İçerik Dağıtım Ağı

Dünyanın birçok yerine dağıtılmış sunucuların oluşturduğu bir alt yapıdır ve CDN *(Content Delivery Network)*'in amacı ziyaretçilere, site içeriğini en hızlı şekilde ulaştırmaktır. Böylece kullanıcı tarafından çok uzak noktada yer alan bir web sayfası istendiğinde, o sayfa CDN'nin en yakın sunucusundan kullanıcıya iletilir. CDN'lerin içerik sağlayıcılarla doğrudan bir ilişki kurmaları gerekir.

Günümüzdeki CDN modelinde, içerik sağlayıcıları, CDN sağlayıcıları ve ağ operatörleri, CDN'lerin ana temelleri olarak tanınmaktadır. Üç taraf arasındaki ilişkiler, belli müşteri/sağlayıcı açık sözleşmelerine dayanmaktadır. CDN'lerin avantajlarına rağmen bazı eksikleri şunlardan ibarettir [15].

1- CDN Sunucularının nereye yerleştirileceği hususunda gerçek anlamda kontrol sağlanamaz çünkü bu iş birçok internet servis sağlayıcıların birlikte çalışmasını gerektirir.
2- CDN'ler yalnızca içerik sağlayıcılarla imzalanan anlaşmalarında yer alan belli uygulamaların trafiğini hafızasına alır ve kullanıcılara sunar.
3- Farklı CDN'ler arasında iletişim eksikliği bulunmaktadır. Örneği, bir organizasyonda farklı CDN servis sağlayıcılar tarafından yönetilen CDN'ler ve ara bağlantıları karmaşık ad-hoc mekanizmaların tasarımını gerektirir.

Yukarıda listelenen eksiklikler, telekomünikasyon operatörlerini ve ağ yöneticilerini, Internet servis sağlayıcılar İSS tarafından işletilen CDN'lere direk olarak girmesine tereddüt oluşturmaktadır. Ancak bu eksikleri ortadan kaldırmak amacıyla telekomünikasyon operatörleri ve ağ yöneticileri geleneksel ve transparent önbelleği ağlarına entegre ederek İSS'ler tarafından işletilen CDN'lere girmesine önem vermektedirler.

1.6.3. Veri Parçalarının Önbelleklenmesi

CDN sunucular bir biriyle koordine ederek dinamik dosyaların parçalarını (chunks) birçok CDN sunucularına yük dengeleme amacıyla dağıtır [17].

1.6.4. Şeffaf Önbellekleme

Geleneksel web önbelleklemede son kullanıcıların ayarları manuel olarak yapılırken şeffaf önbelleklemede son kullanıcı ayarları politika tabanlı yönlendirme *(Policy based routing)* tekniği ile yapılır [18]. Böylece, operatörler, neyi, nerede ve ne zaman önbellekleneceği konusunda tam kontrolü sağlar. Veriler, geleneksel web önbellek sunuculardan politika tabanlı yönlendirme tekniği ile son sunuculara iletilir. Ayrıca, operatörler ile içerik sağlayıcılar arasında anlaşmalar yapılmasını gerektirmez [15].

1.6.5. Bilgi Merkezli Ağlar

Bir kaynaktan bir hedefe paket göndermek yerine adlandırılmış veri alışverişi yaparak iletişimin gerçekleştiği yeni bir ağ paradigmasıdır. ICN *(Information Centric Networks)* ağlarında kullanılacak tüm yönlendiricilerin yerel önbellek ile donatılmış olması öngörülmüştür. Bu yerel önbelleklerin her birisi verilerin bir parçasını depolayabilirler. [19]. ICN'lerde kaynak sunucunun bulunduğu yere (örneği. IP adrese), uygulamaya ve dağıtım kanalına bakılmaksızın veriler benzersiz adlarla yönlendiricilerin önbelleğine alınır ve sunulur; böylece ağ içi önbelleğe alma/çoğaltma ve veri güvenliği sağlanır. ICN'den, karışık iletişim senaryolarında, gelişmiş veri yayma verimliliği ve dayanıklılığı açısından beklenen faydalar, IoT *(Internet of Things)* alanında, ICN'i inovatif bir ağ paradigması olarak belirtir [20].

ICN'lerde web önbellekleme işlemi, tüm yönlendiriciler tarafından yapılacaktır. Başka bir deyişle ağlarda kullanılan tüm yönlendiricilerin önbellekleme işlemini yapabilmesi için önbellek hafızasına sahip olması gerekir. Yönlendiricilerde önbellek bulunmasının yansıra adları kullanarak yönlendirme işlemini yapabilme özelliği de bulunması beklenmektedir.

Özet olarak literatürde önerilen [19] tüm mimariler aşağıda listelenen özellikleri taşımaktadır.

1- İçerik tabanlı adlandırma ve güvenlik.

2- Ağ içi önbellekleme.

3- Obje adlarına göre içerik keşfi ve dağıtımı.

4- Bağlantısız bir alıcı-odaklı iletişim modeli.

ICN hâlâ kâğıt üzerinde önerilen bir mimaridir. Günümüzde internette kullanılan donanımlar ve yazılımlar ICN mimarisi için uygun olmadığından bu yeni mimarinin uygulaması çok masraflı ve zaman alıcı bir süreç olarak öngörülüyor.

Web önbellekleme evrimine göre günümüzde CDN ile geleneksel ve şeffaf önbelleğin kullanılması en faydalı yaklaşım olarak değerlendirilmektedir. Ancak CDN'e bağlanan bir ağın içinde geleneksel ve şeffaf önbelleklerin nereye yerleştirileceği hâlâ bir sorun olarak yaşanmaktadır.

CAIDA[1] [21] kurumu tarafından yapılan analizlere göre internet HTTPS *(Hyper Text Transfer Protocol)* trafiği son on yılda dikkat çekici bir şekilde artmıştır. Bu HTTPS trafiğin artışı ağların bazı yerlerinde (örneğin bir yerel ağda kullanıcılar tarafından erişilen web sitelerin büyük kısmı HTTPS protokolü kullanan web siteler ise) web önbelleklerin kullanım önemini düşürmektedir. HTTPS, standart HTTP protokolün üzerine SSL/TLS *(Secure Sockets Layer/Transport Layer Security)* sertifikasının eklenmiş hâlidir. Başka bir deyişle HTTPS trafiği kullanıcı ve sunucu arasında 3. kişi tarafından okunmaması için veriler SSL/TLS protokolleriyle şifrelenerek gönderilir [22]. Şifrelenmiş veriler sadece sunucu ve kullanıcı arasındaki güvenli şekilde paylaşılan anahtar ile de-şifrelenebilir. Kullanıcı ve sunucu arasında yer alan web önbellek sunucuları üçüncü kişi olarak tanımlandığından HTTPS trafiği önbelleğe alınmayacaktır. Dolayısıyla şifrelenmiş verilerin önbelleklemesi ortadaki adam saldırısı *(man-in-the-middle attack)* olarak tanımlanır.

1.7. Web Önbellek Güncelleme Algoritmalar

Web önbelleklerin evriminde gösterilen tüm teknolojilerin ana bileşeni önbellek güncelleme algoritmalarıdır. Tüm teknolojilerin hafızasının güncellenmesi aynı mantığa dayandığı için geleneksel web önbelleklerde kullanılan algoritmalar başka teknolojilerde de kullanılmaktadır ve geleneksel algoritmalar gibi hepsinin performansı doğrudan önbellek

[1] 1997'de kurulan *Center for Applied Internet Data Analysis (CAIDA)*, ağ araştırmaları yürütür ve bilimsel araştırma topluluklarına büyük ölçekli veri toplama, kürasyon ve veri dağıtımını desteklemek için araştırma altyapısı oluşturur.

güncelleme algoritmalarına bağlıdır [23]. Bildiğimiz üzere tüm web önbellek sunucuları belli boyutta bir hafızaya sahiptirler. Önbellek güncelleme algoritmalar, web önbelleğin hafızası dolduğunda yeni objenin önbelleklenebilmesi için hafızadan hangi objenin silinip silinmemesine karar veren algoritmalardır. Web önbellek güncelleme algoritmalar, web önbelleğe alma algoritmalar (web caching algorithms) olarak da bazı kaynaklarda yer almıştır [24]. Etkin önbellek algoritmanın tasarlanması web önbelleklerin başarısının artmasına sebep olmaktadır [25]. Literatürde birçok web önbellek güncelleme algoritması önerilmiştir. Genel olarak tüm önbellek güncelleme algoritmalar beş kategori altında yer almaktadır [26] [23].

1.7.1. Yeniliğe Dayanan Algoritmalar

Kısa süre önce istenen nesneler tekrardan istenecektir düşüncesine dayanmaktadır. Aynı nesne yakın zaman aralığında birçok kullanıcı tarafından istendiğinde iyi performans göstermiştir. Avantajı, basit olması ve çok fazla işlemci gücüne ihtiyaç duymamasıdır. Dezavantajı ise, nesne boyutunu, nesne frekansını ve gecikmeyi dikkate almadan işlem yapmasıdır. LRU *(Least Recently Used)* algoritması bu kategorinin en özgün temsilcisi olarak tanınmaktadır. LRU algoritması uzun süre erişilmeyen nesneleri hafızadan kaldırır. Bu geleneksel algoritma, uygulamada en sık kullanılan algoritmadır [27].

1.7.2. Frekansa Dayanan Algoritmalar

En popüler (en büyük kullanım frekansına sahip) nesneler önbellekte saklanır. Bir ağda kullanıcılar tarafından istenilen objeler belli popülerliklere sahipse bu algoritma iyi sonuç verir. Avantajı, LRU gibi basit olması ve çok fazla işlemci gücüne ihtiyaç duymamasıdır. Dezavantajı ise, LRU'nun eksikliklerine ek olarak, bazen, frekansı birden fazla olan nesne sayısı az olduğu durumlarda, bazı kullanılmayan nesneler uzun zaman hafızada kalır. LFU *(Least Frequently Used)* algoritması bu kategorinin en özgün temsilcisi olarak tanınmaktadır. LFU algoritması, önbellekteki her nesne için bir frekans sayısı tutar ve hafızadaki herhangi bir nesne kullanıcılar tarafından tekrar istendiğinde o nesnenin frekans sayısına bir ekler. En düşük frekansa sahip olan nesneler hafızadan kaldırılmak için seçilir.

Birden fazla nesne aynı frekansa sahip olduğu takdirde hangisinin silineceğine ikinci algoritmanın (Örneği LRU algoritma) kuralları uygulanarak karar verilecektir.

1.7.3. Boyut Tabanlı Algoritmalar

Büyük nesneleri hafızadan kaldırarak daha küçük nesneleri önbelleğe saklar. Bilgi içeren web siteleri erişildiğinde genellikle iyi performansa sahiptir. Avantajı, büyük boyutta olan nesnelerin hafızadan kaldırılması daha küçük nesneleri tutulabilmesine yol açar ve böylece daha yüksek bir isabet oranı HR *(Hit Rate)* sağlar. Dezavantajı ise, küçük boyutta olan nesnelerin her zaman hafızada kalmasıdır ki daha düşük bayt isabet oranına BHR *(Byte Hit Rate)* neden olur. Büyük boyutta olan nesnelerin tekrar istenilmesi durumunda daha çok ağ kaynağı tüketilir. Bu kategorinin en özgün temsilcisi olarak tanınan "*Size* algoritması" hafızadaki nesneleri boyutlarına göre sıralar ve boyutu en büyük olan nesnelerin kaldırılmasına karar verir.

1.7.4. Fonksiyon Tabanlı Algoritmalar

Daha fazla parametreleri dikkate alarak çalışan algoritmaları içermektedir. Yüksek işlemci ve bellek kaynaklarına sahip sunucular kullanıldığında daha iyi performans gösterir. Bu kategorideki algoritmalar, nesnelerin erişim geçmişini analiz ederek bazı performans parametreler ve tahmin edilen gelecek erişimlerin trendiyle bir amaç fonksiyonu oluştururlar. Avantajı, birçok parametreyi kullanarak karar vermesidir ki önemli ölçüde HR oranını arttırır. Dezavantajı ise daha çok işlemci gücünü tüketmesidir [26]. Literatür de birçok fonksiyon tabanlı algoritma önerilmiştir ki GDSF *(Greedy Dual Size Frequency)* algoritması bu kategorinin en özgün temsilcisi olarak tanımlanabilir. GDFS algoritmasında bir öncelik kuyruk *(priority queue)* oluşturulur ve aşağıdaki Formula 1'i kullanarak her nesne için bir öncelik değer *(priority key)* üretilir. Önbellek hafızası dolduğunda en düşük öncelik değere sahip olan nesnelerin silinmesine karar verilir.

$$Pr(f) = clock + Fr(f) \times \frac{Cost(f)}{Size(f)} \quad (1)$$

Formül 1'de, *Pr(f), f* nesnenin öncellik değeri, *clock* kuyruğun saati olarak adlandırılmış bir değişkendir ki her nesne için başlangıç değeri sıfırdan başlar ve nesnenin silinmesiyle öncelik kuyrukta bu değer güncellenir. *Fr(f)* nesnenin frekansı, *Size(f)* nesnenin boyutu ve *cost(f)* ise nesne *f* ile ilişkin önbelleğe alma maliyettir. GDSF ile ilgili daha fazla bilgi için referans [28]'a bakınız.

Günümüzde, çeşitli yeni önbellek güncelleme algoritmaları önerilmiştir. Şuana kadar neredeyse önerilen tüm algoritmaların diğerlerinden daha iyi performans gösterdiği iddia edilmiş ve hangi algoritmanın kullanılmasının gerektiği konusunda bir karışıklık vardır [26]. Literatürdeki araştırmalara göre, belli bir algoritma yalnızca belirli ortamlarda iyi ve her algoritma farklı ağlarda farklı performans göstermektedir [29].

1.7.5. Rasgele Önbellek Güncelleme Politikalar

Karmaşık yapılara sahip algoritmalar, hiçbir karar kuralı gerektirmeden çalışan rasgele önbellek güncelleme politikaların ortaya çıkmasına neden olmuştur. Bu kategorinin özgün temsilcisi "RAND" politikasıdır. RAND politikası hiçbir karara bağlı olmadan nesneleri rasgele önbellekten kaldırır. Bu nedenle bu politika iyi sonuç vermemektedir [30]. Bu tür politikalar hem bellek hem de işlemci gücünü tasarruf etmektedir.

1.8. Veri Madenciliği

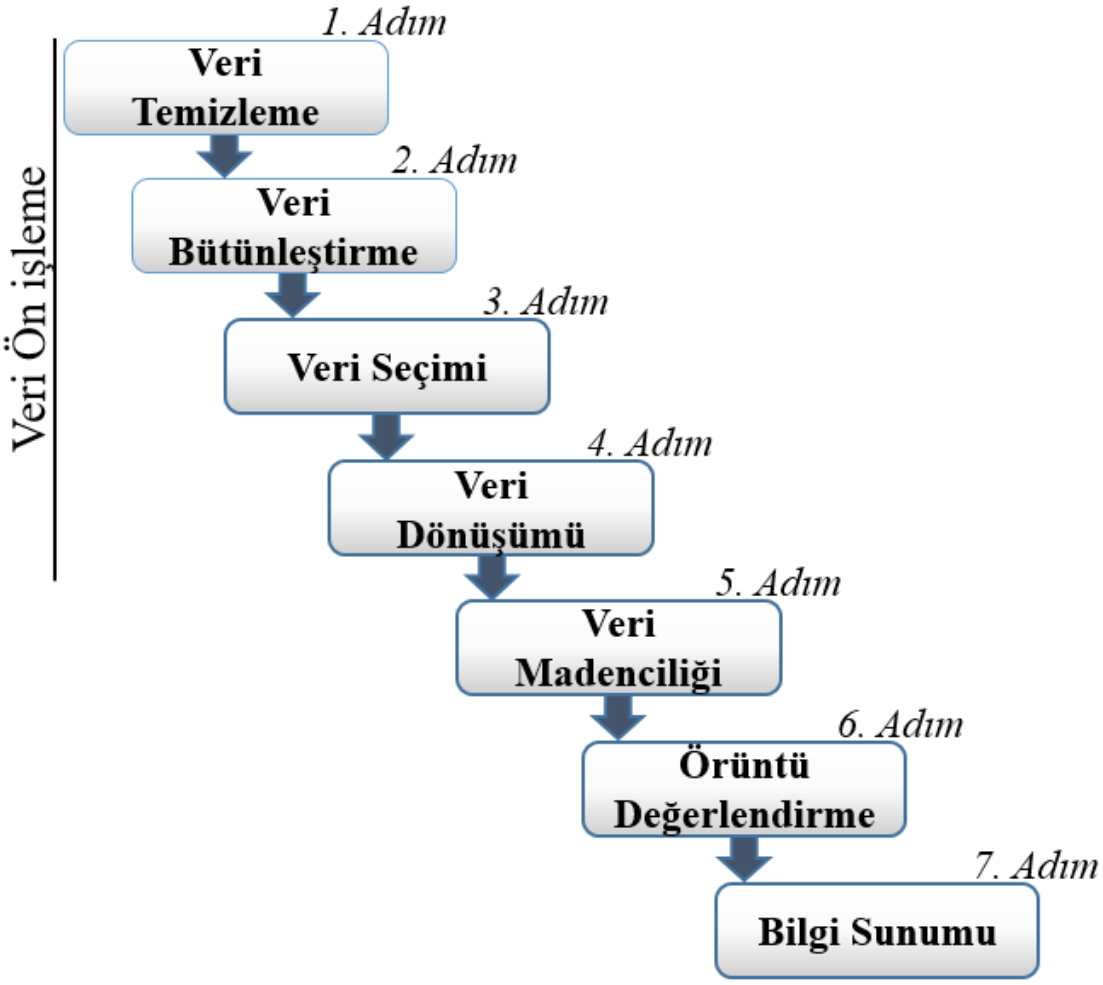
Basitçe belirtmek gerekirse, veri madenciliği, büyük miktarda olan veriden bilginin keşfi, çıkarılması veya "madenciliği" anlamına gelir [31]. Veya Veri madenciliği, büyük veri kümelerindeki faydalı bilgileri ve eğilimleri keşfetmesidir [32]. Günümüzde veri madenciliği birçok alanlarda kullanılmakta olup ve çok önemli öngörülmelere de kullanımı yaygınlaşmıştır. Örneği, MIT Technology Review raporu [33] 'na göre: Amerika başkanlık seçiminde Başkan Obama'nın kazanmasına veri madenciliği etkili bir şekilde kullanılmıştır. Obama kampanya ekibi, önce bir veri madenciliği modelini kullanarak muhtemel seçmenleri belirlemişler ve daha sonra belirleyen muhtemel seçmenlerin sandık başına gittiklerini doğrulamışlar. Obama Kampanya ekibi ayrıca her eyalete göre seçim sonuçlarını tahmin etmek için ayrı bir veri madenciliği modelini kullanmıştı. İki partinin de hemen hemen aynı sayıda taraftarı olan *Ohio* eyaletinde, model Obama'nın %57,68 oy getireceğini öngörmüştü; seçim sonuçlarına göre bu eyalette Obama'nın oy payı %57,16 olmuştu ki modelin hata oranı

sadece %0.52 orandaydı. Bu dakik tahmin, az kaynağın kampanya personeli tarafından çok verimli şeklinde paylaşılması ve kullanılmasını sağladı.

Birçok araştırmacılar, veri madenciliğini popüler olarak kullanılan KDD *(Knowledge Discovery from Data)* terimin eşanlamlısı olarak görüyor. Alternatif olarak, diğerleri veri madenciliğini, KDD sürecinde bir adım olarak görüyor. KDD aşağıdaki adımlardan oluşmaktadır [31].

1- Veri temizleme *(Data cleaning)*: Gürültüyü ve tutarsız verileri kaldırılması.
2- Veri bütünleştirme *(Data integration)*: Birden fazla veri kaynağının birleştirilmesi.
3- Veri seçimi *(Data selection)*: Üzerinde analize gerçekleşecek verilerin seçilmesi.
4- Veri dönüşümü *(Data transformation)*: Verilerin üzerinde özetleme veya toplama işlemleri gerçekleştirerek madencilik için uygun bir forma dönüştürülmesi veya birleştirilmesi.
5- Veri madenciliği *(Data mining)*: Akıllı yöntemleri kullanarak ver örüntülerinin çıkarılması.
6- Örüntü değerlendirme *(Pattern evaluation)*: Elde edilmiş bilgiyi temsil eden ilginç örüntüleri bazı ölçümlere göre tanımlanması.
7- Bilgi sunumu *(Knowledge presentation)*: Madencilik işleminin sonucunda elde edilen bilgilerin kullanıcıya sunulması.

Adım 1-4 veri önişleme *(Data preprocessing)* işleminin teknikleri olarak geçiyor, elbette bu işlemler verilerin madenciliğe hazırlanması için kullanılmaktadır. Bu teknikler birbirinden karşılıklı olarak münhasır değildir ve birlikte çalışabilirler. Örneğin, Veri temizleme, bir tarih alanının tüm girdilerini ortak bir biçimde dönüştürerek yanlış verilerin düzeltilmesi için dönüşümler gerektirebilir. Veri önişleme teknikleri, madencilik öncesi uygulandığında, ortaya çıkarılan örüntülerin genel kalitesini artırabilir ve / veya gerçek madencilik için gerekli süreyi önemli ölçüde azaltabilir. Böylece, veri önişlemi gerçekleşmeden veri madenciliği işleminin sonucu istenilen sonuç olmayıp veya veri üzerinde veri madenciliği işleminin yapılması mümkün değildir. Veri madenciliği sürecinde ilginç örüntüler kullanıcıya sunulur ve bilgi tabanında yeni bilgi olarak saklanır. Bu görüşe göre, veri madenciliği, değerlendirme için gizli örüntüleri ortaya çıkarması için KDD'nın en önemli bir adımı olarak da tınılabilir. Şekil 6'da tipik bir veri madenciliği sisteminin mimarisi gösterilmiştir.



Şekil 6. Tipik bir veri madenciliği sisteminin mimarisi

Veri madenciliğinin KDD sürecinde bir adım olduğuna rağmen günümüzde bu terim endüstride, medyada ve veritabanı araştırma ortamında, KDD'den daha popüler hale gelmektedir. Bu nedenle, bu tezde, veri madenciliği terimi kullanılmıştır.

Günümüzde yaygın şekilde kullanılan makine öğrenme *(machine learning)*, adlı bir terim daha var. Birçok insanlar su soruyla karşılaşıyorlar "Veri madenciliği ile makine öğrenme terimlerinin farkı nedir?" basitçe makine öğrenme, veri bilimleri *(data science)* bölümün bir alt bölümüdür ki genellikle algoritmaların dizaynı üzerinde ve tahmin yapma *(prediction)* işlemlerini yapar. Veri madenciliğinde makine öğrenme algoritmaları yaygın bir şekilde kullanılmaktadır [34].

Veri madenciliği, her tür veri seti üzerinde uygulanabilir örneği ilişkisel veritabanları *(relational databases)*, veri ambarları *(data warehouses)*, işlemsel veritabanlar *(transactional databases)*, gelişmiş veritabanı sistemleri *(advanced database systems)*, düz

dosyalar *(flat files)*, veri akışları *(data streams)* vb. Veri setlerin incelemesi bu tezin kapsamı dışındadır.

Yıllar boyunca büyük veri setlerinden değerli bilgilerin çıkarılması için birçok algoritmalar oluşturulmuştur. Bu algoritmalar yapacak işlemlerine göre farklı kategorilere sınıflandırılmıştır. Örneğin, Sınıflandırma *(classification),* kümeleme *(clustering),* Birliktelik kuralları *(association rules)* vb. [35]. Sınıflandırma *(classification)* verilerden bilgi çıkarmak ve verilerin ortak özelliklerine göre sınıflara ayırabilme kabiliyetine sahip olan ve günümüzde çok yaygın bir şekilde kullanılmakta olan veri madenciliğinin bir metodudur. Sınıflandırma iki alt kategoriye ayırılmıştır. 1- Gözetimli sınıflandırma *(supervised classification)*: Gözetimli sınıflandırma algoritmaları, verilen dağılım şeklini eğitim kümesinden ve daha sonra sınıfının belirli olmadığı test verileri geldiğinde doğru şekilde sınıflandırmaya çalışırlar. Yanı segmantasyon verilerin benzerlik kavramlarına göre yapılacaktır. 2- Gözetimsiz sınıflandırma *(unsupervised classification or clustering)*: Genellikle bu alt kategori kümeleme *(clustering)* olarak geçmiştir. Kümeleme algoritmalarının, verilerin dağılım şekli ile ilgili önceden hiç bilgisi yoktur ve segmantasyon işlemini veri setin değişkenlerinin benzerliklerine göre yapacaktır [36]. Her kategori birçok algoritmaları içeriyor ve gün geçtikçe yeni algoritmalar onarılarak bu algoritmaların sayısı artmaktadır. Bu tezde *k-means* kümeleme algoritması kullanıldığı için bu bölümde *k-means* algoritmalarının incelemesine yer verilmiştir.

1.8.1. K-Ortalama (*K-Means)* Kümeleme Algoritması

Bu algoritma 1967 yılında *J.B MacQueen* [37] tarafından önerilen en eski ve günümüzde veri madenciliği dünyasında en çok kullanılan ve en popular kümeleme algoritmasıdır. Kümeleme, veritabanda yer alan verileri alt kümelere ayırmak işlemine denir. Kümeleme alanındaki benzer çoğu algoritmalar bu algoritmanın geliştirilmesiyle ya da bu algoritmadan esinlenerek önerilmiş ve ortaya çıkmıştır [38]. Her kümede aynı özelliklere sahip olan elemanlar yer alacaklardır. K-ortalama algoritmasının atama mekanizması göre, her verinin sadece bir kümeye atanmasına izin verilmesi nedeniyle bu algoritma keskin bir kümeleme algoritmasıdır. Bu metodun ana fikri merkez noktanın kümeyi temsil edilmesidir ve eşit boyutlarda küresel kümeleri bulmaya eğilimlidir [39]. K-ortalama algoritmasının genel mantığı, bir veritabanda yer alan n adet veri nesnesini, önceden belirtilen k adet kümeye bölümlemektir. Amaç, gerçekleştirilen kümeleme işlemi sonunda elde edilen

kümelerin, küme içi benzerliklerinin maksimum ve kümeler arası benzerliklerinin minimum olmasını sağlamaktır [40].

K-ortalama kümeleme algoritmanın değerlendirilmesi için çok yaygın bir şekilde karesel hata kriteri SSE *(sum of squared errors)* kullanılmaktadır. SSE değeri düşük olan kümeleme sonuçları en iyi sonuç olarak kabul edilir. Her kümede nesneler ve merkez noktanın arasında olan uzaklıkların karelerinin toplamı aşağıdaki 2 nolu eşitlikle hesaplanmaktadır [41].

$$SSE = \sum_{i=1}^{K} dist^2(m_i, x) \tag{2}$$

Bu eşitlikten beklenen sonuç, k tane kümenin mümkün olduğu kaderince yoğun ve birbirinden ayrı sonuçlanmasıdır. K-ortalama algoritması, n tane veriden oluşan veri setini kullanıcı tarafından verilen k harfi ile belirtilen k adet kümeye bölmeye çalışır. Küme merkezleri kümelerin birbirinden ayıran değerdir ve bu değer kümedeki nesnelerin ortalama değeri ile hesaplanır [42].

1.8.2. İstatistiksel Normalleştirme

İstatistiksel normalleştirme, özellikle, veri madenciliği *(data mining)* gibi bilgisayar bilimlerinin istatistiksel veri işleme alanlarında sık kullanılan bir yöntemdir. Bazen veriler çok büyük veya birbirlerinden çok farklı olabilir. Bu yüzden istatistiksel normalleştirme, hesaplama kolaylığı için verileri belirli bir aralığa çekme işlemidir. Diğer bir kullanılışı ise farklı ölçekleme sisteminde bulunan verilerin birbiri ile karşılaştırılabilmesidir. Buradaki amaç, matematiksel fonksiyonlar kullanarak, farklı sistemlerde bulunan verileri, ortak bir sisteme taşımak ve karşılaştırılabilir hale getirmektir [43].

İstatistiksel normalleştirmede farklı normalleştirme fonksiyonları kullanılmaktadır.

1.8.2.1. Asgari – Azami Normalleştirmesi

Bu yöntemde, bir grup verinin içerisindeki en büyük ve en küçük değerler ele alınır. Diğer bütün veriler, bu değerlere göre normalleştirilir. Buradaki amaç en küçük değeri 0 ve

en büyük değeri 1 olacak şekilde normalleştirmek ve diğer bütün verileri bu 0-1 aralığına yaymaktır.

1.8.2.2. Standart Skor

Diğer bir normalleştirme yöntemidir. Bir önceki yöntemde, sayılar en yüksek ve en düşük değerlere göre normalleştirilmişti. Bu yöntemde ise ortalama değer *(mean value)* ve standart sapma *(standard deviation)* değerleri göz önüne alınır. Sistemde kullanılan standart sapmaya atfen, standart skor *(standard score)* olarak da isimlendirilir. Oldukça popüler normalleştirme yöntemlerinden birisidir.

1.9. Ağ İzleme Programları

Ağ izleme, teknoloji ve bilişim alanlarında sıkça kullanılan bir IT *(Information Technology)* terimidir. Ağ izleme, özelleştirilmiş yönetim yazılımı araçlarını kullanarak bir bilgisayar ağının çalışmasını nezaret etme anlamına gelir. Ağ izleme sistemleri, bilgisayarlar ve ağ servislerinin kullanılabilirliğini ve genel performansının standart olarak çalışmasının sağlamak için kullanılır. Ağ yöneticileri ağ izleme programlarıyla erişimi, yönlendiricileri, güvenlik duvarları, ağ anahtarları, sunucuları ve yavaş veya başarısız bileşenleri izleyebilirler. Ağ izleme sistemleri genellikle büyük ölçekli kurumsal ve üniversite ağları gibi geniş alan ağlarda kullanılır [44].

Ağ izleme programları, ağ kullanılabilirliği, bant genişliği kullanımı, servis kalitesi, bellek yükü ve işlemci kullanımı gibi parametrelerin izleme işlemini uzaktaki cihazlar üzerinde bile yapar. Ağ izleme programları, bağlantıların, yönlendiricilerin, güvenlik duvarlarının, sunucuların ve diğer ağ bileşenlerinin verimliliğini, düzenini ve kurulumunu optimize etmek için sistem yöneticilerine canlı ve periyodik kullanım rapor ve grafikleri sunar [45].

1.10. EIGRP Bağlantı Değeri

EIGRP yönlendirme algoritması en iyi yolu teşhis etmesinde veya bağlantı değerinin hesaplamasında aşağıda verilen metrik parametrelerini kullanır. EIGRP en düşük bağlantı değere sahip bağlantıyı en iyi yol olarak seçer [46].

Bandwidth (K1)	Minimum bant genişliği *(Lowest bandwidth of route)*
Load (K2)	Paket oranına göre rotadaki en büyük yük *(Worst load on route, based on packet rate)*
Delay (K3)	Hedef ağ rotasındaki toplam gecikme *(Cumulative interface delay of route)*
Reliability (K4)	*keep alive* veriye göre rotanın en kötü güvenilirliği (*Worst reliability of route based on keep alive)*
MTU (K5)	Rotadaki en küçük MTU *(Maximum Transmission Unite) (Smallest MTU in path)*

Varsayılan olarak EIGRP'de hedef ağa giden rotadaki bağlantıların minimum bant genişliği (K1) ve toplam gecikmesi (K3) kullanılır. Gerçi ağ yöneticisi olarak EIGRP'yi konfigürasyon yaparken başka metrikleri de kullanabilirsiniz ancak ağda döngüler oluşturabileceğinden dolayı önerilmemiştir [47]. Bant genişliği ve gecikme metrikleri, hedef ağa giden rotada yönlendiriciler interface'nde yapılandırılan değerlerden tespit edilir.

1.11. Zipf Kuralı

Dilbilimci George Kingsley Zipf tarafından 1949 yılında önerilen zipf yasasına göre, belli bir dilde yazılan bir eserde az sayıda birtakım sözcük her zaman kullanılıyorken sözcüklerin büyük bir çoğunluğu düşük frekansla kullanılıyor. Sözcükler kullanım frekansına göre sıralandığında çarpıcı bir örüntü ortaya çıkıyor. Birinci sırada yer alan sözcük, ikinci sıradaki sözcüğün hep iki katı kadar sıklıkta, üçüncü sıradaki sözcüğün de hep üç katı kadar sıklıkta kullanılıyor olması fark ediliyor. Zipf 1949 [48]'den buyana, frekans ve sıra rakamlarının normal logaritmasına göre grafik oluşturarak verilerin Zipf yasasına uyup uymaması tespit edilmektedir. Oluşturulan grafiğin eğimi her ne kadar eksi bir (-1) rakamına yakınsa verilerin Zipf yasasına uygunluğu yüksek demektir. Elbette bazı örnekler, denkleme bu kadar iyi uymaz, ama örnekler logaritmik grafiğe oturtulduğunda, şaşırtıcı derecede düz bir çizgi verir.

Zipf yasası şehirler üzerine uygulandığında, kentlerin nüfuslarını en büyüğünden en küçüğüne doğru sıralayıp X koordinatında kent büyüklüklerinin logaritması ve Y koordinatında ise bunlara karşılık gelen sıralamanın logaritması alındığında, eğimin -1'e eşit veya çok yakın bir rakam olduğu beklenmektedir [49].

Zipf yasası, şirket büyüklükleri sıralamaları, gelir sıralamaları, web sayfalarının ziyaretçi sıralamaları vb. birçok örnekler için de uygulanmıştır.

Zipf yasası geçen yüzyıldan beri bir sır olarak devam etmektedir. Zipf Yasası'nı şehirlere uygulama hakkında 2006 yılında yazı yazan Paul Krugman herkesçe çok iyi bilinen şu ifadeleri kullanıyor:

"Ekonomi kuramıyla ilgili genel şikâyet modellerimizin çok basitleştirilmiş olduğu yönündedir, yani modellerimizin karmaşık ve dağınık gerçeği fazlasıyla düzenli bakış açılarıyla sunduğundan yakınılır. [Zipf Yasası'nda ise] tersi doğrudur: Karmaşık ve dağınık modellerimiz var, buna karşılık gerçek şaşırtıcı şekilde düzenli ve basittir."

1.12. Kullanılan Veri Seti

Önerilen web önbellek güncelleme algoritmanın ve önbellek yerleştirme metodunun performansını ölçmek için, Boston üniversitesi tarafından sunulan veri setleri kullanılmıştır [50]. Sunulan veri seti altı ay boyunca 762 farklı kullanıcıdan gelen 9633 dosya ve 1143839 web isteği içermektedir. İz kayıt dosyaları, kullanıcıya yerel web önbellekten ve internetten erişimi sağlanan WWW nesnelerden oluşmaktadır. Veri setindeki her iz kaydına ilişkin, one-way fonksiyonu ile dönüştürülmüş bir kullanıcı ID'si *(Identity Document)*, kullanıcı tarafından kullanılan cihazın ID'si, isteğin gerçekleştiği zaman kaydı, istenilen nesnenin URL'i *(Uniform Resource Locator),* boyutu ve indirme süresi saniye olarak bulunmaktadır.

Trafik izleri kaydedilen Boston üniversitesinde toplam 37 bilgisayar bulunmaktadır, bilgisayarlar yüksek lisans laboratuvar ve lisans laboratuvarı olarak iki sete ayırılmıştır. Yüksek lisans laboratuvarında 32 bilgisayar ve lisans laboratuvarında ise beş bilgisayar yerleştirilmiştir. Bu yazıda lisans laboratuvarındaki bilgisayarların trafik iz kayıtları veri seti L ve yüksek lisans laboratuvarındaki bilgisayarların trafik iz kayıtları ise veri seti Y olarak adlandırılmıştır.

1.13. Literatürdeki Çalışmalar

Literatürde web önbellek güncelleme algoritmaları ile ilgili birçok çalışma bulunmaktadır, ancak önerilen algoritmalarda; bir, iki ve en fazla üç parametre kullanılmıştır. Hâlbuki algoritmaların performansına doğrudan etkisi olan birçok parametre bulunmaktadır.

Bilgimiz dâhilinde, literatürde, internet trafiğini ve kullanılan protokolü dikkate alarak geniş alan ağlar veya İSS'lerde web önbelleklerin yerleştirmesi için yapılan bir çalışma bulunmamaktadır.

Burada web önbellek güncelleme algoritmaları ile ilgili yapılan çalışmalar özetlenmiştir.

Khaleel ve ark. [29] önerdiği ALFUR *(Average Least Frequency Used Removal)* adlı web önbellek güncelleme algoritması frekans, boyut ve zaman parametrelerine bir ortalama değer hesaplanarak çalışır. Bu algoritmanın çalışma mekanizmasına göre her objenin frekans, boyut ve zaman parametreleri için birer değer hesaplanır ve objeler değerlerine göre önbellekten silinir. Örneğin eğer objenin frekans değeri hesaplanan ortalama değerden büyük ise web önbellekten silinmez. Frekans değeri hesaplanan ortalama değeri ile eşit olduğunda, objenin boyut değerine müracaat edilir. Objenin boyut değeri ortalama değerden küçük ise objenin silinmesine karar verilir, eğer ortalama değer ile eşit ise zaman değerine göre objenin önbellekten silinip silinmeyeceğine karar verilir. Bu işlemler algoritmada belirtilen dört ajan ile yapılmaktadır. Çalışmada gösterilen simülasyon sonuçlarına göre ALFUR algoritmanın performansı farklı web önbellek boyutlarında bazen LFU algoritmanın performansından düşüktür.

Noriaki ve ark. [51]'de CDN'ler için orijin sunucu ile web önbellek arasındaki mesafeyi *(hop counts)* dikkate alarak bir web önbellek güncelleme algoritması önermişlerdir. Önerilen algoritmada, önbelleğin hafızası birçok sanal önbellek olarak bölünmüştür ve her sanal hafızada yer alan nesneler orijin sunucusunun mesafesine göre ayrı ayrı yönetilir. Yazar önerilen algoritmanın üç mevcut önbellekleme algoritmalarıyla *(LRU, GreedyDual-Size ve GreedyDual)* kıyaslayarak bağlantı yükünü %10'dan daha fazla azaltabileceğini iddia etmektedir. Makalede sunulan simülasyon sonuçlarına göre önerilen algoritmanın isabet oranı LRU algoritmasından daha düşük olduğu görülmektedir. Ayrıca, bazen önbellekten çok uzak olmayan orijin sunucular ve önbellek arasındaki bağlantıların yükü çok fazla olduğu için veri erişimi çok yavaş olabilir. Böyle bir durumlarda yakın orijin sunuculardan

istenilen nesnelerin erişiminin arttırmak konusunda bu algoritma iyi performans göstermeyebilir.

Andre ve ark. [52]'de SACS *(Semantics Aware Caching System)* adlı bir algoritma önermişler. SACS sistemi, mesafe, yenilik ve frekans metriklerini kullanarak çalışır. Yakınlık ve frekans metrikleri, geleneksel LRU ve LFU algoritmalarında kullanıldığı gibi kullanılmıştır ama mesafe adlı yeni parametre objelerin URL'lerinden hesaplanır. Mesafe metriği, önbellek hafızasında bir web sayfasının birçok objeleri mevcut olduğu zaman her objenin mesafe metriği objelerin URL'lerinden en son indirilen objenin URL'ne göre hesaplanır. En son objenin URL'ne yakın objeler önbellekte saklanır ve mesafesi en son objeye göre uzak olan objeler hafızadan kaldırılır. Bu algoritma objelerin silinip silinmemesine karar verirken frekans metriğini de dikkate alır. Makalede sunulan simülasyon sonuçlarına göre küçük hafıza boyutlarında, LFU algoritması SACS algoritmasından daha iyi sonuç vermektedir. Ayrıca önerilen birçok algoritmalar gibi SACS algoritması da gecikmeyi dikkate almamaktadır.

Waleed ve ark. [53]'da LRU algoritması üzerinde destekçi vektör makinesi (support vector machine), naif Bayes sınıflandırıcısı ve bir karar ağacı gibi popüler makine öğrenme algoritmaları uygulayarak bir algoritma önermişlerdir. Bu algoritma LRU algoritmasının performansını arttırmak amacıyla önerilmiştir. Ve simülasyon sonuçlarına göre LRU algoritmasından daha iyi performans göstermektedir.

Hiba ve ark. [54]'de bulanık mantığına dayalı çalışan algoritmayı önermiştir. Önerilen algoritma iki tarafta (Parent ve Child) uygulanır. Bu algoritma LRU, LFU ve "*Size* algoritmaları"nın birleştirilmesi ile geliştirilmiştir ve akıllı bir karar almak için bulanık mantığı kullanılmıştır. Simülasyon sonuçlarına göre bu algoritma büyük boyutu olan önbelleklerde HR ve BHR oranı açısından, LRU, LFU ve size algoritmalarından iyi sonuç vermektedir.

Samiee [55]'de adaptif bir önbellek güncelleme algoritması önerilmiştir. Bu algoritma, sistem üzerinde düşük yüke sahiptir ve uygulanması kolaydır. Bu model, üç faktöre göre önbellekteki sayfaların sıralamasına dayanan Ağırlık Değiştirme Politikası (WRP) olarak adlandırılmıştır. Bir objenin silinmesi gerektiğinde en düşük sıralama değerine sahip bir sayfa yeni istenilen sayfa ile değiştirilmesi için seçilir. Bu modelin en büyük avantajı, hem LRU hem de LFU'ya benzerliğidir ki bu, her ikisinin de sunduğu faydalardan oluşuyor. Bu algoritma LRU ve LFU algoritmalarına göre çok az bir fark ile iyi sonuç göstermektedir ve literatürdeki çalışmaların bir-çoğu gibi gecikme faktörünü dikkate almamaktadır.

Megiddo ve Modha [56]‘de, değişen erişim modellerine dinamik olarak yanıt veren, iş yükü ve yenilik ile frekans özellikleri arasındaki dengeyi sürekli olarak dengeleyen LRU’dan daha iyi performans gösteren ARC *(Adaptive Replacement Cache)* uyarlanabilir yedek önbellek algoritması önerilmiştir. Bu algoritma sadece LRU algoritmasından daha iyi sonuç üretmektedir. Bir çok simülasyon sonuçlarına göre LFU ve Size gibi geleneksel algoritmalar da bazı ağlarda LRU‘dan daha iyi sonuç vermektedir.

Literatür tarama özeti olarak, önerilen algoritmalarda bir, iki ve en fazla üç faktör dikkate alınmıştır ve genellikle gecikme faktörü kullanılmamıştır. Bu eksikliğin yansıra, literatürde önerilen algoritmalar belirli bir önbellek boyutu ve belirli ağlarda sadece HR ve BHR oranlarına göre iyi sonuç vermektedir ve hiç-birisi tüm ağlarda HR, BHR ve DR oranlarına göre iyi sonucu vermeyi garanti etmemektedir.

1.14. Tezin Yapısı ve Katkıları

Bu makale iki çalışmadan oluşmaktadır. Birinci çalışmada, ağ trafiğinin analizi yapılarak objenin dört (yenilik, frekans, boyut ve gecikme) parametresini dikkate alan ve her ağda sadece HR ve BHR oranı açısından değil belki gecikme oranı DR *(Delay Ratio)* açısından da en iyi performansı gösteren yeni önbellek güncelleme algoritması önerilmiştir.

Önerilen önbellek güncelleme algoritmasında ağın analizi için veri madenciliğinde kullanılan K-ortalama kümeleme algoritması ve objelerin puanlaması için ise istatistiksel normalleştirmede kullanılan formüller manipüle edilerek kullanılmıştır. Önerilen algoritma her ağda en iyi sonucu vermeyi sadece HR ve BHR oranları açısından değil belki DR oranı açısından da başka algoritmalara göre daha iyi sonuç vermeyi garanti etmektedir.

İkinci çalışmada ise, ağ trafiği, bant genişliği ve bağlantıların yükü dikkate alınarak önbelleklerin en uygun yerlerde yerleştirilmesi için bir yöntem önerilmiştir. Web önbellek yerleştirilmesi, EIGRP *(Enhanced Interior Gateway Routing Protocol)* yönlendirme protokolün bağlantı maliyetinin *(link cost)* hesaplamasında kullanılan formüllerin mantığına dayanarak yapılmıştır. EIGRP’te bağlantı maliyet hesaplaması en iyi yol seçebilmek için yapılırken bu çalışmada hesaplanan bağlantı maliyet değerleri POP’lara öncelik değer hesaplamak için kullanılır.

Tezin birinci bölümünde genel bilgiler baslığı altında iletişim ağlar, web önbellekler, web önbelleklerin evrimi ve önbellek güncelleme algoritmalarının önemi günümüzdeki problemleri ile birlikte incelenerek literatürde bulunan ilgili çalışmaların açıklaması ile

bitmiştir. İkinci bölümde ise önerilen web önbellek güncelleme algoritmanın ve optimal web önbellek sunucu yerleştirme metodunun metodolojisi açıklanmıştır; üçüncü bolüm, bulgular ve tartışmaları içerir. Ve son bölümde sonuçlar sunulmuştur.

2. YAPILAN ÇALIŞMALAR

2.1. Giriş

Günümüzde web önbelleklerin evrimindeki tüm teknolojilerde yaşanan ortak problem ise önbellek sunucuların en uygun noktalarda yerleştirilmesidir. Geleneksel olarak web önbellek sunucuları, yerel ağın geçidine (Network Gateway) yerleştirilir ve geniş alan ağlarda ise birçok yerel alan ağın internete bağlanmak üzere birleştiği noktada kurulur. HTTPS trafiğin artması ve bazı ağlarda daha güvenli bir şekilde internete bağlanmak için VPN *(Virtual Private Network)* teknolojisi gibi yöntemlerin kullanılması, geniş alan ağlar ve İSS'lerde web önbelleklerin daha uygun yerlerde yerleştirilmesini ciddi problem olarak ortaya çıkartmaktadır. Geniş alan ağlarda web önbellek sunucuların uygun yerlerde kurulması hem ağ maliyetini verimli etkiler hem de gecikmeyi azaltarak kullanıcıların memnuniyetini arttırır.

Web önbelleklerin performansına doğrudan etkisi olan faktör ise web önbellek güncelleme algoritmalardır. Günümüzde, çeşitli yeni önbellek güncelleme algoritmaları önerilmiştir. Şuana kadar neredeyse önerilen tüm algoritmaların diğerlerinden daha iyi performans gösterdiği iddia edilmiş ve hangi algoritmanın kullanılmasının gerektiği konusunda bir karışıklık vardır [26]. Literatürdeki araştırmalara göre, önerilen birçok web önbellek güncelleme algoritmalarında, en fazla üç metrik kullanılmıştır ve genellikle gecikme metriği dikkate alınmamıştır. Hâlbuki web önbelleklerin çok önemli özelliklerinden birisi algılanan gecikmenin azaltmasıdır. Ve belli bir algoritma yalnızca belirli ortamlarda iyi ve her algoritma farklı ağlarda farklı performans göstermektedir [29].

Web önbelleklerde yaşanan web önbellek sunucu yerleştirme problemi ve sunucuların hafızasının daha verimli şekilde güncellenmesi için bu tezde iki çalışma yapılmıştır. Birinci çalışmada, ağ trafiğinin analizi yapılarak objenin dört (yenilik, frekans, boyut ve gecikme) parametresini dikkate alan ve her ağda sadece HR ve BHR oranları açısından değil belki DR oranı açısından da en iyi performansı gösteren yeni dinamik web önbellek güncelleme algoritması önerilmiştir. Önerilen önbellek güncelleme algoritmasında Objelerin puanlaması için ise istatistiksel normalleştirmede kullanılan formüller manipüle edilerek kullanılmıştır. Ve ağ verilerinin analizi için K-ortalama algoritmasında küme merkezlerini hesaplamak için kullanılan Öklid uzaklık metodu kullanılmıştır. Öklid uzaklık metodu ile puanlanmış önceki

trafik kayıtların yakınlık, frekans, boyut ve gecikme metriklerine ağırlık hesaplanır. Hesaplanan ağırlıklar parametrelerin web önbelleklerde sakladığı önemi gösterir.

İkinci çalışmada ise, ağ trafiği, bant genişliği ve bağlantıların yükü dikkate alınarak web önbellek sunucularının en uygun yerlerde yerleştirilmesi için bir yöntem önerilmiştir. Web önbellek yerleştirilme yönteminde, EIGRP yönlendirme protokolün bağlantı maliyeti hesaplaması gibi her önbellek kurulabilecek noktaların bağlantı/bağlantılarına bir değer hesaplanır. Bu hesaplanan bağlantı değeri ve ağ verileri kullanılarak her web önbellek kurulabilecek nokta için bir öncelik değer hesaplanır. Hesaplanan öncelik değerler hangi noktanın web önbellek kurulması için daha önemli olduğunu gösterir. Böylece geniş alan ağlarda web önbellek sunucuları kurulurken en uygun noktalarda kurulması sağlanır ve kurulan önbellekler kullanıcıların erişim hızını iyileştirirken bant genişliğini de maksimum miktarda tasarruf eder.

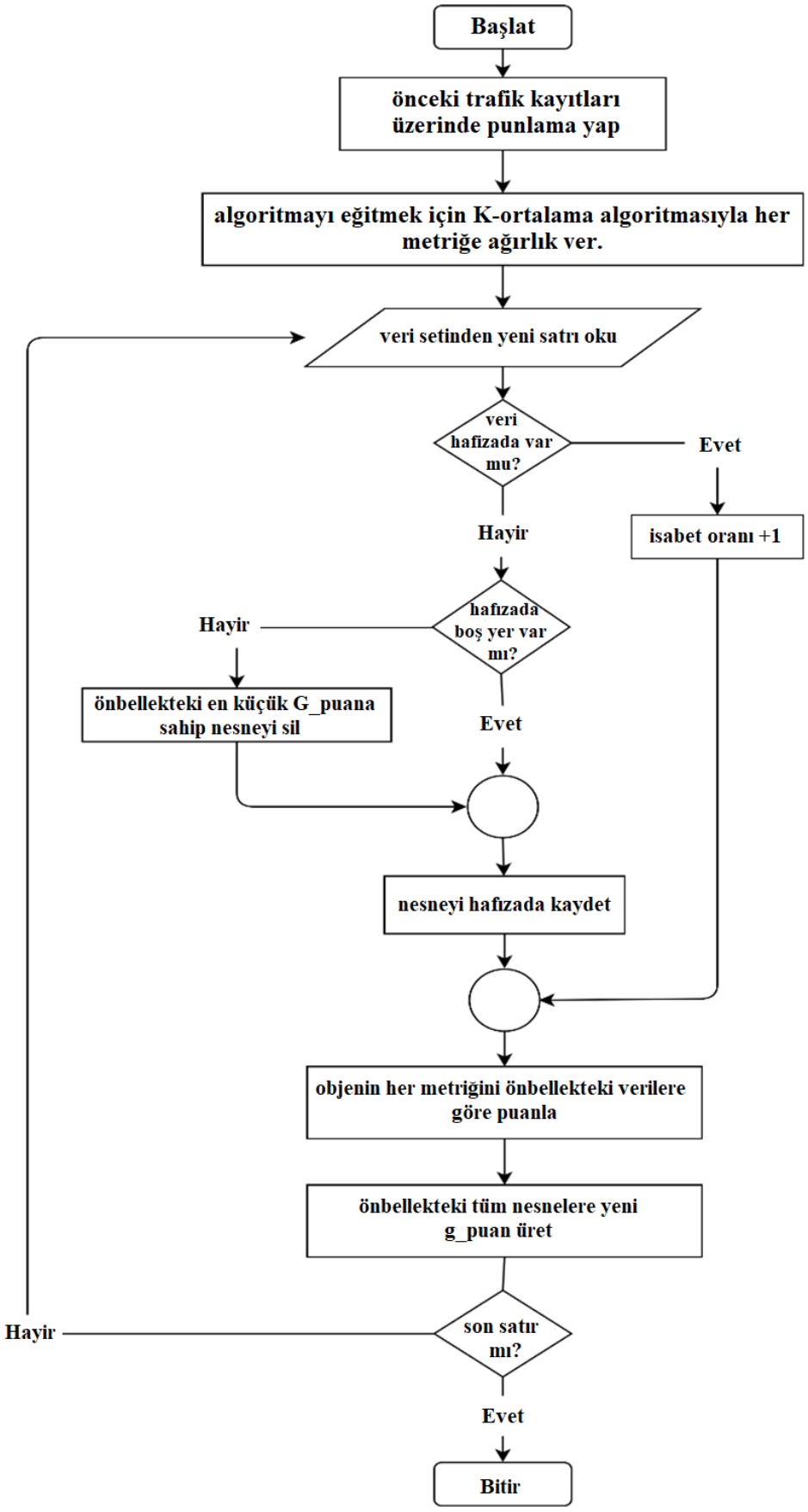
2.2. Dinamik Web Önbellek Güncelleme Algoritması

En sık kullanılan web önbellek güncelleme algoritmaları web trafiğinin yalnızca bir parametresini dikkate aldığı ve web önbelleğin verimliliğini etkileyebilecek diğer faktörleri yok saydığından yeterince verimli değildir ve önbellek kirliliği sorununu yaşatabilirler [57]. Bu makalede önerilen web önbellek güncelleme algoritması belli bir zaman aralığında ağın önceki iz kayıtlarının RFSD *(Recency, Frequency, Size ve Delay)* metrikleri üzerinde puanlama yapar ve K-ortalama algoritmasıyla ağ trafiğini analiz eder. Metriklerin önbelleğin performansında olan etkisine göre (k-ortalama algoritmasıyla hesaplanan ortalamalara göre) RFSD metriklerine ağırlık verir. Belirleyen ağırlıklara göre her obje için bir G_puan (General puan) hesaplanır. Algoritma objelerin önbellekten kaldırma işlemini G_puanlara göre yapar. RFSD metriklerin puanlaması, istatistiksel normalleştirme formülünün manipüle edilmiş formül 4 ve 5 ile yapılmaktadır.

Ağ trafiğinin analizi, veri madenciliğinde yaygın olarak kullanılan k-ortalama algoritması ile yapılır. Ağ trafiğinin analizi, önerilen algoritmanın her ağda optimum performans göstermesini sağlar. Algoritma ve algoritmanın akış diyagramı şekil 7 ve 8'de gösterilmiştir.

```
1    Get previous logs and score the objects using normalization formula
2    Run K-means algorithm on scored logs to analyze the network and to training the
     algorithm by giving weights to each metric
3    Read input URL
4    if file in cache? Then
5         record hit
6         else if enough space? Then
7              place file in the cache
8              else remove objects with the lowest general score from the cache
9                   place file in the cache!
end
10   Score each metric of the object according to the logs;
11   generate a general score for each object inside the cache memory;
12   if no other input?
     end
13    else Read new input
```

Şekil 7. Önerilen algoritma



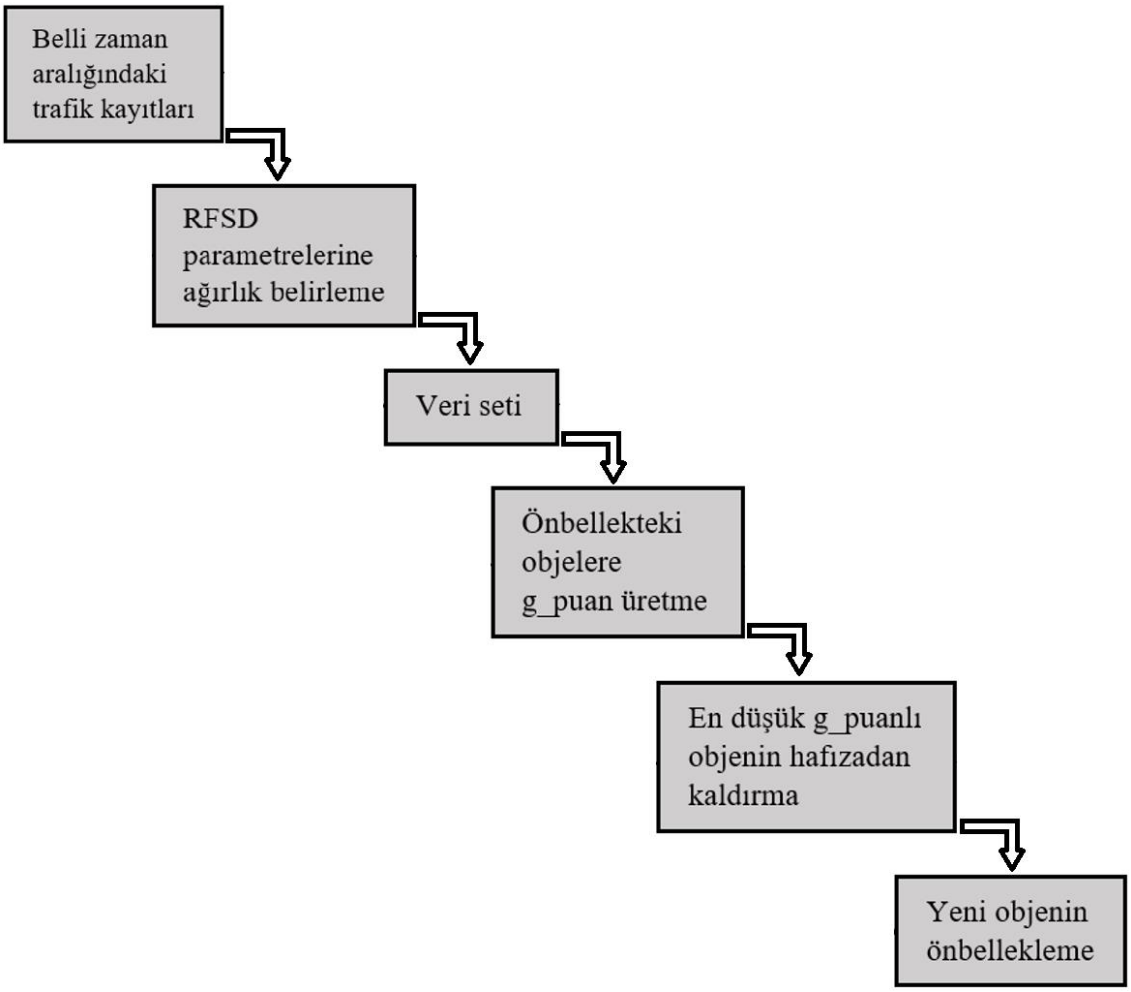
Şekil 8. Önerilen algoritmanın akış diyagramı

2.2.1. Dinamik Web Önbellek Güncelleme Algoritmasının Sistem Modeli

Genel olarak internette üç tür web önbellek kullanılmaktadır [12]. Bu çalışmada, kullanıcılar ve web sunucuları arasındaki gecikmeyi azaltmak, bant genişliğini daha verimli kullanılması konusunda kilit rol oynayan ve en yaygın kullanılmakta olan web önbellek sunucular söz konusudur. Şekil 2'de gösterildiği gibi, web önbellek sunucular, istemci cihazı ve orijinal sunucular arasında bulunur.

Web önbellek sunucuları tarayıcı önbelleği ile aynı prensibe dayalıdır, ancak tarayıcı önbelleği bir istemciye hizmet verirken web önbellek sunucular birden çok istemciye hizmet verir. Web önbellek sunucusu bir istek alındığında, ilk önce kendi hafızasını kontrol eder. İstenilen belge hafızasında mevcutsa, belgeyi istemciye gönderir. Eğer istenilen belge hafızasında mevcut değilse veya geçerliliği kalkmış ise önbellek sunucusu isteği orijin sunucuya iletir ve belgeyi aldıktan sonra istemciye gönderir. Belgenin bir kopyası önbellek hafızasına gelecek istekler için kaydedilir. Önbelleğin hafızası dolduğunda web önbellek güncelleme algoritmaları hangi objenin silinmesine karar verir. Web önbellek sunucusunun çalışma prensibi şekil 3'te grafiksel olarak gösterilmiştir [12].

Dinamik web önbellek güncelleme algoritması her ağda iyi performans göstermek amacıyla önceki trafik kayıtları üzerinde puanlama yaparak hangi parametrenin önbelleğin performansına ne kadar önemli olduğunu hesaplar. Bu hesaplamalara göre algoritma parametrelerini ağ trafik özelliklerine göre ayarlar. Önbelleğin parametre ayarlama işlemi ağ yöneticisi tarafından belirleyen belli aralıklara veya belli zamanlarda yeniden hesaplanır. Şekil 9'da dinamik web önbellek güncelleme algoritmanın sistem modeli gösterilmiştir.



Şekil 9. Önerilen önbellek güncelleme algoritmanın modeli

Bu çalışmada kullanılan veri setini, hazırlanan web önbellek simülasyonlarında kullanılabilir hale getirmek için bu tezin genel bilgiler kısmında belirlenen veri madenciliği adımları birinci adımdan dördüncü adıma kadar veri ön işleme *(data preprocessing)* olarak veri seti üzerinde uygulanır.

2.2.2. Veri Önişleme

Trafik izleri kaydedilen laboratuvarlarda toplam 37 bilgisayar bulunmaktadır, bilgisayarlar yüksek lisans ve lisans laboratuvarı olarak iki sete ayırılmıştır. Yüksek lisans laboratuvarında 32 bilgisayar ve lisans laboratuvarında ise beş bilgisayar yerleştirilmiştir.

Bu tezde yapılan veri önişleme daha iyi anlaşılabilmesi için aşağıda iki genel adım olarak verilmiştir.

1- Bostun Üniversitesi tarafından sunulan veri seti her bilgisayarın trafik kayıtları için günlük olarak ayrı ayrı dosyalar oluşturulmuştur. Sunulan dosyalar bir metin dosyası olmasına rağmen dosya uzantısı mevcut değildir. Bu dosyalara ilk önce *Extension Changer*[2] programı ile *.txt* uzantısı eklendikten sonra *Ms-DOS* programı kullanılarak *copy *.txt all.txt* komutu ile tek bir metin dosyasına dönüştürülmüştür.

2- Tüm dosyaların birleştirilmesinden ortaya çıkan veri setinde, web önbelleklerin performansına etkisi olmayan, örneğin başarısız HTTP durum kodlarına sahip kayıtlar vb. birçok veri kayıtlar da bulunmaktadır. Bu tür kirli verileri, veri setinden kaldırmak ve veri setini daha temiz bir hâle dönüştürmek için sadece başarılı HTTP durum koduna sahip, 'cgi-bin' ve '?' alt dizileri içermeyen ve '.cgi' gibi dosya uzantısı olmayan URL'ler önbelleklenebilir (cacheable) URL olarak kabul edilmiştir [58]. İşlemi basitleştirmek ve işlemci yükünü azaltmak amacıyla her URL benzersiz rasgele bir rakam ile değiştirilmiştir.

Veri önişleme sürecin ikinci adımının gerçekleşmesi için R Programlama dili R-Stüdyo[3] programı ortamında kullanılmıştır. Bu çalışmada R-Stüdyo programın seçilmesi R-stüdyoda SQL komutlarının rahatça kullanılabilmesidir.

2.2.3. Puanlama İçin İstatistiksel Normalleştirmenin Kullanımı

İstatistiksel normalleştirme, veri madenciliği alanında verilerin ön işlemesinde özellikle veri normalleştirme aşamasında yaygın kullanılan bir yöntemdir. Bazen bir veri setinde yer alan veriler birbirlerinden çok farklı veya çok küçük/büyük olabilir. Bu yüzden istatistiksel normalleştirme, hesaplama kolaylığı için verileri belirli bir aralığa ayarlama işlemidir. İstatistiksel normalleştirme, farklı ölçekleme sisteminde mevcut olan verilerin birbiri ile karşılaştırabilmek için de kullanılır. Buradaki amaç, farklı sistemlerde yer alan

[2] Bu basit uygulama, bir klasördeki tek bir dosyanın uzantısını veya birden çok dosyayı kolayca değiştirmenizi sağlar. *Extension Changer,* dosya uzantılarını değiştirmek için iki yol sunmaktadır. Birinci yolu tüm dosyaları *drag and drop* yaparak program üzerine bırakıp yeni uzantıyı girdikten sonra değiştir tuşuna basmaktır. İkinci yolu ise dosyaların üstüne sağ tıklayıp uzantıyı değiştir seçeneği seçerek dosyaların uzantısını değiştirmektir.

[3] R Stüdyo, R topluluğunun aktif bir üyesidir. R Stüdyo, R programlama dilini daha kolay kullanılabilmesi için geliştirilmiş bir ortamdır. Önceden kurulmuş bir R programı olmadan Stüdyoyu işlevsel olarak kullanamazsınız.

verileri matematiksel fonksiyonlar arkalı, ortak bir sisteme taşıyarak karşılaştırılabilir hale dönüştürmektir [43]. Önerilen önbellek güncelleme algoritmada objelerin puanlamasında Asgari – Azami Normalleştirme (Min-Max Normalisation) fonksiyonu kullanılmıştır. Bu tür normalleştirmede, bir grup verilerin içerisindeki en küçük ve en büyük değerler ele alınır. Geride kalan diğer bütün veriler, bu değerlere göre normalleştirilir. Buradaki amaç en küçük değeri 0 ve en büyük değeri 1 olacak şekilde normalleştirmek ve diğer bütün verileri bu 0 - 1 aralığına yaymaktır. Her değerin normalleştirilmiş hâli aşağıda verilen formül 3'e göre hesaplanır [32].

$$X_{Normal} = \frac{X - X_{Asgari}}{X_{Azami} - X_{Asgari}} \tag{3}$$

Bu çalışmada değerleri 1-9 aralığında normalleştirmek için formül 3 aşağıdaki formül 4'te gösterildiği gibi manipüle edilmiştir.

$$Skor\_X_{rfd} = 8 \times \left(\frac{(X - X_{Asgari})}{(X_{Azami} - X_{Asgari})}\right) + 1 \tag{4}$$

Formül 3 ve 4'te, X puanlanma işlemi üzerinde gerçekleşecek olan obje, X_{Normal}, objenin normalleştirilmiş değeri, $X_{Asgari,}$ verti tabandaki objelerin asgari değeri, $X_{Azami,}$ verit abandaki objelerin azami değeri ve $Skor\_X_{fd}$, ise X objenin frekans ve gecikme metriklerinin puanlanmış hâlini gösterir.

Formül 4 sadece frekans ve gecikme metriklerin puanlanmasına kullanılır. Hâlbuki algoritma dört metrik üzerinde puanlama yaparak çalışmaktadır. Algoritmada kullanılan yenilik ve boyut metrikleri başka metriklerin tersine puanlanması gerekmektedir. Objelerin yenilik ve boyut metriklerinin puanlanması için formül 5 kullanılmıştır.

$$Skor\_X_{rs} = 8 \times \left(\frac{(X_{max} - X)}{(X_{max} - X_{min})}\right) + 1 \tag{5}$$

Objelerin yenilik ve boyut değerleri (X_{rs}) Formül 5 ile ters olarak hesaplanır. Yani en büyük değer en küçük puanı alır ve en küçük değer ise en büyük puan ile puanlanır. Başka

bir deyişle, en büyük boyutu olan objelerin önbellekten silinmesine öncelik verilmiştir bundan dolayı en büyük boyuta sahip olan objelere bir ve en küçük boyutu olan objelere dokuz puanı verilir. En büyük ve en küçük boyut değerleri arasında olan değerler, bir ve dokuz arasında olan puanları alırlar. Bu mantık geleneksel olarak kullanılan *size* web önbellek güncelleme algoritmasına dayalıdır. Size algoritması önbellek hafızası dolduğunda en büyük boyutta olan objeleri önbellek hafızasından kaldırır ve en küçük boyuta sahip nesneleri hafızada saklar.

Puanlama mekanizmasını daha kolay anlaşılabilmesi için tablo 1 örnek veri tabanındaki verilerin normalleştirilmiş hâlini gösterir. Normalleşme işlemi gerçekleşmeden önce, yenilik, objenin en son kaç milli saniye önce istenilmesini, frekans, objenin istenilen sıklığını, boyut, objenin kaç bayt olduğunu ve gecikme ise objenin indirilmesi kaç mili saniye de gerçekleşmiş olduğunu gösteren değerlerdir. Bu verilere göre her metrik, 1-9 aralığına değer alır.

Tablo 1. Örnek verilerin 1-9 aralığına normalleştirilmesi

Obje URL	**Yenilik**	**Frekans**	**Boyut**	**Gecikme**
www.a.com/a.avi	8	6	9	4
www.b.com/b.jpg	7	8	9	3
www.c.com/c.pdf	9	2	8	6
www.d.com/c.html	7	5	8	3
www.e.com/e.html	6	6	7	5
www.f.com/f.jpg	2	3	9	1
www.g.com/g.txt	7	7	1	9

2.2.4. Algoritmanın Eğitmesine K-Ortalama Algoritmanın Kullanımı

K-ortalama algoritması, *n* tane veriden oluşan veri setini kullanıcı tarafından verilen k harfi ile belirtilen k adet kümeye bölmeye çalışır. Küme merkezleri kümelerin birbirinden ayıran değerdir ve bu değer kümedeki nesnelerin ortalama değeri ile hesaplanır [42]. K-ortalama algoritması her küme için rasgele bir merkez belirleyerek verilerin merkezden mesafesine göre kümelere yerleştirir. Yapılan kümelemeye göre yeni merkezler belirler ve kümeleme işlemi kararlı hala (stable state) gelene kadar bu işlem devam eder. Merkez

belirleme işlemi için birçok uzaklık hesaplama yöntemi kullanmaktadır. Öklid uzaklık hesaplama yöntemi en yaygın olarak kullanılan ve birçok çalışmada en iyi sonuç veren yöntemlerden biridir [59]. Bu çalışmada da Öklid uzaklığı kullanılmıştır. Aşağıda denklem 6 olarak verilen denklem Öklid uzaklığın hesaplaması için kullanılan formülü gösterir.

$$d(m,n) = \sqrt{|m_1 - n_1|^2 + |m_2 - m_2|^2 + \ldots + |m_i - n_i|^2} \quad (6)$$

Denklem 6'da, *m* ve *n* veri setindeki veriler ve *d* ise verilerin arasındaki mesafeyi gösterir.

Önerilen web önbellek güncelleme algoritmasının öğreneme (training) aşamasında belli bir zaman aralığında olan önceki ağ trafiği üzerinde k-ortalama algoritması çalıştırılır. K-ortalama kümeleme algoritmasının küme sayısı bir olarak girildiğinde tablo 1'de verilen metriklerin ortalama değerleri tablo 2'de gösterildiği gibi hesaplanır.

Tablo 2. K-ortalama algoritmasıyla metriklerin ortalama değerlerinin hesaplanması

Metrikler	**Hesaplanan ortalamalar**
Yenilik	7.41
Frekans	5.42
Boyut	8.21
Gecikme	4.23

2.2.5. Objelere General-Puan Üretme

Hesaplanan ortalama değerlere göre algoritma metriklere 1, 10, 100 ve 1000 olarak ağırlık belirler ve aşağıdaki formül 7'ye göre önbellekteki her nesneye G_puan üretir. Formül 7, tablo 2'deki bilgilere göre yazılmıştır.

$$G\_puan = (10^3 x Size) + (10^2 x Recency) + (10 x Frequency) + Delay \quad (7)$$

Örneği, bir objenin yenilik değeri 8, frekans değeri 7, boyut değeri 6 ve gecikme değeri 3 olduğunda, yukarıdaki tabloya göre objenin puanı 6873 olacaktır.

Tablo 3. Objelere G_puan (General Puan) üretilmesi

Obje URL	Yenilik(10^2)	Frekans(10)	Boyut (10^3)	Gecikme	G_Puan
www.a.com/a.html	8	6	9	4	9864
www.b.com/b.jpg	7	8	9	3	9783
www.c.com/c.txt	9	2	8	6	8936
www.d.com/c.html	7	5	8	3	8753
www.e.com/e.avi	6	6	7	5	7665
www.f.com/f.jpg	2	3	9	1	9231
www.g.com/g.html	7	7	1	9	1779

Tablo 3'teki veriler web önbellek hafızasındaki objeler olarak düşünüldüğünde, yeni bir objenin yerleştirilmesi gerektiğinde en küçük G_puan'a (1779) sahip obje hafızadan kaldırılır ve yeni obje yerine yerleştirilir. Eğer yeni objenin boyutu, silinmiş olan objenin boyutundan büyükse ikinci küçük puanı olan obje de hafızadan kaldırılır.

2.3. Önbellek Güncelleme Algoritmanın Performans Metrikleri

Literatürdeki çalışılmalarda önbellek güncelleme algoritmaların performansını ölçmesi için genel olarak HR ve BHR parametreleri dikkate alınmıştır. Hâlbuki web önbellekler gecikmeyi önemli bir ölçüde azaltırken, azaltılan DR oranı da algoritmaların performans ölçmesinde ve karşılaştırmasında önemli bir parametre olarak birçok bilimsel yazılarda ismi geçmiştir.

Eğer önbellekten tatmin edilen objelerin isabet oranı h_r, önbellekten tatmin edilemeyen objelerin isabet oranı m_r, önbellekten tatmin edilen bayt isabet oranı h_b, önbellekten tatmin edilemeyen bayt isabet oranı m_b, toplam azalan gecekime d_i, ve toplam obje sayısı t olarak varsayılırsa, çalışmada önerilen algoritmanın performans ölçümleri, aşağıdaki 8, 9, 10 formüllerine göre yapılır.

2.3.1. İsabet Oranı

Önbellekten sağlanan istek sayısı, kullanıcılar tarafından istenilen objelerin toplam sayısına bölünerek elde edilir. İsabet oranı hesaplama formülü aşağıda formül 8 olarak verilmiştir.

$$HR = \frac{\sum h_r}{\sum h_r + \sum m_r} \times 100 \quad (8)$$

2.3.2. Bayt İsabet Oranı

Önbellek tarafından sunulan baytların toplamı, kullanıcıların talep ettiği toplam bayt sayısına bölünmesi sonucunda elde edilen orandır. Formül 9 BHR oranını hesaplamak için kullanılmıştır.

$$BHR = \frac{\sum h_b}{\sum h_b + \sum m_b} \times 100 \quad (9)$$

2.3.3. Gecikme Oranı

Bu makalede, gecikme, kullanıcı tarafından istenilen nesnelerin internetten indirilmesi için sarf edilen zaman olarak tanınmaktadır. Nesne web önbellekten tatmin edildiğinde gecikmesi 0 olarak farz edilmiştir ve DR oranı ise web önbellek tarafından azaltılan zaman ve toplam isteklerin internetten tatmin edilmesi için lazım olan zamana göre hesaplanır. DR oranı aşağıdaki formüle 10 ile hesaplanmaktadır.

$$DR = \frac{\sum d_i}{\sum d_t} \times 100 \quad (10)$$

2.4. Geniş Alan Ağlarda Web Önbellek Sunucuların En Uygun Yerleştirilmesi

Web önbellekler uzun zamandan günümüze kadar ağların yükünü azaltmak ve kullanıcıların erişim hızını daha iyileştirmek için uygun bir çözüm olarak kullanılmaktadır.

Ancak bir İSS veya geniş alan ağın tüm kenar *(edge)* noktalarında web önbellek yerleştirilmesi çok maliyetli bir çözüm olmasının yanı sıra optimum performansı göstermemektedir. Önbelleklerin en uygun noktalarda yerleştirmesi hem ağ performansını ve hem de ağ maliyetini verimli etkiler.

Bu makalenin ikinci çalışması olarak geniş alan ağlarda web önbelleklerin kurulması için en uygun yerleri tespit etmesine, ağların HTTP trafik hacmi, bant genişliği ve bağlantı yükü dikkate alınarak bir yöntem önerilmiştir.

Web önbellek yerleştirme yöntemi için dikkate alınan bilgiler tablo 4'te verilmiştir. Bu ağ bilgileri ağ izleme programları (Örneğin: PRTG *(Paessler Router Traffic Grapher)* Network Monitor [60]) ile ağ yöneticisi tarafından sunulabilen bilgilerdir. Tabloda, POP ID, birbirine bağlanan ve müşterilere hizmet veren POP'ların ID'si, HTTP trafik %, her POP'ın toplam trafiğinin üzerinden HTTP trafik yüzdesi, bağlantı yükü, her POP'ın ağ geçidi olan bağlantının ortalama yükü ve bant genişliği ise her POP'ın ağ geçidi bağlantısının bant genişliğini gösterir. Bu çalışmada bağlantıların bant genişliği her POP için 1000 Kb *(Kilo Bit)* olarak dikkate alınmıştır. Bağlantı yükü rasgele üretilmiştir ve HTTP trafik % ise kullanılan veri setindeki verilere göre yazılmıştır. Bu veriler ağların bir günlük veya bir aylık trafiği üzerinden çıkartılabilir. Tablo 4'te verilen bilgiler bir aylık trafik üzerinden yazılmıştır.

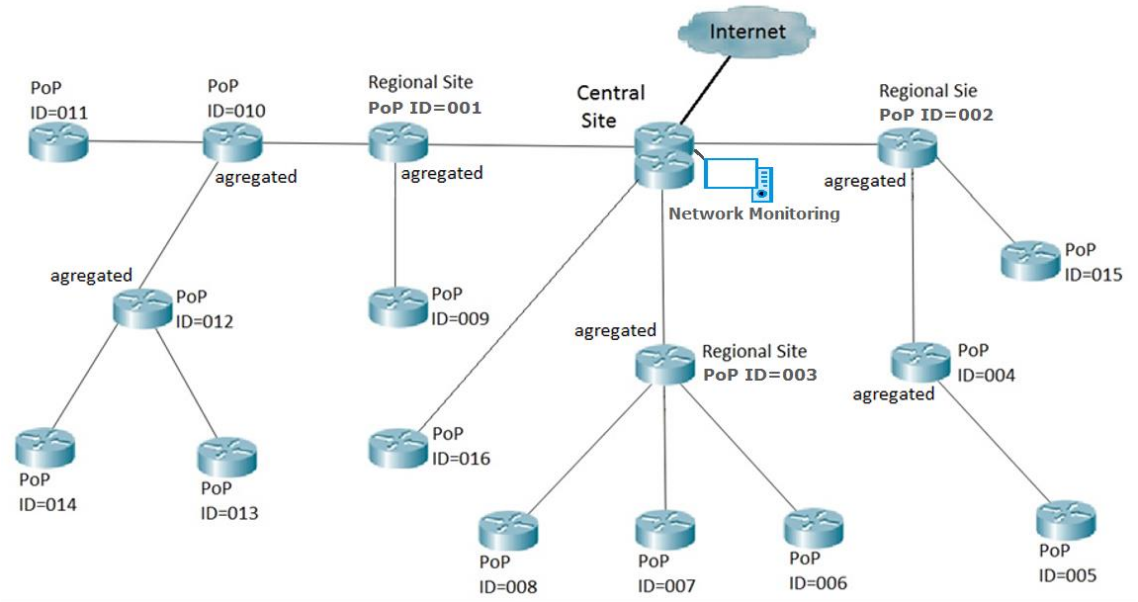
Tablo 4. Geniş alan ağ/İSS'nin bilgileri

POP ID	HTTP Trafik %	Bağlantı Yükü	Bant Genişliği Kb
_001	44	52	4000
_002	57	51	3000
_003	49	72	3000
_004	70	26	2000
_005	70	51	1000
_006	10	47	1000
_007	15	75	1000
_008	60	73	1000
_009	10	48	1000
_010	50	55	3000
_011	50	57	1000
_012	35	80	2000
_013	30	50	1000
_014	20	33	1000
_015	30	54	1000
_016	40	38	1000

Tablo 4'teki bilgiler kullanılarak, EIGRP yönlendirme algoritmasının bağlantı değeri (Link Cost) hesaplamasında kullanılan formülün mantığından faydalanarak her POP için bir bağlantı değeri üretilir. Bağlantı değeri hesaplama işlemi denklem 13'te verilen formül ile yapılır. Ve denklem 14 ile her POP için bir öncelik değeri hesaplanır.

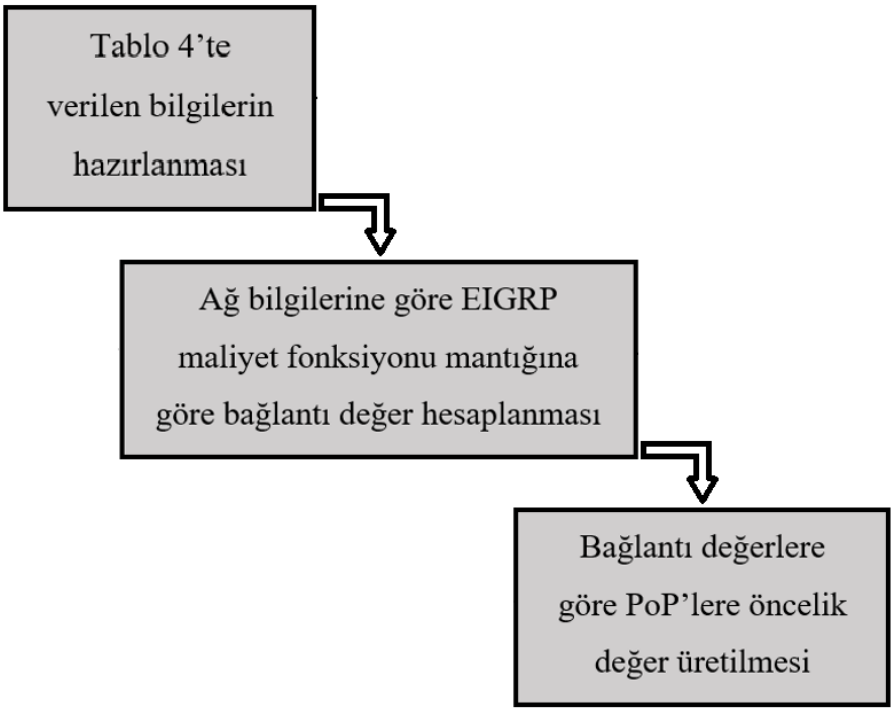
2.4.1. Kullanılan Topoloji ve Sistem Modeli

Web önbellek sunucularının en uygun yerleştirilmesi için önerilen metodun üzerinde uygulandığı geniş alan ağ/İSS topolojisi şekil 10'da gösterilmiştir. Gösterilen topolojide web önbellek sunucusu her POP üzerinde kurulması mümkündür. Ancak tüm POP'larda web önbellek sunucusunun kurulması çok masraflı bir çözümdür. Önerilen yöntemin önbellek sunucuların sayısını ve kurulması için en uygun POP'ları teşhis etmesine gerekken bilgiler, ağ izleme programları (Örnegin: PRTG Network Monitor [60]) ile elde edilir.



Şekil 10. Çalışmada kullanılan ağ topolojisi

Ağ izleme programları, İSS/geniş alan ağların ağ operasyon merkezleri NOC *(Network Operation Center)*'de bir bilgisayar üzerinde kurularak tüm ağda izleme işlemini yapar. Web önbellek sunucularının optimal yerleştirilmesi için önerilen metodunun şekil 11'de gösterilen sistem modeline göre web önbellek sunucuların en uygun yerleştirilmesi üç aşamada yapılır. İlk önce ağ yöneticisi tarafından metodun çalışmasına gerekken tüm bilgilerin ağ izleme programlarıyla hazırlanması gerekmektedir. İkinci aşamada, hazırlanan bilgiler kullanılarak EIGRP yönlendirme algoritmasında her bağlantının maliyeti hesaplandığı gibi topolojide gösterilen tüm bağlantılar için bağlantı değeri hesaplanır. Üçüncü aşamada ise birinci aşamada hazırlanan bilgiler ve ikinci aşamada üretilen bağlantı değerleri kullanılarak her POP için öncelik değeri hesaplanır. Hesaplanan öncelik değerlere göre ağ yöneticisi hangi POP'ta önbellek sunucusunun yerleştirmesi daha önemli olduğunu basitçe anlar ve sunucuların en uygun noktalarda yerleşimini yapar.



Şekil 11. Önbellek sunucu yerleştirme metodunun sistem modeli

2.4.2. Bağlantı Değerin Hesaplanması

EIGRP yönlendirme algoritması en iyi yolu teşhis etmesinde veya bağlantı değerinin hesaplamasında aşağıda verilen metrik parametrelerini kullanır [46]:

Bandwidth (K1)	Rotadaki minimum bant genişliği (Lowest bandwidth of route)
Load (K2)	Paket oranına göre rotadaki en büyük yük (Worst load on route, based on packet rate)
Delay (K3)	Hedef ağ rotasındaki toplam gecikme (Cumulative interface delay of route)
Reliability (K4)	keep alive veriye göre rotanın en kötü güvenilirliği (Worst reliability of route based on keep alive)
MTU (K5)	Rotadaki en küçük MTU (Smallest MTU in path)

Varsayılan olarak EIGRP'de hedef ağa giden rotadaki bağlantıların minimum bant genişliği (K1) ve toplam gecikmesi (K3) kullanılır. Gerçi ağ yöneticisi olarak EIGRP'yi konfigürasyon yaparken başka metrikleri de kullanabilirsiniz ancak ağda döngüler oluşturabileceğinden dolayı önerilmemiştir [47]. Bant genişliği ve gecikme metrikleri, hedef ağa giden rotada yönlendiricilerin interface'nde yapılandırılan değerlerden tespit edilir.

EIGRP yönlendirme algoritmasında en iyi yolun teşhis edilmesi, bağlantıların metrik değerine bağlıdır. Metrik hesabı birden fazla verilere dayanmaktadır. Aşağıdaki formül 11 EIGRP'de tek bir 32 bit metrik değeri üretmek için kullanılır.

$$Total\ Metrik = \left[\left(K_1.Bandwidth_E + \frac{K_2.Bandwidth_E}{256 - Load} + K_3.Delay_E\right).\frac{K_5}{K_4 + Reliability}\right].256 \quad (11)$$

Toplam metrik formülünün en önemli kısmı olan K değerleri, metrik hesaplamada hangi verilerin kullanılıp kullanılmamasına karar verir. K sadece 0 veya 1 değerleri alabilir. 0 değeri aldığında ilişkin metrik toplam metrik hesaplamada katkı sağlamaz.

EIGRP'de var sayılan K değerleri aşağıda verilmiştir.

K1 = 1

K2 = 0

K3 = 1

K4 = 0

K5 = 0

Var sayılan k değerler formüle 4'te kullanıldığında aşağıdaki formül 12 elde edilir.

$$Genel\ Metrik = (bant\ genişliği + gecikme) \times 256 \quad (12)$$

Bu çalışmada EIGRP metrik hesaplama mantığına dayanarak POP'ların bir üst POP'a veya internete bağlanan bağlantılarının toplam metriğini hesaplamak için, K1 değeri (bağlantının bant genişliği) ve K2 (bağlantının ortalama yükü) kullanılmıştır. Bağlantı değeri aşağıdaki formül 13 ile hesaplanmıştır.

$$Bağlantı\ değeri = \frac{bant\ genişliği}{ortalama\ yük} \quad (13)$$

2.4.3. Öncelik Değerin Hesaplanması

Tablo 5'te gösterildiği gibi Formül 13 ile her bağlantı için bir bağlantı değeri hesaplanır. Hesaplanan bağlantı değerler kullanılarak formül 14 ile web önbellek sunucuların yerleştirilmesi mümkün olan noktalar için öncelik değeri üretilir.

$$\text{Öncelik değeri} = \frac{\sum http\ trafik}{\text{Bağlantı Değeri}} \tag{14}$$

Tablo 5. Hesaplanan bağlantı değerleri ve üretilen öncelik değerleri

POP ID	HTTP Trafik %	Bağlantı Yükü	Bant Genişliği	Bağlantı Değeri	Öncelik Değeri
_008	60	73	1000	13.7	4.4
_005	70	51	1000	19.6	3.6
_011	50	57	1000	17.5	2.9
_015	30	54	1000	18.5	1.6
_013	30	50	1000	20.0	1.5
_016	40	38	1000	26.3	1.5
_012	35	80	2000	25.0	1.4
_003	49	72	3000	41.7	1.2
_007	15	75	1000	13.3	1.1
_002	57	51	3000	58.8	1.0
_004	70	26	2000	76.9	0.9
_010	50	55	3000	54.5	0.9
_014	20	33	1000	30.3	0.7
_001	44	52	4000	76.9	0.6
_006	10	47	1000	21.3	0.5
_009	10	48	1000	20.8	0.5

Üretilen öncelik değerlerden en büyük öncelik rakama sahip POP'lar web önbelleğin kurulması için önceliğe sahip olan POP'lardır.

2.5. Önbellek Yerleştirme Metodun Performans Metrikleri

2.5.1. Tasarruf Edilen Toplam Bant Genişliği

Bu çalışmada önerilen, web önbellek yerleştirme metodun mümkün olduğu kadar bant genişliğini tasarruf edebileceğini ispatlamak için sıralanan web önbelleklerin tasarruf ettiği bant genişliği oranı aşağıdaki formül 15 ve 16 ile hesaplanmıştır.

$$tasarruf\ link_i = \left(\left(\frac{100}{toplam\ bayt\ PoP_i} \right) \times \left(tasarruf\ edilen\ bayt\ PoP_i \right) \right) \quad (15)$$

$$Toplam\ Bantgenislik\ Tasarruf = \sum_{i=1}^{n} link_i \quad (16)$$

Formül 15'te *tasarruf link*$_i$ önbelleğin *i* POP'ın bağlantısına göre bant genişlik tasarrufu, *tasarruf edilen bayt POP*$_i$, önbellek kurulan POP'ta önbellekten temin edilen baytlar boyutu ve *toplam tasarruf* ise tüm ağa göre önbelleğin tasarruf ettiği bant genişliğini gösterir.

2.5.2. Maliyet Grafiği

Ağlarda web önbellekleri optimum sayıda yerleştirilmesi için tasarruf edilen bant genişliği maliyet grafiği ve önbellek maliyet grafiği kullanılmıştır. Tasarruf edilen bant genişliğinin maliyeti aylık 1Mb bant genişliği için ödenecek maliyet üzerinden ve her önbelleğin toplam fiyatı üzerinden aylık maliyeti hesaplanır. Sıralanan önbelleklerin tasarruf edilen bant genişliği grafiği ve her önbelleğin aylık maliyet grafiği çizildiğinde çizgilerin çatıştığı nokta X koordinat üzerinde yer alan web önbellek sayısının hangisine isabet ederse o rakam optimum önbellek sayısını gösterir. Her önbellek için aylık maliyeti ve tasarruf edilen bant genişlik maliyeti ağ yöneticisinin kendi hesaplamalarına göre yapılır.

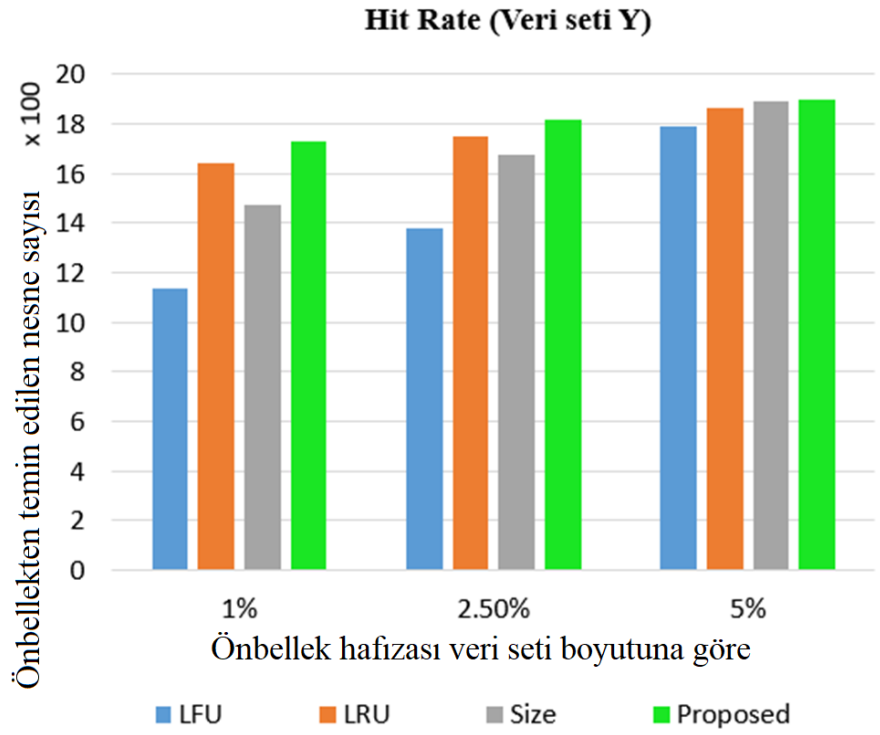
3. BULGULAR VE TARTIŞMALAR

3.1.Web Önbellek Güncelleme Algoritması

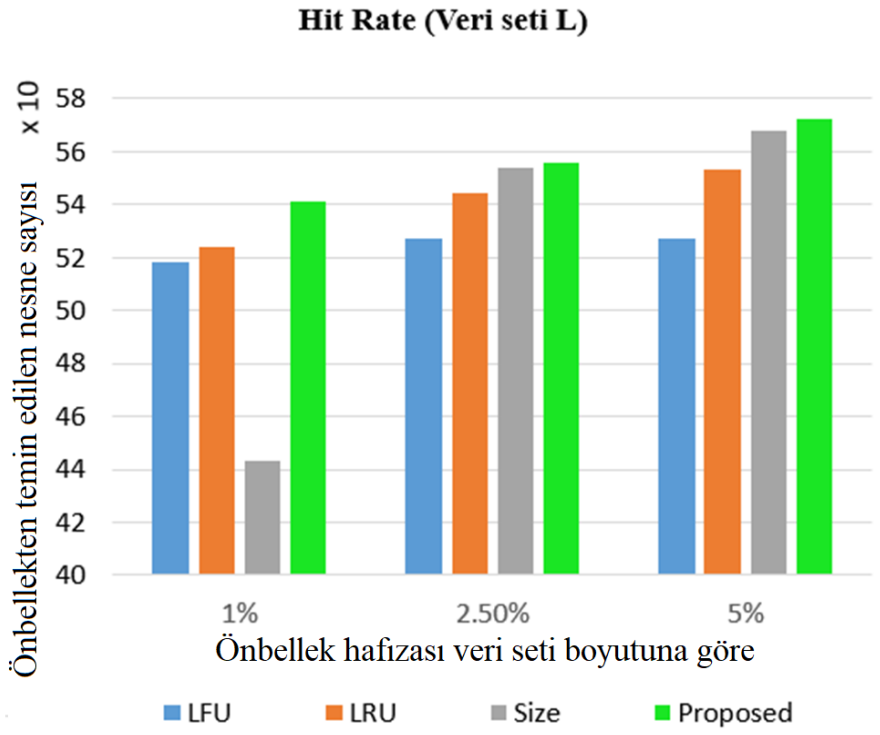
Önerilen web önbellek güncelleme algoritmasının performansını başka kategorilerin özgün temsilcisi olan ve geleneksel olarak kullanılmakta olan algoritmalarla karşılaştırmak için, önerilen algoritma ile birlikte geleneksel olarak kullanılan LRU, LFU ve Size algoritması da Matlab 2013 ortamında simüle edilmiştir. Algoritmaların performansını ölçmek için Bostun Üniversitesi tarafından sunulan veri seti kullanılmıştır. Algoritmayı bir günlük trafik izlerini kullanarak eğittikten sonra algoritmanın eğitilmesinde kullanılan veri setinin devamı olarak 24 saatlik trafik kayıtları algoritmaları çalıştırmak için simülasyona sokulmuştur. Önerilen algoritma her ağda en iyi performansı gösterir iddiasını ispatlamak için, algoritma, 2 laboratuvardan toplanan veri seti üzerinde ayrı ayrı çalıştırılmıştır.

İşlemi basitleştirmek ve işlemci yükünü azaltmak amacıyla her URL benzersiz rasgele bir rakam ile değiştirilmiştir. Bu işlemler gerçekleştikten sonra, algoritma performansına etkisi olan yararlı bilgileri içeren ortaya çıkan veri setini simülasyona sokarak algoritmaların performansları karşılaştırılmıştır.

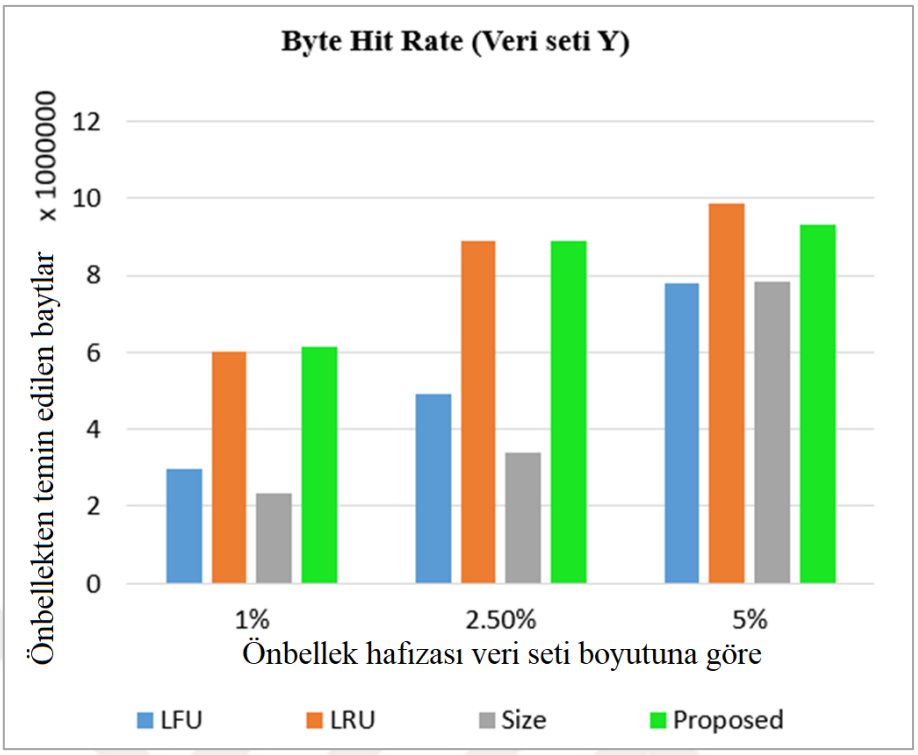
Literatürdeki çalışmaların simülasyon sonuçlarına göre, önbellek hafızasının boyutu, önbellek sunucusunun üzerinden geçen trafik hacmine göre çok büyük olduğu takdirde, tüm algoritmalar aynı performansı gösterir. Bu nedenle bu çalışmada yapılan simülasyonda web önbelleklerin hafızası, veri setinin toplam boyutunun %1, %2.5 ve %5'i olarak ayarlanmıştır. Şekil 12-a ve şekil 12-b algoritmaların farklı veri setlerinde sakladığı HR oranını göstermektedir.



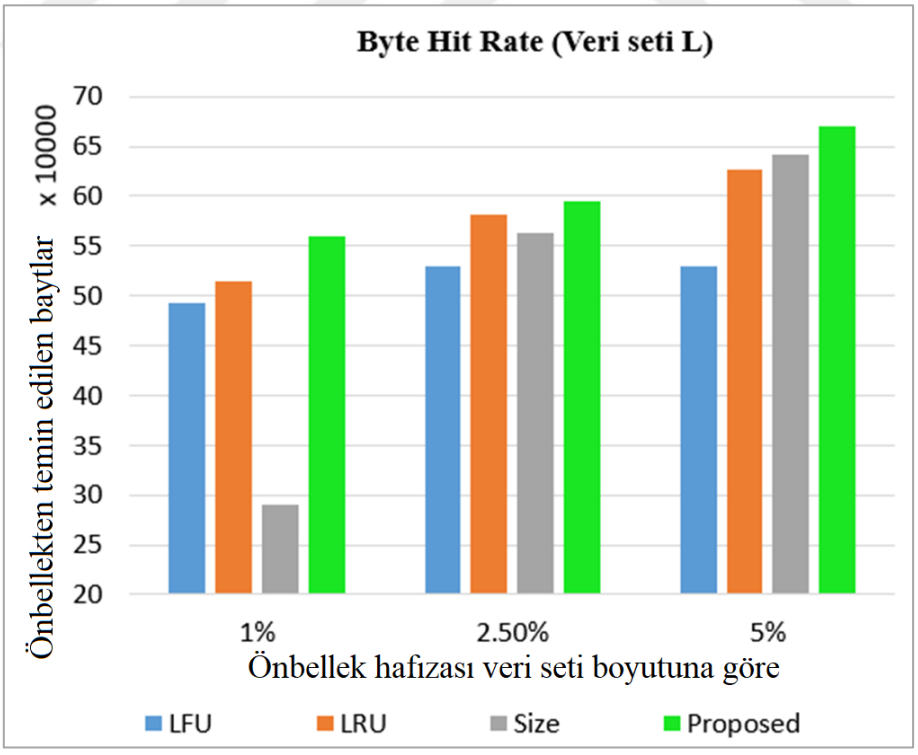
Şekil 12-a. Algoritmaların HR oranı Y veri setine göre



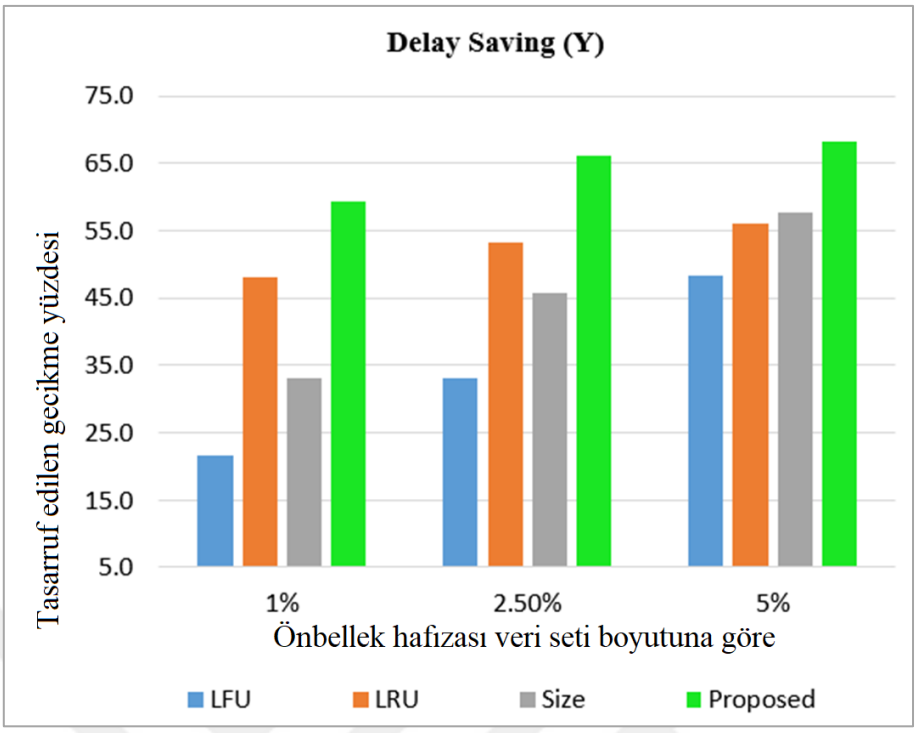
Şekil 12-b. Algoritmaların HR oranı L veri setine göre



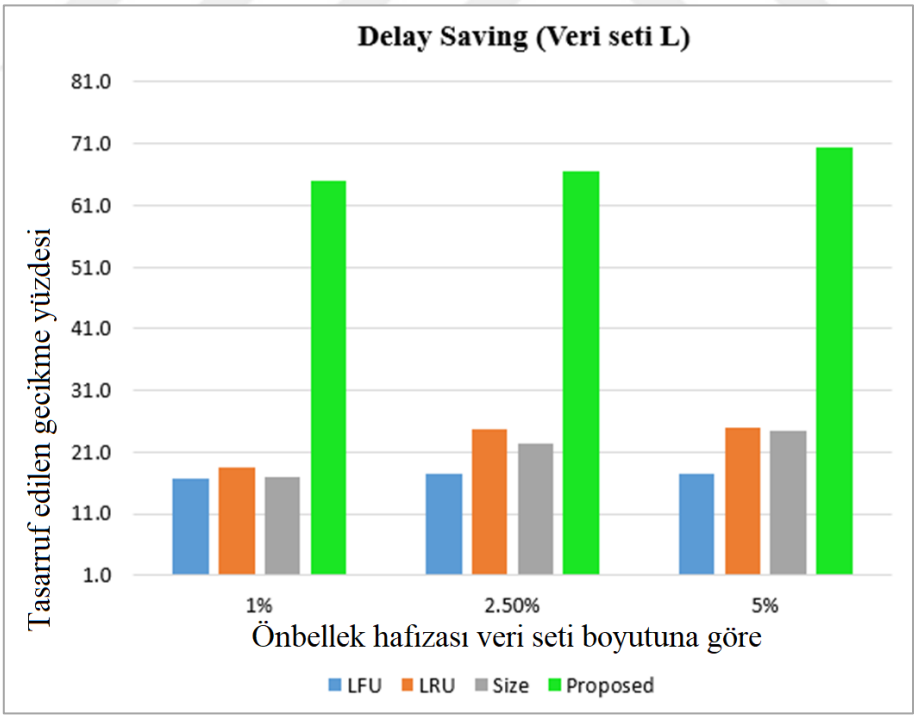
Şekil 13-a. Algoritmaların BHR oranı Y veri setine göre



Şekil 13-b. Algoritmaların BHR oranı L veri setine göre



Şekil 14-a. Algoritmaların DR oranı Y veri setine göre



Şekil 14-b. Algoritmaların DR oranı L veri setine göre

Simülasyon sonuçlarına göre önerilen algoritma, uygulandığı iki veri setinde en iyi performansı göstermektedir. Grafikte görüldüğü gibi önerilen algoritma küçük hafızaya

sahip önbelleklere daha iyi sonuç vermektedir. Şekil 13-a ve şekil 13-b'de önbellekten temin edilen bayt oranı gösterilmiştir. Önerilen algoritma en büyük BHR oranına sahipken LRU algoritma çok küçük fark ile ikinci sırada yer almaktadır. Şekil 14-a ve şekil 14-b algoritmaların DR oranı göstermektedir. Azalan DR oranları incelendiğinde, önerilen algoritma ile başka algoritmalar arasında çok büyük fark görülmektedir. Önerilen algoritmanın azaltan DR oranı bazı veri setlerinde başka algoritmaların hemen hemen iki katıdır.

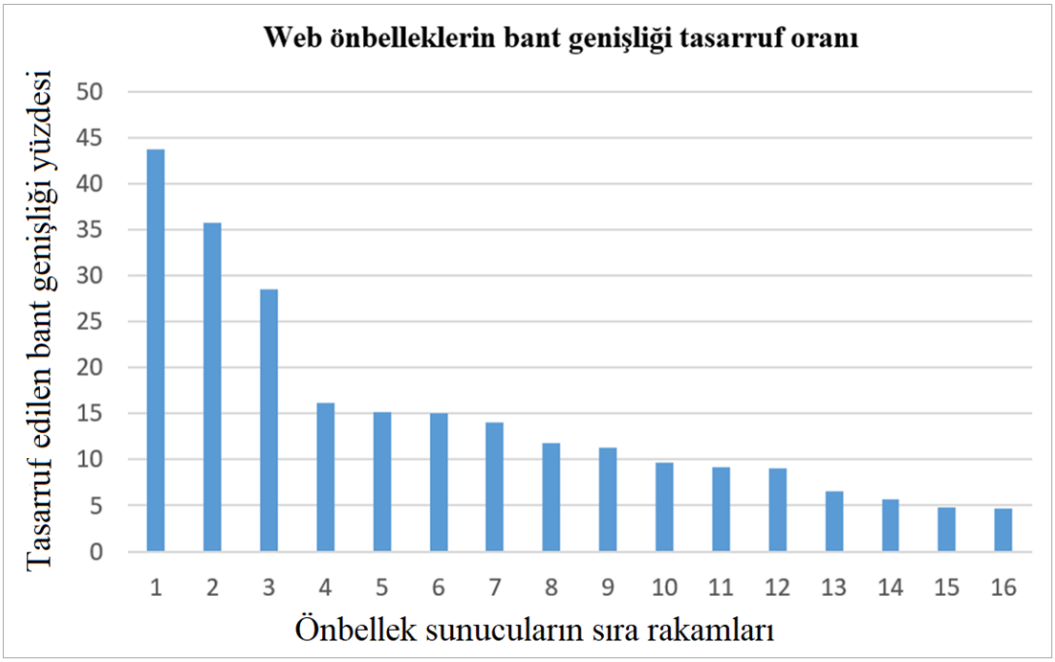
Yapılan simülasyon ile üretilen grafikler, önerilen algoritmanın tüm performans metriklerinin değeri başka algoritmalara göre daha iyi olduğunu çok net bir şekilde göstermektedir. LRU algoritması, BHR ve DR oranı metriklerine göre ikinci sırada yer almaktadır.

3.2. Web Önbellek Sunucuların En Uygun Yerleştirilmesi

En uygun web önbellek yerleştirilme çalışmasının simülasyonunda gerekli veri setini hazırlamak için Boston Üniversitesi tarafından sunulan veri setinin yüksek lisans laboratuvarında yer alan bilgisayarların trafik kayıtları kullanılmıştır. Boston Üniversitesinin yüksek lisans laboratuvarında toplam 32 bilgisayar bulunmaktadır. Bu çalışmada 32 bilgisayar isimlerine göre sekil 3'te gösterilen on kenarda yer alan her POP'ta iki, üç ve dört bilgisayar bağlanmış şekilde rasgele dağıtılmıştır. Önerilen web önbellek sunucu yerleştirme metodu şekil 10'da verilen topoloji üzerinde iki şekilde uygulanmıştır. 1- toplam POP'lara göre (kenar POP'lar ve Bölgesel siteler) web önbellek sunucuların kurulacak yerlerinin sıralanması. 2- sadece kenar POP'lara göre (Bölgesel siteler hariç) web önbelleklerin kurulabilecek mümkün yerlerin sıralanması.

3.2.1. Ağdaki Tüm POP'lara Göre Web Önbellek Sunucu Yerleştirme

Önerilen metot ile sıralanan noktalarda web önbellek simülasyonunu çalıştırarak her önbelleğin ağa katan bant genişliği tasarrufu formül 15 ve 16 ile hesaplandığında aşağıda gösterilen şekil 15-a'dakı grafik ortaya çıkmıştır.



Şekil 15-a. Sıralanan web önbelleklerin tasarruf eden bant genişliği oranı

Şekil 15-a'ya göre birinci sırada yer alan web önbellek kendisinden sonraki sıralarda yer alan tüm önbelleklerden daha yüksek bant genişliği tasarruf oranına sahiptir. Ve aynı şekilde ikinci, üçüncü, dördüncü… sıralarda yer alan önbelleklerin her hangibiri kendilerinden sonradaki sıralarda yer alan web önbelleklerden daha yüksek bant genişliği tasarruf oranına sahiptirler.

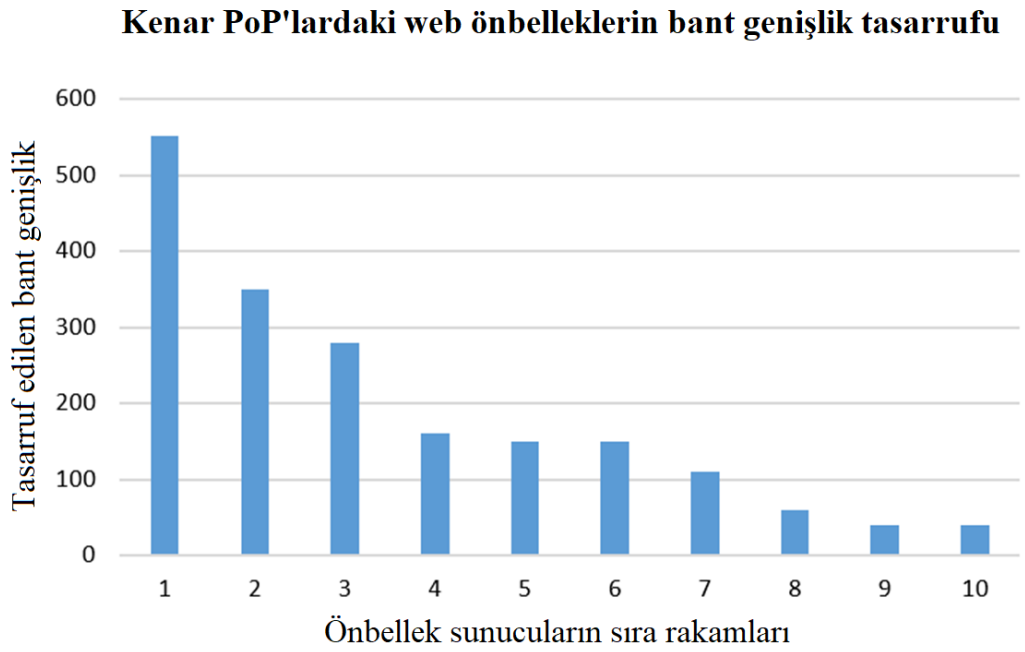
3.2.2. Kenar POP'lara Göre Web Önbellek Sunucu Yerleştirme

Önerilen metot sadece kenar POP'lar üzerinde de uygulandığında en çok bant genişliği tasarruf edilebilecek POP'lar, web önbellek sunucusu kurulması için ileri sıralarda yer alır. Bu çalışmada kullanılan topolojide gösterilen kenar POP'ların bilgileri üzerinde sıralama yapıldığında tablo 6'da kenar POP'ların bilgileriyle beraber verilmiş sıra rakamları ortaya çıkmaktadır.

Tablo 6. Topolojideki kenar POP'ların bilgileri

POP ID	HTTP Trafik %	Bağlantı Yükü	Bant Genişliği	Bağlantı Değeri	Öncelik Değeri
_008	60	73	1000	13.7	4.4
_005	70	51	1000	19.6	3.6
_011	50	57	1000	17.5	2.9
_015	30	54	1000	18.5	1.6
_013	30	50	1000	20.0	1.5
_016	40	38	1000	26.3	1.5
_007	15	75	1000	13.3	1.1
_014	20	33	1000	30.3	0.7
_006	10	47	1000	21.3	0.5
_009	10	48	1000	20.8	0.5

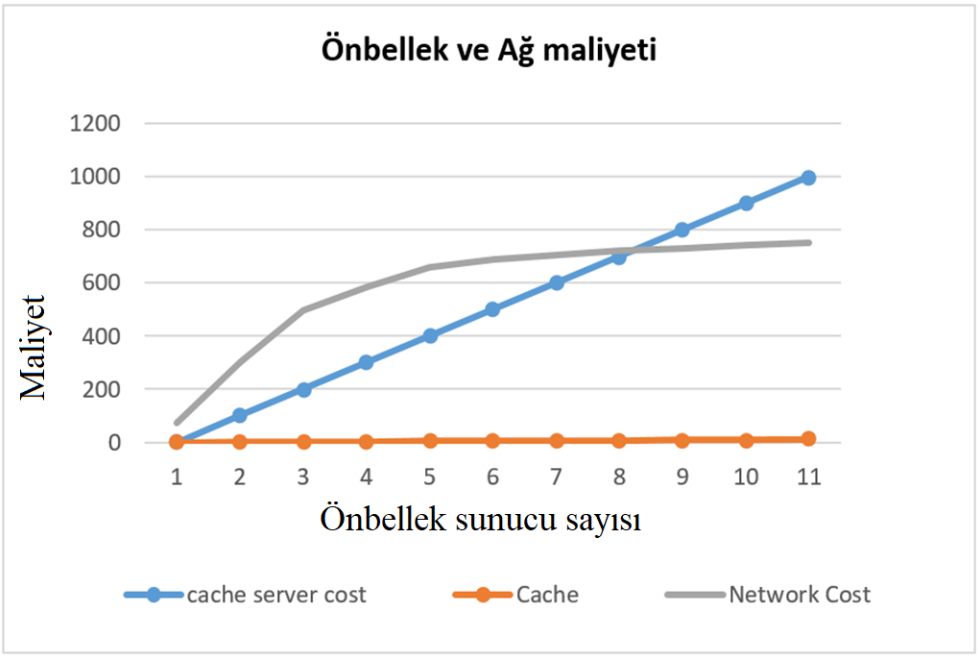
Tablo 6'deki sıralamaya göre önbelleklerin yerleştirilmesi yapıldığında, web önbelleklerin tasarruf ettiği bant genişliğinin oranları şekil 15-b'de gösterilmektedir.



Şekil 15-b. Sıralanan web önbelleklerin tasarruf eden bant genişliği oranı

3.3. Optimum Sayıda Web Önbellek Sunucu Yerleştirme

Bir ağda web önbelleklerin optimum sayısı, tasarruf edilen bant genişliğinin maliyeti ve web önbelleklerin maliyetinin grafiği ile belirlenebilir. Şekil 16'deki grafiğe göre bu çalışmada kullanılan topolojide sekiz adet sunucunun öncelik değerlerine göre yerleştirilmesi optimum olarak önerilmiştir. Şekil 15-a, bu çalışmada önerilen metot ile belirlenen noktaların web önbellek sunucusunun yerleştirilmesi için en uygun noktalar olduğunu çok net bir şekilde göstermektedir.



Şekil 16. Önbellek ve ağ maliyetine göre optimum önbellek sayısı

3.4. Web Önbellek Yerleştirme Metodu İle Yedek Bağlantılar İçin Lazım Olan Bant Genişliğin Belirlenmesi

Web önbellek yerleştirme metodu ile sıralanan önbellek sunucuların tasarruf ettiği bant genişlik oranlarından ağ/İSS yöneticisi her POP için yaklaşık ne kadar bant genişliği olan yedek bağlantı lazım olduğunu tahmin edebilir. Yani web önbellek yerleştirilen POP'lar için tasarruf edilen bant genişliği ve ana bağlantının yüküne göre ve web önbellek yerleştirilmeyen POP'lar için sadece ana bağlantının yüküne göre lazım olan bağlantı yükü tahmin edilebilir.

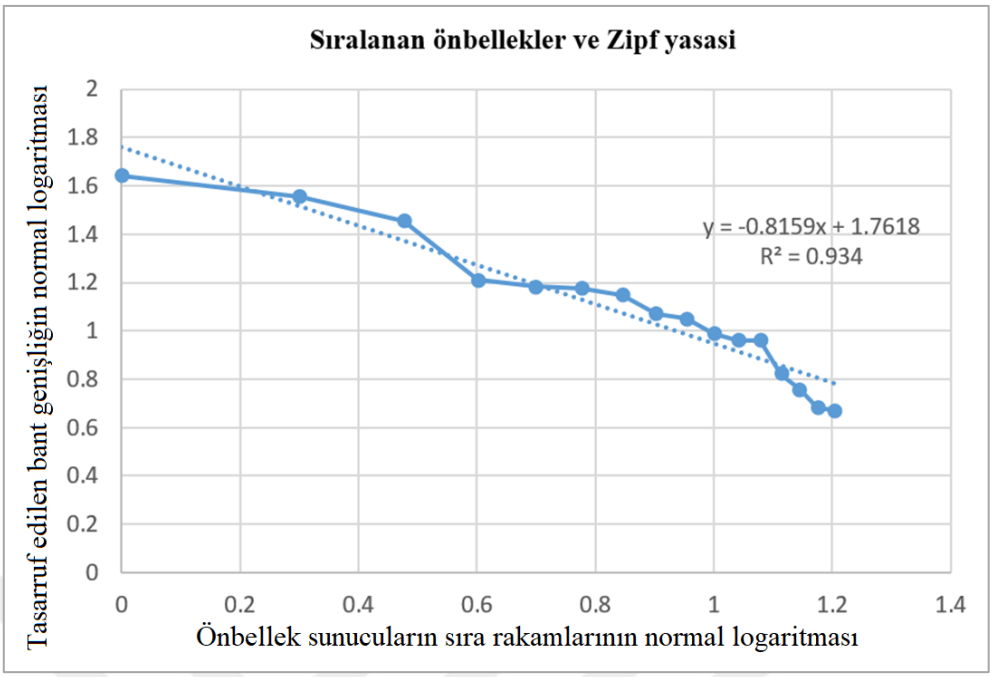
3.5. Sıralanan Önbelleklerin Zipf Yasasına Uygunluğu

Dilbilimci George Kingsley Zipf tarafından 1949 yılında önerilen Zipf yasasına göre, belli bir dilde yazılan bir eserde az sayıda birtakım sözcük her zaman kullanılıyorken sözcüklerin büyük bir çoğunluğu düşük frekansla kullanılıyor. Sözcükler kullanım frekansına göre sıralandığında çarpıcı bir örüntü ortaya çıkıyor. Birinci sırada yer alan sözcük, ikinci sıradaki sözcüğün hep iki katı kadar sıklıkta, üçüncü sıradaki sözcüğün de hep üç katı kadar sıklıkta kullanılıyor olması fark ediliyor. Zipf 1949 [48]'dan bu yana, frekans ve sıra rakamlarının normal logaritmasına göre grafik oluşturarak verilerin zipf yasasına uyup uymaması tespit edilmektedir. Oluşturulan grafiğin eğimi her ne kadar eksi bir rakamına yakınsa verilerin zipf yasasına uygunluğu yüksek demektir. Elbette bazı örnekler, denkleme bu kadar iyi uymaz, ama örnekler logaritmik grafiğe oturtulduğunda, şaşırtıcı derecede düz bir çizgi verir.

Zipf yasası şehirler üzerine uygulandığında, kentlerin nüfuslarını en büyüğünden en küçüğüne doğru sıralayıp X koordinatında kent büyüklüklerinin logaritması ve Y koordinatında ise bunlara karşılık gelen sıralamanın logaritması alındığında, eğimin -1'e eşit veya çok yakın bir rakam olduğu beklenmektedir [49] [61].

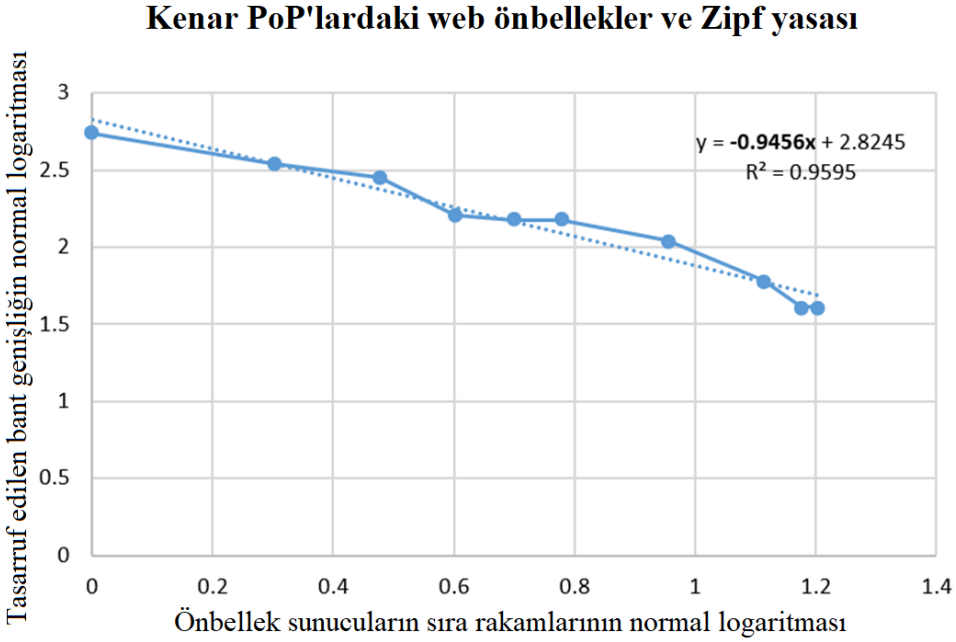
Zipf yasası, şirket büyüklükleri sıralamaları, gelir sıralamaları, web sayfalarının ziyaretçi sıralamaları [62] vb. birçok örnekler için de uygulanmıştır.

Önbelleklerin sıra rakamlarının normal logaritması ile tasarruf edilen bant genişliği miktarının normal logaritmasının grafiği çizildiğinde ortaya çıkan çizginin eğimi şekil 17-a'da gösterildiği gibi -0.8159 çıkmaktadır. Bu rakam -1 de çok yakın bir değer olduğu, web önbelleklerin Zipf yasasına uyduğu ortaya çıkmaktadır. Sıralanan önbelleklerin tasarruf eden bant genişliklerine göre Zipf yasasına uyduğu ilk kez bu çalışmada gösterilmiştir. Ancak bu çalışma simülasyon ortamında yapılmıştır. Gelecek çalışmalarda sıralanan önbellekler tasarruf edilen bant genişliğe göre Zipf yasasına uyduğu gerçek ağlar üzerinde yapılması planlamıştır.



Şekil 17-a. Sıra-Tasarruf edilen bant genişliğin Zipf yasasına uygunluğu

Web önbellek yerleştirmesi sadece kenar POP'lar üzerinde yapıldığında sıralanan web önbellek sunucularının tasarruf ettiği bant genişliği oranlarına göre Zipf yasasına uygunluğu şekil 17-b'de gösterilmektedir.



Şekil 17-b. Sıra-Tasarruf edilen bant genişliğin Zipf yasasına uygunluğu

Şekil 17-b'de gösterilen grafiğe göre kenar POP'larda yerleştirilen web önbelleklerin Zipf yasasına uygunluğu tüm ağda yer alan web önbelleklerden daha iyi çıkmaktadır. Şekil 17-a ve 17-b'ye göre aynı düzeyde yer alan POP'ların Zipf yasasına uyma ihtimali daha yüksektir.

4. SONUÇLAR

Web önbelleklerin tarihi çok eskilerde dayanmasına rağmen günümüze kadar çok uygun bir çözüm olarak internet dünyasında kullanılmaktadır. Son yıllarda ICN ve CDN teknolojileri yeni mimari olarak önerilmiştir ancak ICN hâlâ kâğıt üzerinde önerilen bir mimaridir. Günümüzde internette kullanılan donanımlar ve yazılımlar ICN mimarisi için uygun olmadığından bu yeni mimarinin uygulanması çok masraflı ve zaman alıcı bir süreç olarak öngörülüyor. CDN teknolojisi çok ilgi çekmesine rağmen telekomünikasyon operatörleri ve ağ yöneticileri, İSS tarafından işletilen CDN'lere, transparent ve geleneksel önbelleği ağlarına entegre ederek girmesine ilgi göstermektedirler. Bundan dolayı İSS içinde web önbelleklerin en uygun noktalarda yerleştirmesi ve önder önbellek güncelleme algoritmanın kullanılması ağ performansının arttırmasının yanı sıra ağ maliyetini verimli etkiler.

Literatürde önerilen birçok web önbellek güncelleme algoritmalarında, en fazla üç metrik kullanılmıştır ve genellikle gecikme metriği dikkate alınmamıştır. Hâlbuki web önbelleklerin çok önemli özelliklerinden birisi algılanan gecikmenin azaltmasıdır. Bu çalışmada önerilen dinamik web önbellek güncelleme algoritması her ağda en iyi performansı gösterebilmesi için ağın önceki trafik izlerini analiz ederek parametrelerini trafik özelliklerine göre ayarlar. Literatürde önerilen algoritmalar sadece HR ve BHR oranlarına göre belli ağlarda iyi sonuç verirken bu çalışmada önerilen dinamik web önbellek güncelleme algoritma farklı veri setleri üzerine uygulandığında, sadece HR ve BHR oranları değil DR oranı de günümüzde kullanılan algoritmaların değerlerinden daha yüksek çıkmaktadır.

İSS veya geniş alan ağlarda, web önbelleklerin en uygun noktalarda optimum sayıda yerleştirilmesi için önerilen metot, ağların HTTP trafik hacmi, bant genişliği ve bağlantı yükünü dikkate alır. EIGRP metrik hesaplamasında kullanılan formlulara dayanarak her bağlantı için bağlantı değeri hesaplar ve web önbellek kurulması için tüm mümkün noktalara öncelik değer üretir. Önbellek kurulmasında en yüksek öncelik değerine sahip POP'lara öncelik verilir. Simülasyon sonuçlarına göre, yüksek öncelik değere sahip POP'larda yerleştirilen önbellekler kendisinden küçük öncelik değere sahip POP'lardaki önbelleklerden daha çok bant genişliği tasarruf etmektedir.

Önbelleklerin sıra rakamlarının normal logaritması ile tasarruf edilen bant genişliği miktarının normal logaritmasının grafiği çizildiğinde ortaya çıkan çizginin eğimi Zipf yasasına uygun olduğunu göstermektedir.

Bu tezde yapılan birinci çalışmadan "Enhanced Web Cache Replacement Policy Based On Data Mining and RFSD scoring" konulu bir bildiri *"2nd World Conference on Technology, Innovation and Entrepreneurship"* uluslararası konferansında sözlü olarak sunulmuş ve *PressAcademia Procedia* web sayfasında yayınlanmıştır [63].

5. ÖNERİLER

Web önbellek güncelleme algoritmaların performansını arttırabilecek daha kullanılmayan bazı faktörler bulunmaktadır. Örneği objelerin konumu, eğer bir İSS'de IPV6 kullanılırsa nesnelerin en çok nereden istendiği IP adresi üzerinden çıkartılabilir. Böylece İSS'nin merkezi sitesinde yerleştirilen web önbellek POP'ların konumunda meşhur olan/olmakta olan nesneleri POP'ın önbelleğine olup olmadığını kontrol eder ve mevcut olmadığı takdirde POP'ın önbelleğine gönderir ve gelecek istekler için depolar. Böyle bir durumlarda merkezi sitede ve kanar POP'larda farklı web önbellek güncelleme algoritmaları kullanılması daha uygun olabilir.

Tüm fonksiyon tabanlı web önbellek güncelleme algoritmalar geleneksel algoritmalardan daha çok işlemci gücünü tüketmektedir. Bu çalışmada önerilen algoritma da bu kategoride yer aldığından bu özelliği taşımaktadır. Önerilen algoritmanın işlemci yük tüketimini azaltmak için puanlama ve ağ analizi başka bir bilgisayar ile yapılabilir. Böylece daha yakın zaman aralıkları ile ağ trafiğinin analizi yapılarak algoritmanın performansı da arttırılabilir.

6. KAYNAKLAR

1. Pahlavan, K. ve Krishnamurthy, P., Networking Fundamentals: Wide, Local and Personal Area Communications, John Wiley & Sons, Chippenham, 2009.

2. Regan, P., Wide Area Networks, Prentice Hall, New Jersey, 2004.

3. Stallings, W., Local & Metropolitan Area Networks, Prentice Hall, New Jersey, 2000.

4. Behrouz, A. F., Data Communications and Networking, McGraw-Hill, Singapore, 2007.

5. Cisco, Network Caching Technologies. http://docwiki.cisco.com/wiki/Network_Caching_Technologies 22 Ağustos 2017.

6. Rouse, M., ISP (Internet Service Provider). http://searchwindevelopment.techtarget.com/definition/ISP 20 Aralık 2017.

7. Abdalla, A., Sulaiman, S., ve Ali, W., Intelligent Web Objects Prediction Approach in Web Proxy Cache Using Supervised Machine Learning and Feature Selection, Int. J. Advance Soft Compu. Appl, 7,3 (2015).

8. Huston, G., Web Caching, The Internet Protocol Journal, 2,3 (1999) 2-20.

9. Mehta, K., ve Jha, J. J., Web Cache Technique Responsive Web Design, International journal of innovative research in technology, 1,1 (2014) 29-31.

10. Kumar, C., ve John, B. N., A new approach for a proxy-level web caching mechanism, Decision Support Systems, 46,1 (2008) 52-60.

11. Cacere R., ve ark., Web proxy caching: the devil is in the details, ACM SIGMETRICS Performance Evaluation Review, 26,3 (1998) 11-15.

12. Ali, W., Shamsuddin, S. M. ve İsmail, A. S., A Survey of Web Caching and Prefetching, International Journal of Advances in Soft Computing and its Applications, 3,1(2011) 18-44.

13. Nottingham, M., Caching Tutorial for Web Authors and Webmasters. https://www.mnot.net/cache_docs/#DEFINITION 19 Aralık 2017.

14. Prusty, N., http://qnimate.com/all-about-web-caching/#Advantage_of_web_caching 18 Aralık 2017.

15. Carofiglio, G., Morabito, G., Muscariello, L., Solis, I., ve Varvello, M., From content delivery today to information centric networking, Comput. Networks, 57,16 (2013) 3116–3127.

16. Rabinovich, M., ve Spatschek, O., Web caching and replication, Addison Wesley Longman Publishing Co. Inc., Boston, USA 2002.

17. Park, K., ve Pai, V. S., Scale and Performance in the CoBlitz Large-File Distribution Service, 3rd Symposium on Networked Systems Design and Implementation (NSDI), San Jose, CA, 2006.

18. Lindsay, H., network control, visibility, management. https://lkhill.com/transparent-caching-wont-save-us/ 25 Kasım 2017.

19. Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D. K., ve Ohlman, B., A survey of information-centric networking, IEEE Commun. Mag., 50,7 (2012) 26-36.

20. Amadeo, M., Campolo, C., Quevedo, J., Corujo, D., Molinaro, A., Iera, A., Aguiar, R. L., ve Vasilakos, A. V., Information-centric networking for the internet of things: challenges and opportunities, IEEE Network, 30,2 (2016) 92-100.

21. Caida, http://www.caida.org/data/passive/trace_stats/ 12 Nisan 2017.

22. Callegati, F., Cerroni, W., ve Ramilli, M., Man-in-the-Middle Attack to the HTTPS Protocol, IEEE Secur. Privacy, 7,1 (2009) 78 - 81.

23. Zhang, J., Replacement Strategy of Web Cache Based on Data Mining, 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2015, 821-823.

24. Koskela, T., Heikkonen, J., ve Kaski, K., Web cache optimization with nonlinear model using object features, Comput. Networks, 43,6 (2003) 805–817.

25. Toly, C., Obtaining the optimal cache document replacement policy for the caching system of an EC website, European Journal of Operational Research, 181,2 (2007) 828–841.

26. Wong, K. Y., Web cache replacement policies: a pragmatic approach, IEEE Network, 20,1 (2006) 28-34.

27. Aybar, E., Optimizing both hit rate and byte hit rate in web cache replacement policies, Yüksek Lisans Tezi, Tech University, Texas, 2002.

28. Cherkasova, L., Improving WWW Proxies Performance with Greedy-Dual-Size-Frequency Caching Policy, Hewlett-Packard Laboratories, 1998.

29. Khaleel, M. S. A., Osman, S. E. F., ve Sirour, H. A. N., Improvement of least frequency used web cache replacement technology using intelligent agents, International journal of advanced research (IJAR), 5,1 (2017) 2589-2599.

30. Podlipnig, S. ve Boszormenyi, L., A survey of Web cache replacement strategies, ACM Computing Surveys (CSUR), 35,4 (2013) 374-398.

31. Han, J., Kamber, M. ve Pei, J., Data Mining: Concepts and Techniques, ELSEVIER, USA, 2011.

32. Larose, D. T., Discovering Knowledge in Data: An Introduction to Data Mining, John Wiley & Sons, USA, 2014.

33. Issenberg, S., How Obama's Team Used Big Data to Rally Voters, MIT Technology Review, New York, 2012.

34. Witten, I. H. ve Frank, E., Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, New York, 2016.

35. Voznika, F. ve Viana, L., Data Mining Classification. http://courses.cs.washington.edu/courses/csep521/07wi/prj/leonardo_fabricio.pdf 20 Aralık 2017.

36. Aggarwal, C. C., Data Classification: Algorithms and Applications, CRC press, New York, 2014.

37. MacQueen, J., Some methods for classification and analysis of multivariate observations, Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1967, 281-297 .

38. Silahtaroğlu, G., Veri madenciliği, Altan Basım, İstanbul, 2008.

39. Işık, M. ve Çamurcu, A. Y., K-means, K-medoids ve Bulanık C-means Algoritmalarının uygulamalı olarak performanslarının tespiti, İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi, 6,11 (2007) 31-45.

40. Sarıman, G., Veri Madenciliğinde Kümeleme Teknikleri Üzerine Bir Çalışma: K-Means ve K-Medoids Kümeleme Algoritmalarının Karşılaştırılması, Süleyman Demirel Üniversitesi Fen Bilimleri Enstitüsü Dergisi, 15,3 (2011) 192-202.

41. Tan, P. N., Steinbach, M., ve Kumar, V., Introduction to Data Mining, Addison Wesley, USA, 2006.

42. Xu, R., ve Wunsch, D., Survey of clustering algorithms, IEEE Transactions on neural networks, 16,3 (2005) 645-678.

43. Şeker, Ş. E., bilgisayarkavramlari. http://bilgisayarkavramlari.sadievrenseker.com/2012/01/29/istatistiksel-normallestirme-statistical-normalisation/ 25 Ekim 2017.

44. Mitchell, B., What Is Network Monitoring?. https://www.lifewire.com/what-is-network-monitoring-817816 13 Aralık 2017.

45. PAESSLER, PRTG Manual. https://www.paessler.com/manuals/prtg/introduction 12 Aralık 2017.

46. Lemma, E. S., Hussain, S. A. ve Anjelo, W. W., Performance Comparison of EIGRP/ IS-IS and OSPF/ IS-IS, Blekinge Institute of Technology, Karlskrona, 2009.

47. Cisco, http://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/16406-eigrp-toc.html 30 Mayıs 2017.

48. Zipf, G. K., Zipf 1949, Addison-Wesley, 1950.

49. Marin, M. C., 1985 Sonrası Türkiye'deki kentsel sistemin dönüşümü: Zipf yasasının ampirik testi, Gazi Üniversitesi Mühendislik Mimarlik Fakültesi Dergisi, 1,22 (2007) 33-38.

50. Boston University, (BU), The Internet Traffic Archive. http://ita.ee.lbl.gov/html/contrib/BU-Web-Client.html 3 Aralık 2016.

51. Kamiyama, N., Nakano, Y., ve Shiomoto, K., Cache Replacement Based on Distance to Origin Servers, Ieee Transactıons On Network And Servıce Management, 3,4 (2016) 848-859.

52. Negrão, A. P., Roque, C., Ferreira, P., ve Veiga, L., An adaptive semantics-aware replacement algorithm for web caching, Journal of Internet Services and Applications, 6,4 (2015) 1-14.

53. Waleed, A., Sarina, S., ve Norbahiah, A., Performance Improvement of Least-Recently-Used Policy in Web Proxy Cache Replacement Using Supervised Machine Learning., International Journal of Advances in Soft Computing & Its Applications, 6,1 (2014) 1-38.

54. Hiba, A. N., Yahia, A. M., ve Amir, A. E., Agent-based Proxy Cache Cleanup Model using Fuzzy logic, proceedings of International Conference of Computing Electrical and Electronic Engineering (ICCEEE), 2013, Khartoum, Sudan, 486-491.

55. Samiee, K., A replacement algorithm based on weighting and ranking cache objects, International Journal of Hybrid Information Technology, 2,2 (2009) 93-104.

56. Megiddo, N., ve Modha, D. S., Outperforming LRU with an adaptive replacement cache algorithm, Computer, 37,4 (2004) 58-65.

57. Sathiyamoorthi, V., A Novel Cache Replacement Policy for Web Proxy Caching System Using Web Usage Mining, Int. J. Inf. Technol. Web. Eng., 11,2 (2016) 1-13.

58. Arlitt, M., Cherkasova, L., Dilley, J., Friedrich, R., ve Jin, T., Evaluating content management techniques for web proxy caches, ACM SIGMETRICS Performance Evaluation Review, 27,4 (2000) 3-11.

59. Aggarwal, C. C., ve Reddy, C. K., Data clustering: algorithms and applications, CRC Press, Boca Raton, 2013.

60. PAESSLER Network Monitoring Company, https://www.paessler.com/prtg 10 Aralık 2017.

61. Jiang, B., Yin, J., ve Liu, Q., Zipf’s law for all the natural cities around the world, International journal of geographical information science, 29,3 (2015) 498-522.

62. Jiang, Q., Tan, C. H., Phang, C. W., Sutanto, J., ve Wei, K. K., Understanding Chinese online users and their visits to websites: Application of Zipf's law, International Journal of Information Management, 33,5 (2013) 752-763.

63. Rajaby, M. E., ve Çavdar, T., Enahanced Web Cache Replacement Policy Based On Data Mining and RFSD Scoring, 2nd World Conference on Technology, Innovation and Entrepreneurship, 2017, Istanbul, 293-298.

ÖZGEÇMİŞ

1991 yılında Afganistan'ın Cevizcan ilinde doğdu. İlk, orta ve lise öğrenimini Cevizcan 'da İbn-i-Yamin Lisesi'nde tamamladı. 2011 yılında Kabil Üniversitesi Bilgisayar Bilimleri bölümünden mezun olup farklı firmalarda bilişim teknoloji müdürü, IP mühendisi ve Afgan Kızıl Ay'da kıdemli bilişim teknoloji müdürü olarak çalıştı. 2014 yılında Türkiye bursları programına başvurdu ve aynı yılda bu programda kabul oldu. Bir yıl Türkçe eğitimi aldıktan sonra 2015 yılında Karadeniz Teknik Üniversitesi Bilgisayar Mühendisliği Bölümünde yüksek lisans eğitimine başladı. Ağlar alanında CCNA ve CCNP routing and switching sertifikalarına sahiptir ve iyi derecede İngilizce biliyor.