WAP TRAFİĞİNDE TIKANIKLIK DENETİMİ VE ULAŞIM KATMANI PROTOKOLLERİ

Sinan TOKLU

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

MAYIS 2007
ANKARA

Sinan TOKLU tarafından hazırlanan WAP TRAFİĞİNDE TIKANIKLIK DENETİMİ VE ULAŞIM KATMANI PROTOKOLLERİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Doç. Dr. M. Ali AKCAYOL
Tez Yöneticisi

Bu çalışma, jürimiz tarafından oy birliği ile Bilgisayar Mühendisliği Anabilim Dalında Yüksek lisans tezi olarak kabul edilmiştir.

Başkan: : Prof. Dr. Sezai DİNÇER

Üye : Doç. Dr. M. Ali AKCAYOL

Üye : Yrd. Doç. Dr. H. Şakir Bilge

Tarih : 28/05/2007

Bu tez, Gazi Üniversitesi Fen Bilimleri Enstitüsü tez yazım kurallarına uygundur.

TEZ BİLDİRİMİ

Tez içindeki bütün bilgilerin etik davranış ve akademik kurallar çerçevesinde elde edilerek sunulduğunu, ayrıca tez yazım kurallarına uygun olarak hazırlanan bu çalışmada orijinal olmayan her türlü kaynağa eksiksiz atıf yapıldığını bildiririm.

Sinan TOKLU

WAP TRAFİĞİNDE TIKANIKLIK DENETİMİ VE ULAŞIM KATMANI PROTOKOLLERİ

(Yüksek Lisans Tezi)

Sinan TOKLU

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

Mayıs 2007

ÖZET

Kablolu ve kablosuz ağlarda bazı HTML sayfalarına gelen aşırı talep durumlarında, ağda tıkanıklık oluşmakta ve sayfayı görüntülemek isteyen kullanıcı ya çok fazla beklemek zorunda kalmakta veya sayfaya erişememektedir. Bu tezde WAP hizmeti veren ağlardaki tıkanıklık problemini çözmek için bir protokol geliştirilmiştir. Geliştirilen protokol ns-2 benzetim aracı kullanılarak test edilmiştir. Benzetimde değişik kuyruk tiplerinin farklı yük yoğunluğundaki ağlarda çalışması incelenmiş ve elde edilen sonuçlar diğer ulaşım katmanı protokolleriyle karşılaştırılmıştır. Deneysel sonuçlar geliştirilen protokolün WAP trafiğindeki tıkanıklığı önlemede başarılı olduğunu göstermiştir.

Bilim Kodu : **902.1.014**
Anahtar Kelimeler : **WAP, Kuyruk Yönetimi, Tıkanıklık, Protokol**
Sayfa Adedi : **82**
Tez Yöneticisi : **Doç. Dr. M. Ali AKCAYOL**

CONGESTION CONTROL IN WAP TRAFFIC AND TRANSPORT LAYER PROTOCOLS

(M.Sc.Thesis)

Sinan TOKLU

GAZİ UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

May 2007

ABSTRACT

In wireless and wired networks some HTML pages take extreme request. Because of this congestion is occuring in networks and the users can not be access the HTML pages. In this thesis, a congestion prevention protocol has been developed for WAP networks with queue management. ns 2(network simulator) is used as simulation tool. In this simulation different queue types are being used in different weight of networks and experimental results are compared with eachother and with transport layer protokols. The result of this compare shows that developed protocol is gives more successful result.

Science Code : 902.1.014
Key Words : WAP, Queue Management, Congestion, Protocol
Page Number : 82
Adviser : Assoc. Prof. Dr. M. Ali AKCAYOL

TEŞEKKÜR

Çalışma hayatım boyunca yardım ve katkılarıyla beni yönlendiren ve bana destek olan değerli hocam Doç.Dr.M. Ali AKCAYOL'a, Yüksek Lisans eğitimimi yaparken yardımlarını esirgemeyen Doç.Dr. Şeref SAĞIROĞLU ve Yrd.Doç.Dr. Hasan Şakir BİLGE hocalarıma, hayatım boyunca bana her türlü desteği veren aileme ve Prof.Dr. Bilal TOKLU'ya teşekkür ederim.

SİMGELER VE KISALTMALAR

Bu tezde kullanılan simgeler ve kısaltmalar, aşağıda açıklamalarıyla belirtilmiştir.

Simgeler	**Açıklama**
α	Alçak geçiren filtre kazancı
β	Gidiş dönüş zaman değişimi
π	Pi sayısı
s	Saniye

Kısaltmalar	**Açıklama**
ACK	Acknowledgement
CGI	Common Gateway Interface
CNR	Communication Network Riser
CPU	Central Processing Unit
CWND	Congestion Window
FTP	File Transfer Protokol
HDTP	Handheld Device Transport Protocol
HDML	Handheld Device Markup Language
HTML	Hyper Text Markup Language
ICMP	Internet Control Message Protocol
ITTP	Intelligent Terminal Transfer Protocol
KG	Kannel Gateway
LVS	Linux Virtual Server
OMA	Open Mobile Alliance
OSI	Open System Interconnection
PAP	Push Access Control
POTA	Push Over The Air
PPG	Push Proxy Gateway
RTT	Round Trip Timing

Kısaltmalar	Açıklama
SMTP	Simple Mail Transfer Protocol
SRTT	Average Round Trip Timing
SSL	Secure Socket Layer
SWG	Scalable WAP Gateway
TCP	Transmission Control Protocol
TID	Transaction Identifier
TLS	Transport Layer Security
TTML	Tagged Text Markup Language
UDP	User Datagram Protocol
URL	Uniform Resource Locators
WAE	Wireless Application Environment
WAP	Wireless Application Protocol
WCMP	Wireless Control Message Protocol
WDP	Wireless Datagram Protocol
WLC	Weighted Least Connection
WML	Wireless Markup Language
WRR	Weighted Round Robin
WSP	Wireless Session Protocol
WTA	Wireless Telephone Application
WTAI	Wireless Telephone Application Interface
WTLS	Wireless Transport Layer Security
WTP	Wireless Transaction Protocol
XML	Xtensible Markup Language

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

Şekil **Sayfa**

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

1. GİRİŞ

Günümüzde yaygın olarak kullanılan mobil cihazlar ve bu cihazların sunduğu hizmetler hızla artmaktadır. Gün geçtikçe yeni özellikler eklenen bu cihazlar artık veri iletişiminde de kullanıcılarına hizmet vermektedir. Askeri uygulamalarda internet protokolü IP(Internet Protocol-İnternet Protokolü) kullanılması yönünde çalışmalar bulunmaktadır [1]. Gelecekteki savaşlarda çok büyük boyutlardaki bilginin çok hızlı bir şekilde işlenmesi öngörülmektedir.

Hareket kabiliyeti diğer önemli bir noktadır. İletişim yalnızca noktadan noktaya olmamalı, aynı zamanda broadcast ve multicast iletişime de izin vermelidir. Mobil paket ağlarındaki iletişim bağının karakteristik özelliklerinden önemli iki tanesi güvenilmezlik ve değişkenliktir. Günümüzde kullanılan uygulama ve protokollerin çoğu askeri uygulamalarda da kullanılmaya başlanmıştır [2]. Bunlardan biride WAP(Wireless Application Protocol-Kablosuz Uygulama Protokolü) protokol kümesidir. Askeri uygulamalarda yaygın olarak IP ağları için TCP(Transmission Control Protocol-İletim Kontrol Protokolü) ve UDP(User Datagram Protocol-Kullanıcı Datagram Protokolü) kullanılmaktadır.

Günlük hayattaki kullanımda mobil cihazların dar bant, küçük pencere, düşük hızlı CPU(Central Processing Unit-Merkezi İşlem Birimi) gibi özelliklerinden dolayı cihazlardaki veri iletişiminin daha hızlı ve güvenli yapılabilmesi için kullanılan protokoller kullanılamamaktadır[2]. Bundan dolayı mobil cihazlarda kullanılmak üzere web protokolleri temel alınarak WAP protokolleri oluşturulmuştur. Bu sayede mobil cihazların birbiriyle ve web sunucularla veri iletişimi önemli ölçüde geliştirilmiş ve hızlanmıştır [2]. Mobil cihazların ikili formatta çalışmasından ve karakteristik özelliklerinden dolayı genellikle WAP sayfaları metin halinde mobil kullanıcıya sunulmaktadır. Bu WAP sayfaları veya web sayfalarının mobil cihazlarda görüntülenebilmesi ve işlem yapılabilmesi için mutlaka bu çevrimleri yapacak ve iletişimi sağlayacak bir arabirime ihtiyaç duyulmaktadır [3].

Kablosuz hizmetlerin sunulması ve alınmasına yönelik çalışmalar 1990'lı yıllarda başlatılmıştır. 1995 yılında Ericsson kendine özel bir protokol geliştirmiştir. Bu protokol mobil ağda kullanılmış ve ITTP(Intelligent Terminal Transfer Protocol-Zeki Terminal Transfer Protokol) olarak adlandırılmıştır. Bu protokol uygulamanın bulunduğu hizmet noktası ile uygulamayı kullanabilen kablosuz cihaz arasında iletişimi sağlamaktadır. 1996 Unwired Planet, HDML(Handheld Device Markup Language) adında bir arabirim ve bunun içinde HDTP(Handheld Device Transport Protocol) adında bir protokol geliştirmiştir. 1997 Mart ayında ise Nokia, Smart Messaging'i tanımladı. Bu cihazlar kablosuz cihaz ile web sunucu arasında kısa mesaj servisi ve TTML (Tagged Text Markup Language-İşaretli Metin İşaret Dili) kullandılar [1,3].

Bütün bu firmaların kendine özgü yapılan protokolleri sadece kendi kablosuz cihazlarında desteklenmekteydi. WAP hizmetinin daha geniş alanlara ve bütün cihazları kaplayacak şekilde bir standart belirlenmesi gereksiniminin oluşması üzerine bu şirketler bir araya gelerek 1997 yılında WAP forumu oluşturdular [3].

WAP forumun amacı kablosuz mobil bir ortamda herkesin serbestçe bilgi transferi sağlaması ve uygulama geliştirmesidir. İnternet protokolleri WAP için geliştirilerek internet ve WWW tabanlı olarak tasarlanmıştır. Bu sayede çok büyük bir mobil kablosuz ağ protokolü oluşturulmuştur. WAP forum OMA(Open Mobile Alliance-Açık Mobil Uyum) içinde birleştirilmiştir ve herhangi bir firmaya bağlı değildir [4].

Kannel projesi Wapit Ltd. tarafından 1999 yılında bulunmuştur. WAP'ta tıkanıklığı çözmek için WAP gateway'lerin yapıldığı tarihten itibaren var olan yük dengeleme yeteneğinin açık kod olması bu projenin genel özelliğidir. Kannel Gateway'ler kümeleme mimarisinin temelini oluşturmaktadırlar [5].

WAP'ta tıkanıklığı çözmek için diğer bir donanımsal seçenek olan LVS(Linux Virtual Server-Linux Sanal Sunucu) gerçek sunucuların oluşturduğu kümede kurulumdan itibaren olan ve yüksek derecede ölçeklenebilir özelliklerine sahip olan

bir sunucudur. Aynı zamanda bu sistemde yük dengeleyicisi Linux işletim sisteminde çalışmaktadır [6].

2004 yılında Te-Hsin Lin, Kuochen Wang, Ae-Yun Liu SWG(Scalable WAP Gateway-Ölçeklenebilir WAP Gateway) adında bir makale yayınlamışlardır. SWG bir grup gerçek gateway içermekte ve bunlar birbirine hızlı bir ağda bağlı bulunmaktadırlar. Bu kannel-level front-end distributör'ü bütünleştirmekte ve WAP dispatcher olarak adlandırılan yeni bir mekanizma sunmaktadır [6].

Robert Shorten, Chris King, FabianWirth, Douglas Leith 2006 yılında "Modelling TCP congestion control dynamics in drop-tail environments" adında makale yayınladılar. Bu makalede TCP tıkanıklığını control için drop-tail kuyruk yapısını kullanarak yeni bir yaklaşımı açıkladılar [7].

Kyeong Hur, Doo-Seop Eom, Yeon-Woo Lee, Jae-Ho Lee, Seokjoong Kang 2007 yılında "Priority forwarding for improving the TCP performance in mobile IP based networks with packet buffering" adında bir makale yayıladılar. Bu makalede mobil IP ağlarında kullanılan hareketli düğümlerin yönlendirme yaparak TCP performansını geliştirmek için yeni bir yaklaşım gösterdiler. Bu yaklaşımda kuyruk yapısı olarak Red kuyruk yapısı kullanılmıştır [8].

Jinsheng Sun ve Moshe Zukerman 2007 yılında "RaQ: A robust active queue management scheme based on rate and queue length" adında makale yayınladılar. Bu makalede kuyruk uzunluğuna göre yeni bir aktif kuyruk yönetimi yaklaşımı gerçekleştirilmiştir [9].

Shan Chen ve Brahim Bensaou 2006 yılında "Can high-speed networks survive with DropTail queues management?" adında makale yayınladılar [10]. Bu makalede yüksek hızlı ağlarda drop-tail kuyruk yapısı kullanılabilirli hakkında bir çalışma yapmışlardır.

Songtao Guo, Xiaofeng Liao, Chuandong Li, Degang Yang 2006 yılında "Stability analysis of a novel exponential-RED model with heterogeneous delays" adında bir makale yayınladılar. Bu makalede heterojen gecilmeler ile yeni üssel red kuyruk tipinin sağlamlık analizi yapılmıştır [11].

Fengyuan Ren, Chuang Lin, Xunhe Yin 2004 yılında "Design a congestion controller based on sliding mode variable structure control" adında makale yayıladılar. Bu makalede kayan mod değişkeni yapısında kontrol tabanlı tıkanıklık denetimi için yeni bir yaklaşım göstermişlerdir [12].

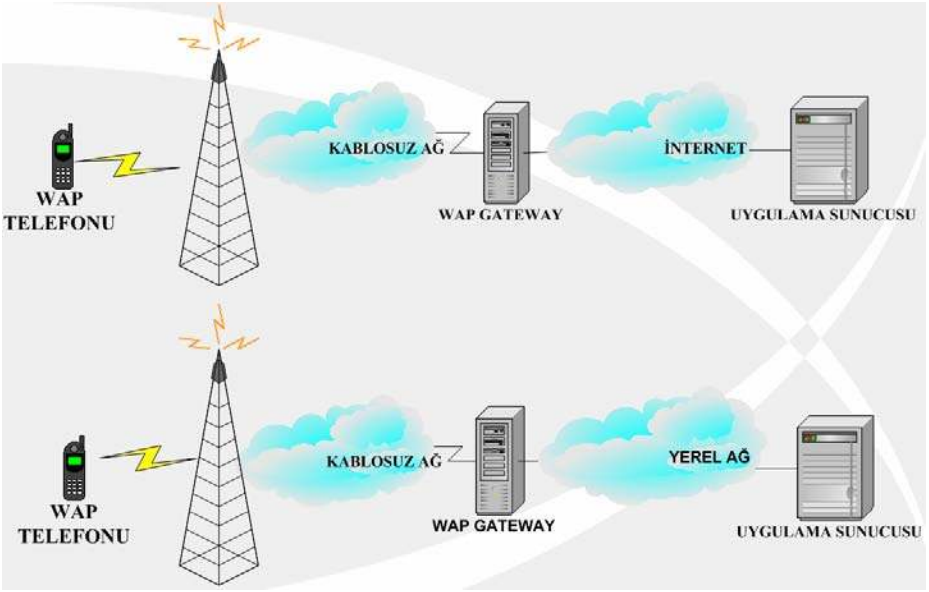
Mehmet Şimşek tarafından 2006 yılında kablosuz ağlarda yönlendirme protokolleriyle tıkanıklığın önlenmesi için çalışmalar yapılmıştır. Yapılan çalışmada, kablosuz ağlarda aşırı talep durumunda meydana gelen tıkanıklığın yük dengeleme ve yönlendirme işlemleri yapılarak önlenmesi için yeni bir yaklaşım sunulmuştur [13].

İbrahim Alper Doğru tarafından 2007 yılında Ad-hoc ağlarda hareketlilik yönetiminde tıkanıklık çözümüyle ilgili bir çalışma yapılmıştır. Ad-hoc ağlarda düğümlerin hareketliliğinden kaynaklanan ve aşırı talep durumunda meydana gelen tıkanıklığı gidermek için yeni bir protokol önerilmiştir [14].

2. WAP

WAP kablosuz iletişim sağlayan mobil telefonlara sahip kullanıcıların lisansa bağımlı olmaksızın internete erişimini sağlayan protokoldür. Ayrıca WAP kablosuz ağlar üzerinde uygulama geliştirme ve hizmet verme amaçlarına yönelik uluslararası kabul görmüş standart protokolleri içeren bir kurallar topluluğu olarak ta tanımlanır. WAP protokolleri esas olarak mevcut olan internet protokollerini temel alırlar. Ancak kablosuz mobil cihazlar ve uygulamalarla beraber çalışacak şekilde optimize edilmişlerdir [1].

WAP kablosuz araçlar için bir uygulama çatısı ve ağ protokolüdür. WAP mobil ünitelerle ağ hizmetlerinin iletişimi için standart araçları tanımlamaktadır. WAP'ın amacı bilginin ve hizmetlerin hızlı ve kolay bir şekilde mobil kullanıcılara sunulmasını sağlamaktadır [3]. Cep telefonları, avuç içi PC'ler, çift yönlü telsiz cihazları ve akıllı telefonlar gibi elde taşınabilen, dijital, kablosuz cihazlar WAP kullanabilmektedir. WAP PalmOS, EPOC, Windows CE, FLEXOS, OS/9, JavaOS gibi herhangi bir işletim sisteminin üzerine kurulabilir [2]. Şekil 2.1'de WAP mimarisinin genel yapısı gösterilmiştir.



Şekil 2.1. WAP mimarisinin genel yapısı

Şekil 2.1'de WAP hizmetini alabilmemiz için gerekli olan WAP telefonları ile WAP gateway ve uygulama sunucuları arasında bulunabilecek ağlar gösterilmektedir. Bu şekilde WAP telefonlar hareketli olduğundan dolayı bir baz istasyonu ve kablosuz ağın olması gerektiği görülmektedir. Uygulama sunucusuna internet üzerinden veya yerel ağ üzerinden ulaşılabilmektedir [1].

WAP'ın kullanılabilmesi için cihazların genel ihtiyaçları

- Her platformda kullanılabilen ortak standartlar(WAP hizmetine uyumlu olan bütün telefonların markası ne olursa olsun WAP hizmetinden faydalanabilmeleri)
- Esnek ve yeni gelişmelere açık bir mimari yapı(Kullanılan yazılım dillerinde gerçekleştirilen uygulamaların rahatlıkla sunucu veya istemcide uygulanabilmesi)
- Mümkün olduğunca çok kablosuz ağları destek
- Cihaz kaynaklarının optimizasyonu (düşük hafıza/CPU kullanımı/güç)
- Güveli uygulama ve haberleşmenin desteklenmesi
- İnsan makine arayüzünün olabildiğince esnek ve kullanışlı olması
- Gelecek üçüncü nesil için operatör ve network uygunluğunu kolaylaştırmak
- Program geliştirme ve uygulama [2].

WAP 'ın sunduğu bazı hizmetler

- Veritabanı hizmetleri
- Kişisel bilgilerin takibi
- E-Posta hizmetlerini
- İnternet üzerindeki bilgi hizmetleri
- Her türlü rezervasyon, erişim, satın alma…vb. hizmetleri sunmaktadır [3]

WAP 'ın avantajları

- WAP uluslararası şartnamesi sayesinde kullanıcılara kolay erişim ve servislerle etkileşim ve geliştirme olanağı sunar

• WML(Wireless Markup Language-Kablosuz İşaret Dili) sayfalarını oluşturmak ve uyarlamak kolaydır
• WAP ile ilgili çok fazla doküman webte ücretsizdir
• WAP taşıyıcılardan bağımsızdır. Herhangi bir network üzerinde çalışır [2]

WAP ‘ın dezavantajları

• Kullanıcı ara yüzü sınırlı ve esnek değildir
• Farklı tarayıcılar gelen sayfaları farklı şekilde gösterebilirler. Dolayısıyla görünüm tarayıcılara bağlıdır
• Farklı WAP hizmetleri ve portalları arasında internette yönlendirme zordur [2].

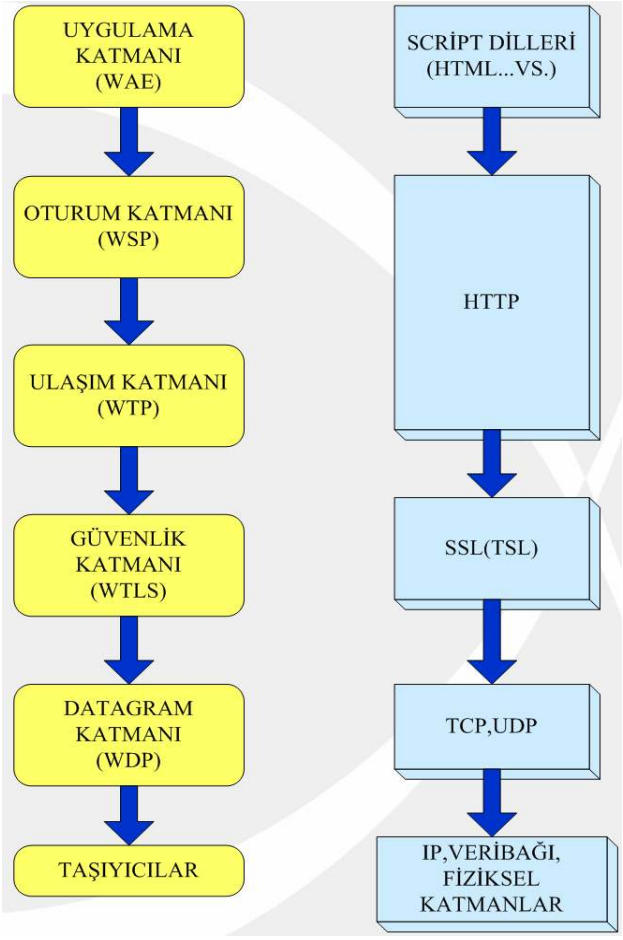
2.1. WAP Katmanları

WAP esnek bir yapıya sahiptir ve temel olarak beş bölüme ayrılmıştır. Bunlar:

• WAE (Wireless Application Environment-Kablosuz Uygulama Katmanı) : WML ve WML Script bu katmanda kullanılmaktadır. HTML ile hemen hemen aynı işlevi görmektedir. OSI’de sunum katmanına denk gelmektedir [3].
• WSP (Wireless Session Protocol-Kablosuz Oturum Katmanı) : Güvenli oturum açma, yönlendirme yeteneği, ayrıca bağlantı kopsa da askıya alınsa da tekrar kaldığı yerden devam etme yeteneği ile itme özellikleri bu katmanda kullanılmaktadır. HTTP ile hemen hemen aynı işlevi görmektedir. OSI’de oturum katmanına denk gelmektedir [2].
• WTP (Wireless Transport Protocol-Kablosuz Ulaşım Katmanı) : İşlemlerin tutulması ve yönetilmesi, yeniden iletim, birden fazla aynı ACK veya paketin silinmesi, birleştirme ve bölme özellikleri bu katmanda kullanılmaktadır. HTTP ile hemen hemen aynı işlevi görmektedir. OSI’de ulaşım katmanına denk gelmektedir [3].
• WTLS (Wireless Transport Layer Security-Kablosuz Ulaşım Katmanı Güvenliği) : Sıkıştırma, şifreleme ve kimlik doğrulama özellikleri bu katmanda kullanılmaktadır. TLS (Transport Layer Security-Ulaşım Katmanı Güvenliği) / SSL (Secure Socket

Layer-Güvenli Soket Katmanı) hemen hemen aynı işlevi görmektedir. OSI'de ağ katmanına denk gelmektedir [3].

• WDP (Wireless Datagram Protocol-Kablosuz Datagram Protokolü) : Port numarasına göre adresleme, parçalarına ayırma ve parçaları tekrar birleştirme, hataları yakalama gibi özellikler bu katmanda kullanılmaktadır. TCP/IP hemen hemen aynı işlevi görmektedir. OSI'de veri iletim katmanına denk gelmektedir [1]. Şekil 2.2' de WAP protokollerinin web protokolleri ile denklikleri gösterilmektedir.



Şekil 2.2. WAP protokollerinin web protokolleri ile denklikleri

Şekil 2.2'de görüldüğü gibi web protokollerinin en üstünde bulunan HTML, script gibi yazılımlar WAP mimarisinde uygulama katmanına denk gelmektedir. Web protokolünde kullanılan HTTP ise WAP katmanında ulaşım ve oturum katmanına denk gelmektedir. Web'te kullanılan bir diğer protokol olan TCP ve UDP ise WAP

mimarisinde datagram katmanına denk gelmektedir. En alt kısımda iki mimaride de taşıyıcılar kullanılmaktadır.

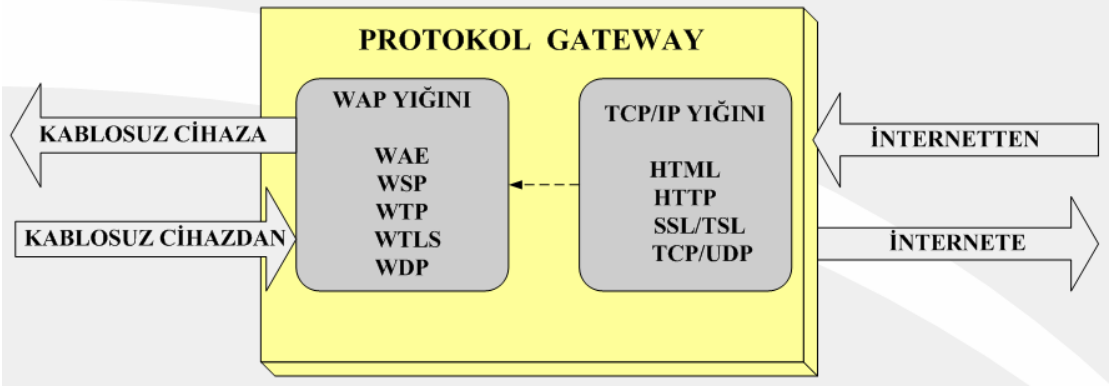
2.1.1. WAE

WAP hiyerarşisinde en üstedir. Temel amacı operatörlerin farklı kablosuz ortamların kolay bir şekilde erişebileceği uygulamaları yapmasına olanak sağlayan bir ortam kurmaktır. Genel olarak WWW teknolojisini temel almaktadır. Mobil üniteler küçük hafıza ve basit işlemleri yapma kapasitesine sahip oldukları için sunucu ve mobil ünitelerin birbirini anlaması için bazı kod şifreleme ve kod çözme işlemleri yapılmaktadır. Bu işlemi yapan cihazlara WAP gateway denmektedir [16].

WAE iki tane mantıksal katmana ayrılmıştır. Kullanıcı ajanı (user-agent) katmanı en üst kısmı oluşturmaktadır ve tarayıcılar, telefon defterleri gibi elemanları içermektedir. Servis ve format katmanı ise kullanıcı ajanı altında çalışmaktadır ve kullanıcı ajanının ihtiyacı olan hizmeti sağlamaktadır (WML ve WML script gibi). En fazla kullanılan kullanıcı ajanı WML kullanıcı ajanıdır [3]. Fakat WAE'de tek bir kullanıcı ajanı yoktur. İhtiyaca göre farklı kullanıcı ajanları tanımlanmıştır. Örneğin telefon uygulamaları için WTA(Wireless Telephone Application-Kablosuz Telefon Uygulama) kullanıcı ajanı kullanılmaktadır [17]. WAE genel olarak aşağıdaki şekilde kullanılmaktadır;

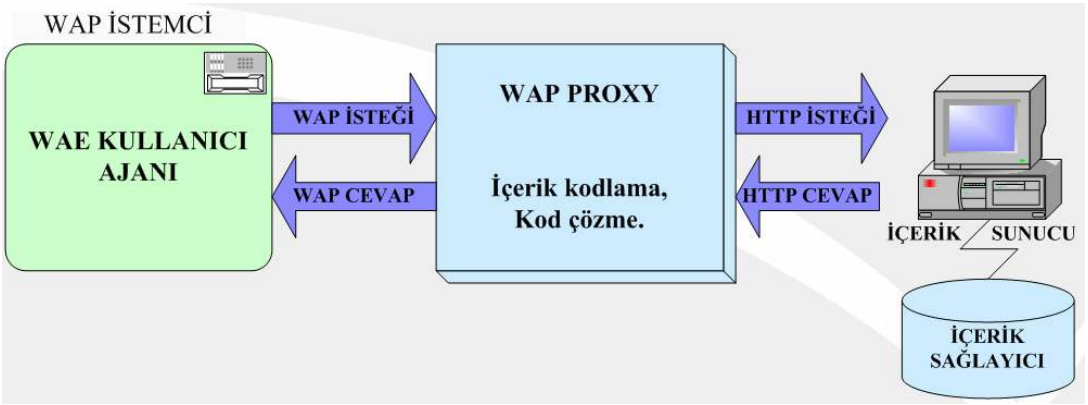
- Adresleme Modeli: WAP internette kullanılan adresleme sisteminin benzerini kullanmaktadır.
- Wireless Markup Language (WML) : XML'e dayanmaktadır.
- WBMP(Wireless Bitmap): Resim standardıdır. WAP tarafından desteklenen tek resim formatıdır.
- WML Script: WML'in tamamlayıcısıdır. Java Script'e benzemektedir.

Şekil 2.3'te WAP istemci ve protokol gateway arasındaki ilişki gösterilmiştir.



Şekil 2.3. WAP istemci ve protokol gateway

Şekil 2.3'te WAP yığınının TCP/IP yığını arasında iletişim olurken çevrimlerin nasıl olacağı gösterilmiştir. İnternetten gelen veriler mutlaka WAP mimarisinin anlayacağı şekle çevrilmelidir. Kablosuz cihazdan gönderilen veriler ise TCP/IP yığınında anlaşılacak şekle çevrilmelidir. Şekil 2.4 'te ise WAE kullanıcı ajanı ile http arasındaki ilişki gösterilmektedir.



Şekil 2.4. WAE kullanıcı ajanı ile HTTP arasındaki ilişki

Şekil 2.4'te görüldüğü gibi WAE kullanıcı ajanından gelen bir WAP isteği İlk önce WAP proxy'ye gönderilmeli ve burada yapılması gereken çevrimler yapıldıktan sonra içerik sunucusuna gönderilmesi gerekmektedir. İçerik sunucusundan cevap gönderileceği zaman ise önceki yapılan işlemin tersi yapılarak WAP istemciye cevap gönderilmektedir.

Mobil telefonların ekranlarının küçük olması, bellek kapasitesinin yetersiz olması…vs. gibi özelliklerinden dolayı verilerin ikili modda olması gerekmektedir. Bu ikili modda yazım WML, WML script…vs. yazılım dilleriyle yapılmaktadır. WAP gateway'ler gelen HTML…vb. yazılım dilinde yazılmış formatları ikili formata yani WML'e çevirmektedirler. Böylece mobil kullanıcı içeriği kendi ekranında görebilmektedir [18]. Dolayısıyla WAP istemci istediği veriyi bir web sayfasından alması gerekiyorsa muhakkak isteğini WAP gateway'e iletmesi gerekmektedir. Bu çevrim işlemi WAP gateway'de iki yönlü olarak çalışmaktadır. WAP gateway içerisinde bulunan WAE bileşenleri ve temel elemanları bu işlemleri kullanıcı kısmı ve içerik sağlayıcılar ile yapmaktadırlar ve bu yapı WAP gateway'lere entegre edilmiştir. WAE sadece tasarım esnasında kullanılacak temel hizmetleri ve formatları tanımlar. Bunun yanında geliştirme kullanıcıların hayal gücüne bağlıdır [17].

WAE'nin temel elemanları

WAE kullanıcı kısmı: Bu yapı WML için kullanılmaktadır. WAP mimarisine entegre edilmiştir. Kullanıcılar bu sayede içeriği görebilirler ve uygulama yapabilirler. Tarayıcı, telefon listesi mesaj editörü…vb. oluşmaktadır.

İçerik sağlayıcılar: Mobil kullanıcının isteği için gerekli olan uygulamayı ana sunucudan gönderilmesi işleminde kullanılır. WAE herhangi bir standart içerik üreticisini tanımlamaz. Bunun yerine merkezi sunucuda yer alan servisler ile gelen isteğe standart bir yanıt üretmek için kullanılır [17].

Standart içerik kodlama: Sıkıştırma işlemidir. Mobil kullanıcı tarafından gelen isteğin veya mobil kullanıcıya sunucudan giden hizmetin okunup, yorumlanabilmesi ve içerik değişimi için şarttır. Gelen WML ve WML Script kodlarının HTML 'e çevrilmesi veya tam tersi işlemlerde kullanılmaktadır [17].

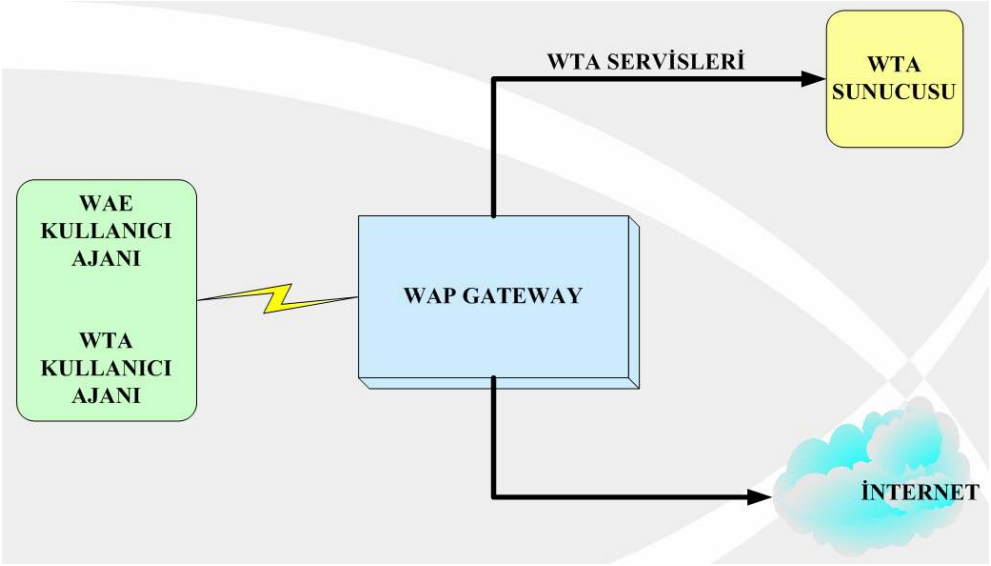
WTA

WTA ve itme özelikleri WAP'ın değerini arttıran önemli iki özelliktir. WTA telefon hizmetleri için bir uygulama çatısıdır. WTA cihazın bir telefon olduğunu göz önünde bulundurarak, telefonun bazı özelliklerinden veya mobil ağdaki hizmetlerden faydalanmayı sağlayan fonksiyonları içerir. Bu fonksiyonlar arama yapma ve arama kabul etme gibi işlemlerdir. WTA internete mobil cihazların fonksiyonlarını kullanabilme özelliği vermektedir. WTA bir grup fonksiyon sayesinde WAP ve WML sayfalarının telefon hizmetleriyle beraber çalışmasını sağlar. Bu işlemlerin yapılabilmesi için WAP uyumlu mobil cihazların WTA hizmet birimlerine sahip olmalıdırlar. Fakat WTA birinci nesil WAP uyumlu mobil cihazlarda uygulanmamıştır (WAP 1.1) ancak 2.nesil WAP uyumlu cihazlarda tam olarak uygulanmaya başlanmıştır (WAP 1.2) [19].

Bütün bu yerel ve ağ hizmetlerinin çalıştırılabilmesi WTA kullanıcı ajanı olarak adlandırılan bir eleman tarafından yapılmaktadır. Bütün bu yerel ve ağ hizmetlerinin fonksiyonları operatör ağda bulunan bir WTA sunucuda bulunan kütüphanelerde tutulmaktadır. WTAI (Wireless Telephone Application Interface-Kablosuz Telefon Uygulama Arayüzü) olarak adlandırılan bir arayüz bu yerel ve ağ fonksiyonlarının WTA kullanıcı ajanı ile etkileşimini sağlamaktadır. Bütün bu tanımlar aşağıdaki şekilde gösterilmiştir. Bir WTA hizmetine ulaşmak istediğimiz zaman, WTA kullanıcı ajanı gateway'e kullanmak istediğimiz fonksiyon ile içinde bulunduğu kütüphanenin adını içeren bir istek mesajı yollar. Gateway fonksiyon kodunu WTA sunucudan ister ve aldığı cevabı WTA kullanıcı ajanına yollar. Bundan sonra WTA kullanıcı ajanı WTAI'nın yardımıyla bu gelen kodu çalıştırır [19].

Her ihtiyaç olduğunda devamlı bu işlemleri yapmak zaman israfına sebep olmaktadır. Bu sebepten dolayı mobil cihazda sıkça kullanılan fonksiyon kodlarının mobil cihazda saklanması sağlanabilmektedir. Ancak bu işlemde fonksiyon kodlarının boyutunun ne kadar olduğu WAP forum tarafından açıklanmamıştır. Aşağıdaki şekil WTA'nın kullanımını göstermektedir[20]. Burada birincil olarak

kullanıcı isteği muhakkak WAP gateway'e yapmak zorundadır. Şekil 2.5'te WAE ve WTA kullanıcı ajanları arasındaki ilişki gösterilmiştir.



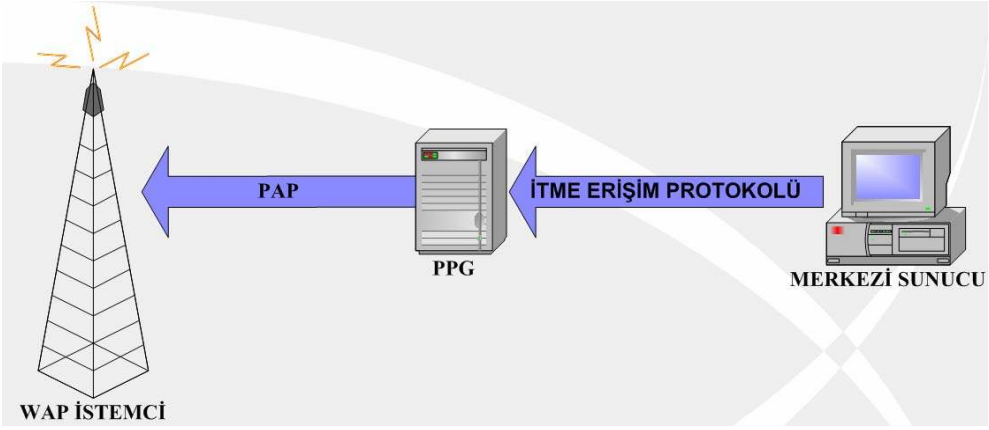
Şekil 2.5. WAE ve WTA kullanıcı ajanları arasındaki ilişki

Şekil 2.5'te gösterildiği gibi WAP isteği WAE veya WTA kullanıcı ajanlarından herhangi birinden gelse de mutlaka WAP gateway'de çevrimi yapılıp ilgili sunuculara isteğin gönderilmesi gerekmektedir. İnternet talepleri için WAE kullanıcı ajanı ile istek internet sunucusuna ve WTA kullanıcısı ile yapılan talepler ise WTA servisleri kullanılarak WTA sunucunsa gönderilmesi gerekmektedir.

WTA kullanıcı ajanı kısmında ayrıca yığın olarak bilinen bir depolama merkezide kullanılmaktadır. Yığın WTA fonksiyonlarının başlatılabilmesi için gerekli kod içeriğini tutmaktadır. Bu yığındaki içerik, WTA sunucusundan yığına yüklenir ve ihtiyaç halinde kullanıcı kısmı bu içeriği yığından ister bu şekilde çalıştırılmak istenen fonksiyon çalıştırılabilir. Böylece direk sunucudan istenen fonksiyon çalıştırma kodları için harcanan ağ trafiği yığından istenerek azaltılmaktadır. Bu işlemler WTA servisleri tarafından yapılmaktadır [20].

İtme

İtme teknolojisi WAP sayfalarını veya gerekli olan ayarlamaları sunucuya herhangi bir istek gelmeden mobil istemcilere gönderilmesini sağlamaktadır. Bu sayede yeni birçok hizmet sağlanmıştır. İtme teknolojisi sayesinde WAP mimarisine yeni eleman eklenmiştir. Bu eleman PPG(Push Proxy gateway-İtme Proxy Gateway)'dir. Sunucu itme mesajını PPG'ye yollar oda bu mesajı WAP terminale gönderir. PPG ve WAP gateway aynı ağda bulunabilmektedir [19]. Şekil 2.6'da itme olayında gerçekleşmesi gereken işlemler gösterilmektedir.



Şekil 2.6. İtme olayı

Şekil 2.6'da gösterildiği gibi WAP istemciye gönderilmesi gereken bir mesajın merkezi sunucu tarafından itme erişim protokollerini uygulayarak mesajı PPG'ye göndermesi onunda PAP(Push Access Control-İtme Erişim Kontrolü) servislerini kullanarak mesajı WAP istemciye göndermesi gerekmektedir. Bu yeni mimari beraberinde 2 tanede protokol getirmiştir. Bunlar;

- PAP
- POTA

PAP sunucu tarafından PPG ile iletişim için kullanılır. Bilgileri proxy'ye yollamak için HTTP 1.1 ve HTTP POST metodunu kullanmaktadır. PAP diğer ve gelecek protokoller ile kanal kurabilecek şekilde dizayn edilmiştir. Kanal diğerlerinin hangi protokolleri kullandığına bakmaksızın bunlar üzerinden protokol mesajları

gönderme yeteneğine denmektedir [19]. PAP diğer internet protokolleri olan FTP(File Transfer Protokol-Dosya Transfer Protokolü) veya SMTP(Simple Mail Transfer Protocol- Basit Mail Transfer Protokolü) gibi protokoller ile uyumluluğu ve ölçeklenmeyi sağlayabilecek şekilde tasarlanmıştır. PAP kullanarak sunucunun yapabileceği bazı operasyonlar;

- PPG üzerinden bilgileri mobil kullanıcılara iletmek.
- İtme başlatıcı, kimlik tanıma ve doğruluğunu kanıtlama işlemi yapar.
- İçerikte hata denetleme mekanizması.
- Adres çözümlemesi yapar.
- Protokol çevirimi yapar.
- İkili kodlama ve kod çözme işlemini yapar.
- İstemci bulma işlevini gerçekleştirir.
- Önceki itme işleminin durumunu okuyabilir. Bu sunucuya gönderilen itme mesajının mobil kullanıcıya gidip gitmediğini gösterir.
- Mesajın kullanıcıya ulaştığı sunucuya bildirilir. Bu işlem öncekiyle benzerdir, farkı ise mesaj terminale ulaştığı hakkında PPG'den bilgi istemektedir.
- Önceki itme işlemini iptal edebilir.
- PPG'den belirli cihaz hakkında bilgileri isteyebilir [20].

POTA protokolü WSP tabanlıdır ve PPG tarafından bilgilerin mobil kullanıcıya gönderilmesini sağlamaktadır. PPG genel olarak WAP gateway ile aynı fonksiyonları taşımaktadır. Örneğin;

- Protokol Gateway rolü üslenmektedir. PAP'tan gelen mesajları POTA protokolüne çevirmektedir.
- İtme mesajlarının hangi cihaza gideceğine karar vermek için adresleri ayrıştırır ve ilgili cihaza yerleştirir.
- Ağdaki verilen cihaz hakkında bilgileri toplar ve istemci tarafından bir istek olursa sunucuya gönderir. Bu işlem bilgilerin cihazın karakteristiğine göre yollanması için gerekli ayarlamaların sunucu tarafından yapılıp mesajın cihaza göre hazırlanmasını sağlamaktadır.

• Havada verinin iletişimi için gerekli formata göre içeriği şifreler.

PAP

PAP internet tabanlı PUSH başlatıcı sayesinde içeriğin PPG üzerinden mobil ağa itilmesi olayına denmektedir. PAP aşağıdaki işlemleri yapmaktadır:

• İtme başlangıç: İtme yapılacak mesaj itme-başlatıcı tarafından gateway'e gönderilir. Bu mesaj XML dokümanıdır. Gateway bu mesajı alınca XML tipinde başlatıcıya cevap mesajı gönderir.
• Sonuç Bildirisi: Mobil cihaza itme yapılan mesaj ulaştığında, itme başlatıcıya başka bir XML sonuç mesajı gönderilir.
• PUSH İptali: Önceden gönderilmiş olan mesajların silinmesi için PUSH başlatıcının yaptığı iptal işlemidir. Sonuç ACK'ları XML tipindedir.
• Durum Sorgusu: İtme başlatıcıya, önceden gönderilmiş mesajın durumu hakkında bilgi isteme hakkı vermektedir. Bütün istek ve cevap ACK'ları XML olarak hazırlanmaktadır.
• İstemci kapasite Sorgusu: İtme başlatıcıya, cihazın kapasitesini ölçmek ve öğrenmek için PPG'ye sorgu yollama hakkı vermektedir [19].

PPG'ye erişim için adres kullanılmaktadır. HTTP'deki URL(Uniform Resource Locators) ve XML'de tutulan adresler gibi. İstemciye iki şekilde mesaj gönderilebilir. Biri belirli bir adresten diğeri broadcast olarak gönderilmektedir [19].

2.1.2. WSP

WSP'nin temel amacı mobil cihazlar ile sunucunun veri iletimini kolay ve hızlı yapabilmeleri için bir oturumun açılmasıdır. İki tip protokol içermektedir. Bunlar;

• Bağlantılı tabanlı oturum hizmetleri-İşlemler WTP üzerinden olmaktadır.
• Bağlantısız oturum hizmetleri-İşlemler WDP üzerinden olmaktadır.

Oturum hizmetleri WSP'nin sağladığı kurala bağlı olmayan hizmetler üzerinden yapılmaktadır. Kurala bağlı olmayan hizmetler istemcinin sunucudan kullanmak

istediği fonksiyonun özelliklerini bildirdiği bir mesaj olarak tanımlanmıştır. Örneğin; bir kurala bağlı olmayan hizmet, s-connect'tir ve sunucu ile bağlantı oluşturmak için kullanılan mesajlardır [2].

Uygulama katmanı ile geriye kalan protokol dizini arasında bir arayüzdür. WSP HTTP 1.1'in ikili versiyonudur. Ekstra bazı özellikleri vardır. Bunlar;

- Uzun Süreli Oturum: Mobil cihazların sınırlı CPU, küçük pencere ve dar bant gibi özelliklere sahip olmasından dolayı kurulacak oturumun uzun süreli olması önemlidir. WML olarak gelen sayfalar mobil cihazlarda card olarak gösterildiklerinden dolayı, bağlantının mobil cihazın özelliklerine göre uzun süreli olması gerekmektedir [3].
- Veri itme: Sunucunun istemcinin haberi olmadan veya haberdar edip onay alarak bazı sistem fonksiyonlarının sunucu tarafından mobil cihaza itilmesi/gönderilmesi olayına itme denir [1].
- Paketlerin nasıl gideceği hakkında iki tarafın el sıkışmasına denir. Paketlerin hangi portları kullanacağı hangi protokollerin uygulanacağına karar verilmesi burada yapılmaktadır [2].

WSP'nin diğer özellikleri;

- HTTP 1.1'in işlevselliğini sağlar
- İstemci ve sunucu başlıklarının değiş tokuşu
- Prosesteki hareketleri kesme
- Sunucudan istemciye eşzamansız olarak içerik iletme

<u>WSP'nin oturum hizmetleri</u>

WSP'nin iki tip oturum hizmeti vardır. Bunlar;

Bağlantılı hizmet modu: Bağlantılı hizmet modu oturumu WTP üzerinde çalışmaktadır. Bağlantı modu oturumunun ana fonksiyonu, istemci ile WAP gateway/Proxy arasında oturumu kurmaktır. Gönderilen mesaj cevaplanmalıdır. Eğer mesaj kaybolursa tekrar gönderilmelidir [3]. Sağladığı özellikler ise;

- Oturum yönetimi: Mobil kullanıcının da dâhil olduğu bu bölüm oturum ile ilgili özelliklerin, kontrolü ve yönetimi, protokollerin seçimi ve sunucu ile bağlantı gibi işlevsellikler kazandırmaktadır. Ayrıca sunucudan gelen bağlantı isteklerini ret gibi birçok özelliğin kullanımını kullanıcıya sağlamaktadır [1].
- Metot başlatma imkânları: İstemcinin sunucudan bir işlem için gerekli bilgileri almasına olanak sağlamaktadır. Genelde HTTP metotları ve kullanıcı tarafından tanımlanan ekleme işlemleri kullanılmaktadır. Tanımlanan ekleme işlemleri sayesinde her iki tarafta işlemin tamamlanması sonucu oluşan başarılı veya başarısız sonuç mesajlarını alabilmektedir [2].
- Hata raporlaması: Sunucunun istemciye işlem esnasında oluşabilecek hataları veya işlemle alakası olmayan durumları bildirmesi olayına hata raporlaması denilmektedir. Hata iki taraflı olabilmektedir [3].
- İtme: Mobil kullanıcının bilerek veya bilmeyerek istemediği bazı özelliklerin ve fonksiyonların işlemde en üst seviyede verim alınabilmesi için sunucu tarafından mobil cihaza itilmesi olayına denilmektedir [3]. İki tür itme yapılmaktadır. Bunlar;

a.Onaylanmış İtme: Burada normal itmeden farkı sadece sunucu tarafından gönderilen içeriğin mobil kullanıcının isteği ve onayı halinde yapılması olayına denmektedir.

b.Oturumların kaldığı yerden devam etmesi: Herhangi bir sebepten dolayı bağlantının kopması durumu ihtimaline karşı bu işlem sayesinde bir önlem alınabilmektedir. Mobil cihazların özelliklerinden dolayı veya taşınabilir olmasından dolayı sunucu ile bağlantının kesilme riskine karşı herhangi bir olumsuz durumda bağlantının askıya alınmasını sağlayan bu özellik kullanıcının tekrar başlatma isteği ile kaldığı yerden bağlantı devam edebilmektedir [2].

Bağlantısız hizmet modu

Bağlantısız hizmet modu datagram ulaşım hizmetinin üzerinde çalışmaktadır. Bağlantısız hizmet modu ise sadece doğrulanmamış araç, gereçleri sağlar. Gönderilen mesajların cevaplanmadığı durumlarda bu kullanılır. Kullanılacak metotların çağrılması ve aktif hale gelmesi ve itmede kullanır. Güvensizdir. WSP veri transferi için 3 itme mekanizması sağlar. Bunlar;

- Onaylanan veriyi itme: Var olan bir oturumdan sunucudan istemciye her an doğrulanmış verinin itilmesini sağlar.
- Onaylanmayan veriyi itme: Var olan bir oturumda onaylama olmadan doğrulanmamış verinin itilmesini sağlar.
- Bağlantısız hizmet modu ile veri itme: Var olmayan bir oturumdan dolayı tek bir yöne güvenilmez ulaşımla verinin itilmesi için kullanılır.

WSP'deki protokoller taşıyıcı ağındaki düşük bant genişliği ve uzun gecikmeyi en iyi şekilde kullanmaktadır [2].

2.1.3. WTP

WTP WAP mimarisinin protokol katmanıdır. İstemci ile sunucu arasında kablosuz bir bağlantı ile WSP veri paketlerinin güvenli bir şekilde transferini sağlar. İsteklerin gönderilmesi, cevaplanması ve bilgi işlemleri için gerekli olan hizmetleri sağlamaktadır [3]. WTP, TCP ve UDP'nin WAP'taki karşılığıdır. Ayrıca WTP, WSP'iyle güvenli bir şekilde bağlantıya geçmeyi sağlar. Bağlantı güvensiz olursa, bağlantıyı güvenli hale getirmek için tekrar transferi gerçekleştirmekten WTP sorumludur [21]. Özellikleri;

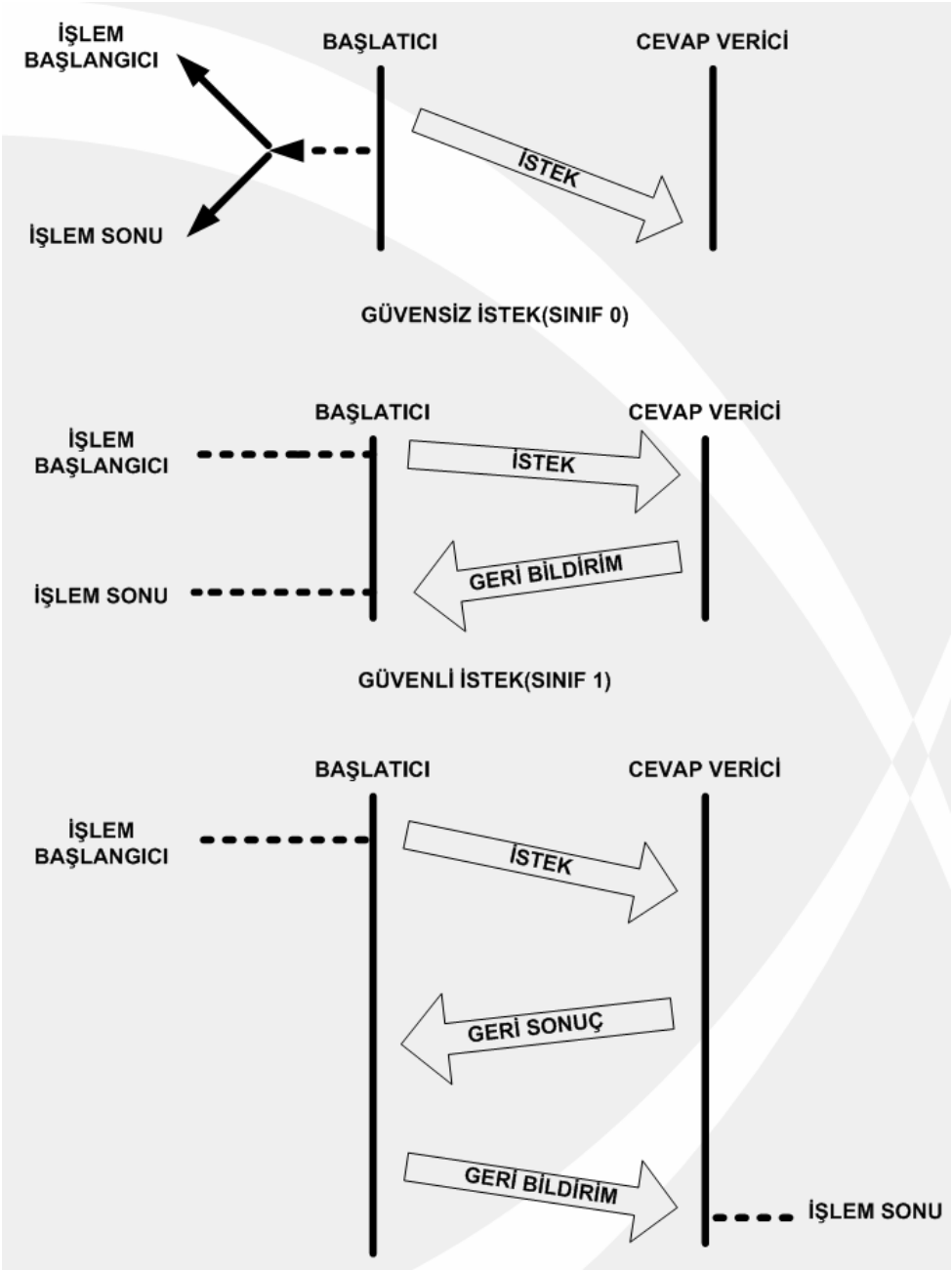
- Datagramlar güvenilir değildirler. Ancak WTP güvenlidir.
- WTP iletim ve ACK gerektirmektedir. (Kullanıcılarına güvenli iletişim sağlamak için.)
- WTP' de bir güvenlik mekanizması yoktur.

• WTP iletim ve alıcı olan mesajların kontrolünden sorumludur ve gerekli olan etkileşimli tarama uygulamasını sağlamaktadır.

WTP 3 tane farklı işlem sınıfı ile çalışmaktadır. İşlem sınıfı ise başlatıcı tarafından oluşturulur ve cevaplayıcıya gönderilen istek mesajında gösterilir.

• Sınıf 0: Güvenilmez istek mesajlarıdır. Bir sonuç mesajı yoktur. Bu mesaj tipi güvensiz datagram hizmetini sağlamaktadır. Mesaj kaybolursa tekrar iletim yoktur. Güvensiz itme gerektiren uygulamalarda kullanılır. Başarısız olma ihtimali yoktur [2]. (Oturum yönetimi, oturumu yeniden başlatma, itme).
• Sınıf 1: Güvenli datagram hizmeti içindir. Alıcıya bağlı olarak ACK gönderilir. Mesajın kaybolması veya iletilememesi durumu için mesaj saklanır. Güvenli PUSH gerektiren uygulamalarda kullanılır. Herhangi bir zamanda başarısız olabilir [1]. (Doğrulanmış itme).
• Sınıf 2: Temel güvenilir istek mesajları, güvenli sonuç mesajını ve temel programları çağırma ve cevaplama işlemlerini sağlar. Bir WSP oturumu bu tipin birçok hareketini içerir [2]. (Oturum yönetimi, metot başlatma, oturumu yeniden başlatma). Şekil 2.7'de WTP'de kullanılan sınıfların mesaj işlemleri gösterilmektedir.

Şekil 2.7'de WTP'de bulunan 3 sınıf için mesaj işlemlerinin genel şekli gösterilmektedir. Burada sınıf 0 için işlem başlangıcı işlemi sayesinde ve başlatıcı kullanılarak mesaj gönderilir, ancak cevap verici bu mesaja cevap vermeyebilir. Sınıf 1'in sınıf 0'dan farkı ise cevap vericinin istek mesajına cevap vermesi gerektiren durumlarda kullanılmasıdır. Sınıf 2 ise cevap verici istek mesajına cevap gönderdikten sonra gönderilen cevap mesajının alınıp alınmadığına dair cevap vericiye bir geri bildirimin olmasını gerektiren durumlarda kullanılmaktadır [21].



Şekil 2.7. WTP'de kullanılan sınıfların mesaj işlemleri

TID

Kaynak ve hedefin adresi ve port numarası tek bir işlem ile adlandırılır ve buna işlem tanımlayıcı denir. TID(Transaction Identifier-İşlem tanımlayıcı) her başlangıç işleminde arttırılır. Böylece gönderilen ve gönderilemeyen mesajlar hakkında bilgi

sahibi olunur ve hata oluşursa veya tekrar gönderimi gerektiren durum olursa TID kullanılır [21].

Mesaj transferi

- Davet Mesajı: Bu her zaman ilk gönderilen mesajdır. Eğer bir davet mesajı gönderilirse gönderici TID'ini arttırır. Sonra gönderici belirli olan bir zamanı başlatır ve bir ACK bekler.
- Doğrulama: Alıcı aldığı davet mesajı için TID numarasını tampon bellekte tutar. Bunun sebebi tekrar aynı mesajın gelmesini önlemek içindir.
- ACK Tutma: Eğer alıcı davet mesajı geldiğinde meşgulse bir ACK tutma mesajı gönderir. Bu zaman dolunca göndericinin tekrardan aynı mesajı göndermesini engeller.
- Sonuç Mesajı: Alıcı veriyi alınca bir ACK yollar ve mesajın ulaştığına dair bir ACK bekler.
- Son ACK: Gönderici mesajın alıcı tarafından alındığında gönderilen ACK aldığında gönderdiği ACK' tır [22].

Birleştirme ve bölme

Datagram başlıklarındaki aşırı yüklenmeleri azaltmak için havadan gönderilen paketlerin sayısının azaltılması gerekmektedir. Paket sayısının azaltılması esnasında birçok protokol ve veri üniteleri tek bir datagrama yerleştirilebilir. Bu prosedüre birleştirme prosedürü denir. Tersine yapılan işlemede bölme prosedürü denir [22]. Birleştirme aynı göndericiden aynı alıcıya birçok paket yollandığında uygulanır. Birleştirme her zaman yapılabilir.

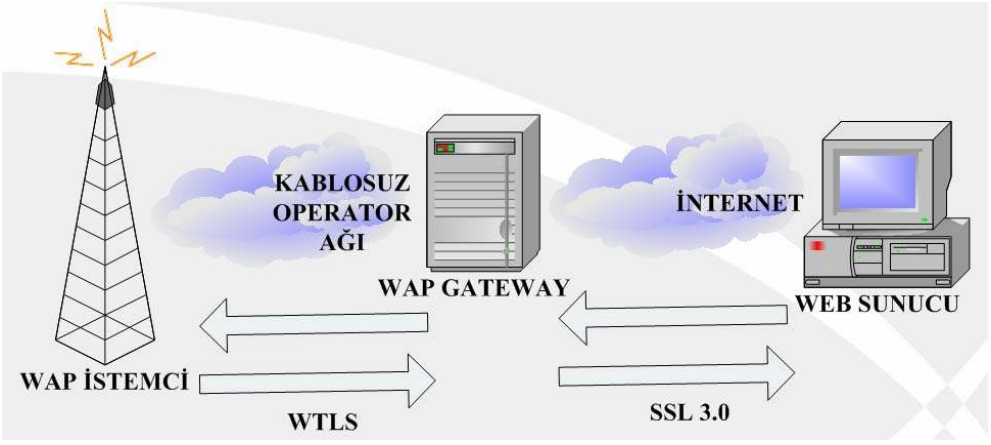
Bölme ve tekrardan toplama

Gönderilen paket taşıyıcının kapasitesinden daha büyükse paketler bölünür ve birçok paket halinde gönderilir. Buna bölme denir. Bu bölünmüş paketlerin tersine yapılan işleme tekrardan toplama denir [21].

2.1.4. WTLS

Mobil cihazların ve uygulamaların hayatın her alanına girmesi WAP protokolünün yaşamın her alanına girmesini sağlamıştır. WAP ağ geçidi, mobil telefonun limitli bant genişliğine ve işlem kapasitesine uygun olarak web dosyalarını, mobil telefona gönderir. Burada açığa çıkan güvenlik açıkları önemini arttırmıştır. Bu ortaya çıkan güvenlik açıklarından dolayı WAP forum bu protokolü sağlamıştır [3].

Kablosuz Ulaşım Katmanı, WAP'ın WDP katmanında çalışmaktadır. Kullanımı tamamen opsiyoneldir. Temel olarak bir oturum kurulur, veri transferleri yapılır ve oturum sonlandırılır. WTLS burada bu oturumun ve sistemin güvenli bir şekilde işlevini görmesini sağlamaktadır. HTTP-SSL veya TLS' nin WAP' taki karşılığıdır. TLS ve SSL Netscape tarafından oluşturulmuştur [22]. E-ticaret gibi olayların artması üzerine özellikle SSL çok gelişmiştir ve sunduğu hizmetler genişlemiştir. Dolayısıyla WTSL SSL 3.0 ile hemen hemen aynı güvenliği sunabilmektedir. WTLS Mobil kullanıcısının isteminden (port numarasından) bağımsız çalışmaktadır. WTLS WAP kullanıcısı ve ağ geçidinde birleştirilerek uygulanmalıdır [23]. Şekil 2.8'de WTLS ve SSL güvenlik ilişkisi gösterilmektedir.



Şekil 2.8. WTLS ve SSL güvenlik ilişkisi

Şekil 2.8'de gösterildiği gibi web sunucu ile WAP gateway arasında SSL 3.0 protokolü WAP gateway ile WAP istemci arasında da WTLS kullanılmaktadır.

Buradan da anlaşılacağı gibi web mimarisinde SSL'in gerçekleştirdiği işlevi WAP mimarisinde WTLS gerçekleştirmektedir.

Eğer kullanıcı kendine ait bir WAP sunucusu yok ise bir IIS veya mobil operatörü aracılığı ile, WAP ağ geçidine erişebilmekteyse, tüm bu trafik normal internet/intranet üzerinden sağlanıyor demektir. WTLS internet/intranet üzerinden gönderilen verileri koruyabilmek için büyük önem taşımaktadır. WTLS güvenliği şifreleme, şifreyi çözme, kullanıcı doğrulama ve veri bütünlüğünü kontrol ederek sağlamaktadır.

Kurala bağlı olmayan hizmetler

WTLS kurala bağlı olmayan hizmetleri güvenlikten emin olmak için kullanmakta [2] ve aşağıdaki işlevleri yerine getirmektedir.

- Güvenli bağlantıya başlanır
- Genel-Anahtar doğrulama veya istemciyle ile anahtar değişimi yapılır.
- El sıkışma(hand-shake) olunca bağlantıda değişiklik yapılır.
- Bağlantı bitirilir.
- Uyarı alarmları çalıştırılır.
- Sunucu tarafından istemciye yeni el sıkışma için istek gönderilir.

El sıkışma protokolü

Bu protokol istemci ve sunucu arasında güvenlik parametreleri üzerinde anlaşma sağlamaları için kullanılmaktadır.

- Rastgele değerler ve algoritmalar üzerinde anlaşma sağlanması için hello mesajları gönderilir.
- Gerekli olan şifreleme ile ilgili parametreler istemci ve sunucunun ilk ana güvenlik üzerinde anlaşmaları için birbirlerine gönderilir.

• İstemci ve sunucunun birbirlerinin kimlik doğrulamasını yapabilmek için sertifikalar ve şifreleme ile ilgili bilgiler gönderilir.
• İlk ana güvenlik ve değişilen rasgele değerlerden ana güvenlik üretilir.
• Kayıt katmanı için güvenlik parametreleri sağlanır.
• Böylece istemci ve sunucu güvenlikleri aynı şekilde hesapladıkları için birbirleriyle anlaşırlar. Bu sisteme bir saldırı yapılması kolay değildir [3].

Bu protokolde paketler birleştirilmiş olarak gönderilir. Paketlerin kaybolma veya hatalı olması önlenmesi amacıyla bu işlem yapılır.

WTLS' in kullandığı güvenlik sistemleri

WTLS' in kullandığı güvenlik sistemleri 4 çeşittir. Bunlar;

• *Cipherspec değiştirme (Privacy) protokolü:* Bu protokol her oturum açılımında başlangıç mesajlarından sonra devreye girer ve istemci ve sunucu arasında şifreleme programlarını çalıştırır. Bunun yanında eğer ki iki taraftan birinden şifre değiştirmekle ilgili bir istek gelirse şifre değiştirme işleminde bu protokol sayesinde yapılır. Bu sayede mobil cihaza alınan verilerin boyutunda önemli kazançlar sağlanmaktadır. Ayrıca sunucu ve istemci arasında olan veri transferlerinde şifrelenmiş içerik başkası tarafından okunamamaktadır ancak şifrelenmiş mesajlar görüntülenebilmektedir [3,23].
• *El sıkışma protokolü:* WAP Gateway, mobil cihaz ile iletişimde, bağlantısız tabanlı WDP' yi kullanırken, internet üzerinde bağlantı tabanlı olan HTTP tarafından kullanılan TCP'yi kullanır. TCP'de verinin doğru yere gittiği kontrolü yapılır, eğer gönderilen adrese gitmediyse paketin tekrar gönderilmesini sağlayan bir özellik vardır. Bu kısımda sunucu ve istemci kimlik doğrulama işlemleri gibi işlemler yapılmaktadır ve bu özelliklerin nasıl kullanılacağı hakkında iki tarafın antlaşması sağlanmaktadır [2].

• *İkaz protokolü:* Herhangi bir şekilde kapatılan veya bağlantısı kopan oturumun tekrardan aynı özellikler kullanılarak tekrardan açılmasını önlemek için bağlantı koptuğunda gönderilen mesajdır [3].

• *Veri bütünlüğü:* Sunucu ve istemci arasında yapılan veri transfer içeriğinin onların haberi olmadan kesinlikle değiştirilememesi işlemi burada yapılmaktadır. Bütün güvenlik işlemleri için önemli bir açık ise WTSL'den SSL'e değişim yapılırken WAP gateway üzerinde şifrelenmemiş içeriğin tutulmasıdır [23].

• *Güvenlik çözümleri:* WAP ile ortaya çıkan güvenlik açığını kapamak için ortaya atılan çözümlerin avantaj ve dezavantajları ile Çizelge 1'de özetlenmiştir [15].

Çizelge 2.1. Ağ geçidi güvenlik çözümleri [15]

Çözüm	Açıklama	Değerlendirme
	Oluşturulan güvenlik duvarı doğru bir yapılandırma ile yapılırsa oluşturulan sistem güvenli olacaktır. Ağ geçit aygıtı ve web sunucuyu aynı tarafa koymak gerekir. Önceden oluşturulmuş güvenlik duvarı ağ geçit aygıtında açık verebilmektedir ancak bu güvenliği azaltmamaktadır.	Ağlarda veri transferi artarsa istemciye verinin ulaşması gecikir. İstemci birden fazla siteye ulaşmak isteyebilir. Bunun için ağ geçit aygıtını kullanmak durumundadır. Bu da hizmet kalitesini düşürebilmektedir. WSP katmanının dezavantajlarından birisi ek hizmet servislerinin olmamasıdır. Karmaşık bir yazılıma sahip olan ağ geçidi yazılımının alınması ve bakımının yapılması gerekmektedir.
Uçtan uca güvenlik	Sunucu ve istemcinin veri iletişim protokollerine bağımlı olmayan	WML ve WML-Script kullanılan şifre fonksiyonları yeterince güçlü değildir.

Çizelge 2.1. (Devam) Ağ geçidi güvenlik çözümleri

Çözüm	Açıklama	Değerlendirme
	Ayrı bir güvenlik üzerinde anlaşması anlamına gelmektedir. Bu işlemlerin uygulaması yapılırken ağ geçidi kullanıldığı için bu aygıtın dezavantajları ile karşı karşıya kalınacaktır.	Güvenliği sağlayan firmanın bu güvenliği kendisinin ayarlıyor olması da ayrı bir problemdir. Burada WAP-Forum desteği yeterince yoktur ve standartlar belirli değildir.
WAP ağ geçidini iptal etme	İnternet üzerinde, uçtan uca şifrelemeyi kullanarak yapılan ticaretlerde elde edilen güvenlik seviyesi mobil ortamlarda da elde etmek mümkün olmaktadır.	Ağ geçidi aygıtının sağladığı çok büyük avantajlardan yararlanılamayacaktır. İleride çok güçlü mobil cihazlar üretilse bile iletişimdeki gecikme problemi aşılamayacaktır. Ağ geçit aygıtını iptal ederek WAP iletişimini yapmak oldukça zorlaşacak ve karmaşıklaşacaktır.

2.1.5. WDP

WDP kendi üzerinde bulunan ve çalışan WAP protokol yığını ile farklı ağlar arasına da taşıma hizmetlerini sunan birbirinden farklı taşıyıcı kullanabilen bir bağlantı noktası görevi üstlenmektedir. Bunları WAP ve diğer ağ'lar arasındaki bağlantı noktası olarak da tanımlayabiliriz. WDP üst katmanlara devamlı ve sürekli hizmet sunar. Bu hizmetler uygulamanın port numarasına göre adreslenmesi, opsiyonel olarak parçalara ayırma, tekrardan toplama ve opsiyonel olarak hata bulma özelliklerini içermektedir. Taşıyıcı hizmetleri, hedef ile sunucu arasındaki kablosuz veri linkidir. Örneğin SMS, bir taşıyıcı servis şeklidir. Her istemci cihazın, bir taşıyıcı servise sahip olması gerekir. Datagram ulaşım servisleri üzerinde çalışmaktadırlar. WAP için olan datagram transport WDP'de tanımlanmıştır. WDP ile taşıyıcı servislerin uyumu uygulamalara farklı taşıyıcı hizmetlerini kullanma

olanağı sağlar. Farklı taşıyıcı kullanmak en uygun performansı sağlamaktadır [3]. WDP bunun için yani farklı taşıyıcı kullanmak için adaptasyon katmanını kullanmaktadır. WDP verilerin DECT sistemi üzerinden taşınabilmesi için 3 farklı şekilde ayarlanabilir. Bunlar;

- DECT SMS
- Bağlantılı
- Bağlantısız

DECT SMS taşıyıcı hizmetleri ETSI standardına göre dökümantasyonu yapılmıştır. Radyo değiştiricisi 9dMMS olarak adlandırılan bir PC ile iletişim yapar. Bu PC radyo sistemi ile iç ağ arasında bir bağlantı görevi görmektedir. Veri sunucudan okunur ve WAP Gateway' e gönderilir. WAP Gateway' de bu bilgi WAP bit formu olarak kodlanır ve PC'ye gönderilir. PC'de bunu radyo santraline yollar [24].

WDP ve IP

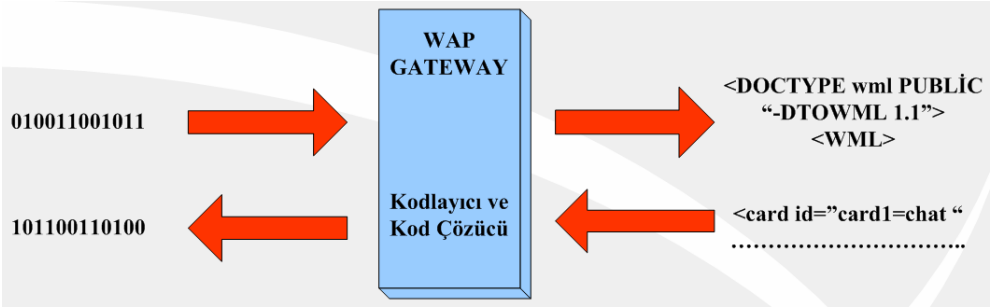
Ağlarda IP yönlendirme protokolü olarak kullanılır, UDP ise WDP' nin yerine uygulanır. IP bağlantısız datagram servislerinde parçalama ve tekrardan toplamaya olanak sağlar. UDP ise port tabanlı ve diğer gerekli fonksiyonlara olanak sağlar. Bu yüzden WDP katmanının ekstra bir fonksiyona ihtiyacı yoktur [2].

Kablosuz kontrol mesaj protokolü

WCMP WDP ve kablosuz veri gateway' leri tarafından datagram işlemleri yapılırken hata raporlaması için kullanılmaktadır. WCMP(Wireless Control Message Protocol-Kablosuz Mesaj Kontrol Protokolü) sadece IP taşıyıcılarının çalışmadığı çevrede çalışmaktadır. IP tabanlı ağlarda bunun yerine genelde ICMP mesajı kullanılmaktadır [3].

2.2. WML Yazılımı

WEB' de HTML kullanılırken WAP'ta WML kullanılmaktadır. WML HTML ile aynı işlevi görmektedir. WML ikili modda ve XML tabanlıdır. WML dar bantlı cihazları, küçük ekran, sınırlı kullanıcı girişi, dar bant ağ bağlantıları, sınırlı hafıza ve basit hesap yapabilen mobil cihazlar için tasarlanmıştır [25]. Şekil 2.9'da WML çevrimi gösterilmektedir.



Şekil 2.9.WML çevrimi

Şekil 2.9'da gösterildiği gibi WAP telefonlara ekranın küçük olması, işlemcilerinin çok güçlü olmamasından dolayı WAP gateway'de çevrim olması gerekmektedir. Yani gelen web sayfasının ikili moda çevrilmesi işlevi WML kodları ile yapmaktadır. Bu çevrim işlemleri ise WAP gateway'de yapılmaktadır.

WML XML'e dayandığı için her üretici firma kendine özgü WAP tarayıcı şekli üretebilmektedirler. Bu yüzden görünüm kullanıcının elinde değil, üreticinin isteğine bağlıdır [25]. Aşağıda bir WML örnek kodu gösterilmektedir.

WML örnek

```
<wml>
 <card>
   <p>
     Seçiniz:
                <select isim="S">
                        <option value="1">birinci</option>
                        <option value="2">ikinci</option>
```

```
                        <option value="3">üçüncü</option>
                        <option value="1;2">birinci ve ikinci</option>
                        <option value="2;3">ikinci ve üçüncü</option>
                        <option value="1 ; 2 ;3">birinci,ikinci ve üçüncü
                        </option>
                </select>
    </p>
  </card>
  <card>
        <p>
      İsminiz: <input type="text"
      isim="birinci"/><br/>
      Soyisminiz: <input type="text"
      isim="ikinci"/> <br/>
      MedeniHal: <input type="text"isim="cinsiyet"/><br/>
      İsminiz ve Soyisminiz: <input type="text"
      isim="birinci"/><br/>
      Soyisminiz ve MedeniHal: <input type="text"
      isim="ikinci"/><br/>
      İsminiz ve Soyisminiz ve MedeniHal: <input type="text"
      isim="cinsiyet" format="*N"/>

        </p>
  </card>
</wml>
```

`Option` komutu burada kullanıcıya seçim hakkı tanımaktadır. `Input` komutuyla değerlerin metin karşılığı olan sayısal değerler alınmaktadır[3]. Aşağıda ikinci bir WML örneği verilmiştir.

```
<wml>
  <card title="Input">
    <p>
      İşiniz: <input name="İşiniz" size="15"/><br/>
      Adres : <input name="Adres" size="15"format="*N"/><br/>
      Maaş  : <input name="Maaş" size="15"/>
    </p>
  </card>
</wml>
```

WML deck ve card'lardan oluşmaktadır. Card'lar toplanarak bir deck oluşturmaktadır. Burada WML'in card olarak yapılandırılmasının sebebi mobil cihazların ekranlarının küçük olmasından dolayı bütün bir sayfayı görüntüleyemiyor olmasından ve verinin hızlı bir şekilde indirilememesinden kaynaklanmaktadır.

Görüntü mobil cihaza card card gelmektedir. Bu sayede ekranda görünüm boyutu azaltılmakta ve ekranda içerik rahatça görülüp işlem yapılabilmektedir. Bunun yanında WML tamamen text'lerden oluşmaktadır. Text olmasının sebebi ise mobil cihazların ekran genişliği, dar bant olması, sınırlı güç tüketimi gibi birçok nedenden dolayı mobil cihazların veriyi hızlı bir şekilde alıp görebilmesi içindir [25]. Şekil 2.10'de WAP istemci ve WAP sunucu arasındaki WML ilişkisi gösterilmiştir.

Şekil 2.10.WAP istemci ve WAP sunucu arasındaki WML ilişkisi

Şekil 2.10'de WAP istemci WAP sunucudan WML içerikli veya WML script içerikli bir dosya isteğinde bulunduğunda gateway üzerinde meydana gelecek işlem gösterilmektedir. WAP istemci WAP gateway üzerinden talebini WAP sunucuya yollar, WAP sunucu cevabı hazırlar ve yine WAP gateway üzerinden cevabı geri gönderir [2].

Kullanılan bazı önemli kütüphaneler

CGI-WML Kütüphanesi: WML sayfalarını daha hızlı oluşturmak için ihtiyaç vardır. WML başlıkları farklı fonksiyonlar için belirlenmiş tag'ları içerir (tablolar, bağlantılar gibi). Text formatı ve URL değer kontrolü CGI-WML yapılandırmasında bazı zararlara neden olabilmektedirler [1].

Dosyalar; cgiwml_lib.c, cgiwml_lib.h içermektedir.
Bunların kazandırdığı özellikler ise;
Newdeck, Enddeck, Endtags, Newcard, endcard, WML text string, Newline …vb.

Dosyalar; Cardlib.c, Cardlib.h içermektedir. Bunların kazandırdığı özellikler ise; İnput, input variables, option value, option in menu, select menu, menu, end select, end tags….vb.

Dosyalar; Nav_task.c
WML_a(quick anchor list), WML_anchor, End_anchor, go, postfield(değer gönderir), end_go…vb,

Bu kodlamalar yapılırken kullanıcının C bilgisine ihtiyacı vardır. Her tag mutlaka bir end tag içermelidir [3].

2.2.1. WML script

WML script java script'ten türetilmiştir. Ancak WML script düşük bant genişliği iletişimi ve düşük CPU'ya sahip olan cihazların iletişimi için değişiklik yapılarak elde edilmiştir [25]. Aşağıda WML script'e bir örnek verilmiştir.

```
function hesapal(fatura)
{
 if(fatura>200)
 {var yeni=fatura;}
 else
 {yeni=200;}
 return yeni;
}
```

2.3. TCP'de Tıkanıklık

Jacobson&Karels 1988 yılında(Congestion Avoidance And Control-Tıkanıklıktan kaçınma ve kontrol) isminde bir fikir ortaya attılar. Bu fikirde, TCP'de yer alan yedi algoritmadan beşini tanımladılar [26].

TCP bağlantısında, durum ‘paketlerin korunumu yasası’na uygun olmalıdır. Yani, paket akışının bir eşitlik içerisinde olması durumuna denmektedir. Bir başka deyişle, dolu olan bir yere eski paketin gönderilmeden yeni paketin alınmaması olayıdır. Burada başarısızlık için 3 neden vardır;

- Bağlantı, korunum ilkesini yerine getirmekte yetersiz kalır.
- Gönderici eski paket için Bir ACK beklemeden yenisini yollar.
- Yol yetersizliği yüzünden bazı paketler daha fazla geçecek ve hattı devamlı dolu tutacaklardır.

Bu durumlarda belirli bir zamanlama uygulanmaktadır. O süre bitene kadar cevap ACK’ının gelmemesi veya gelmesi beklenmeden tekrar gönderimin yapılması tıkanıklık meydana getirecektir [27]. Zamanlama işlemi için yavaş başlama(slow-start) algoritması uygulanmıştır. Slow-start algoritması;

- Her bağlantı için ayrı ayrı pencere eklenmektedir.
- Herhangi bir kayıp meydana geldiği zaman cwnd(pencere) bundan haberdar edilir ve yeni duruma göre ayarlanır. Gönderilmeyen paket cwnd de tutulur gönderilenlerin yerine yenisi alınır.
- Eğer alındı ACK’ı gelirse cwnd bir paket arttırılır ve başarılı olan gönderme işleminde gönderilen paket silinir.

Başarısızlıktaki bir başka yaklaşım ise gidiş dönüş zamanı, yani zamanlamanın paketlerin düzgün ve eşit şekilde gönderilmesi için gerekeli olmasıdır. Zamanlama doğru olmazsa tekrardan gönderimlerden dolayı trafik artacak ve gelen paketlerin tutulduğu kuyruk solacaktır. Buda tıkanıklık meydana getirecektir [28]. TCP ortalama gidiş dönüş zamanını hesaplamak için düşük-geçiş(low-pass) filter kullanmaktadır.

- $SRTT = \alpha * SRTT + (1-\alpha) * RTT$

1. SRTT : Hesaplanmış ortalama RTT’dir.

2. α : filtre kazancıdır. 0.9 olarak kabul edilmiş ve sabittir.
İkinci paketin gönderilmesi olayında β ortaya çıkar.

- β * SRTT

1.β : RTT değişimidir. Kullanılan değer ise genelde 2'dir.

Ağda oluşacak paket trafiği için yapılacak hesaplamalar yani zamanlama doğru yapılmazsa tekrardan gönderim olayı devamlı gerçekleşeceğinden ağda birden fazla aynı paketin dolaşması meydana gelecek ve aşırı bir trafik oluşacak ve tıkanıklık meydana gelecektir. Van Jacobs&Karels'in geliştirdiği yöntemde β değerinin sabit olarak değil, dinamik olarak ayarlanması sayesinde zamanlamanın daha doğru yapılacağı ve bu sayede tıkanıklığın çözüleceği savunulmuştur [16]. Ancak, bu sistemde;

- Hesaplamak için daha fazla CPU kaynağı gerekir.
- Değişim kestiriminde, standart sapmadan daha kötü sonuç vermektedir.

Eğer hata tahmini normal dağılıma uygun ise ortalama sapma ile standart sapma arasında genellikle ilişki vardır.

OrtalamaSapma = sqrt(π)/2 * StandartSapma ~ 1.25 * StandartSapma

Standart sapmanın bir değeri olarak ortalama sapmayı hesaplamak CPU açısından daha kolaydır. Bir başka başarısızlık nedeni ise zamanlama iyi yapılırsa ve bu durumda bile paket kaybolursa bu büyük ihtimalle tıkanıklık nedeniyle meydana gelmiş demektir [26]. Bu durumda;

- Ağda tıkanıklık olursa veya tıkanıklık olma ihtimali ile karşılaşıldığında bu durumlar uç birimlere sinyal yoluyla bildirilmelidir.

• Uç birimler bu sinyali aldıkları zaman ağın performansını düşürüp tıkanıklığı engelleyebilecek ve sinyal olmadığında ise ağın performansını arttırabilecek kapasiteye sahip olmalıdırlar.

Tıkanıklık meydana geldiğinde bu yoğun network trafiğinden oluşacağından dolayı bundan kaçınmak için cwnd'yi yeni koşullara göre ayarlamak gerekmektedir [27].

2.4. WAP'ta Tıkanıklık

WAP Gateway, WAP forum tarafından çalışan ve verimli kablosuz internet erişimini sağlamak için sunulmuş ve geliştirilmiştir. Bu cihazdaki amaç protokol çevrimi ve içerik değişimi yapılarak kablosuz ağda birim zamanda veri transfer miktarını azaltmaktadır. Kablosuz veri hizmeti için talebin büyümesi WAP gateway'lerin performansı, ölçeklenmesi ve uygunluğu konusunda büyük bir gereksinim ortaya koymuştur. Bu sayede WAP için kullanılan özelliklerin her geçen gün daha çok artması sağlanmıştır [6].

Tıkanıklık bir ağın performansını etkileyen en önemli unsurlarda birisidir. Herhangi bir ağda aşırı yükten kaynaklanan paket kayıpları olursa veya paketlerin iletimi sırasında gecikmeler artarsa bu ağda tıkanıklık oluşmuş demektir [13].

Protokol çevrim ve içerik değişim işlemlerinden dolayı ağda yapılan hesaplama yoğunlaşmış ve tek parça gateway'lerin aşırı yüklenmeye başlamış olduğu görülmüş ve bu işlemlerin tek parça gateway ile çözülemeyeceği anlaşılmıştır. Bu büyüyen isteklerin üstesinden gelmenin çalışan bir yol üzerinde ekstra donanım kaynakları yerleştirilerek ve bunları bir küme mimarisi içinde oluşturularak aşılması sağlanmaya çalışılmıştır [29]. WAP gateway'leri daha hızlı modelleriyle değiştirmek yüksek maliyet getirmekte ve talepleri tam olarak karşılamamaktadır. Tek WAP gateway'den kümelenmiş gateway'lere dönmek, maliyet açısından ölçeklenebilir, güvenli ve yüksek performans olanağı sağlamış ve yapılan işlemler için çok kullanışlı olmuştur. Fakat küme mimarisi içerisinde istekleri alıp,

yorumlayacak ve yönlendirecek gateway'i seçmek için çalışabilir ve verimli bir yük dengeleme mekanizmasına ihtiyaç duyulmaktadır [30].

WAP modelde WAP istemcilerin internete erişimlerinden önce isteklerinin anlaşılabilir hale getirilebilmesi ve cevapların görüntülenebilmesi için WAP gateway ile bir bağlantı kurmaları gerekmektedir. WAP gateway bu bağlantıyı iptal edene kadar bu bağlantı sürdürülmektedir. Bu bağlantı zamanı içinde istemci WAP gateway'den sınırsız istek ve talepte bulunabilir. Bu bilinmeyen bağlantı zamanı ve değişik mobil istemcilerden gelen yüklemeler ve isteklerin işleminin yapılabilmesi için küme mimarisi içindeki gateway'lerin yük dengeleme mekanizması iyi konfigüre edilmelidir. Aksi takdirde bu özellik küme içerisindeki bazı gateway'lerin aşırı yüklenmelerini, bazılarının da hafif yüklenmelerini sağlamaktadır [31]. WAP standardı görev atama kararını istemci tarafından birden fazla konfigure edilmiş gateway'ler arasından rasgele seçilerek yapılmasını önermektedir veya gateway'lerin bulunduğu tarafta WSP yeniden yönlendirme mekanizması ile küme içerisinde aşırı yüklenen gateway'lerin yeni yönlendirme ile gelen bağlantı isteğini başka aşırı yüklenmemiş gateway'e yönlendirmesini önermektedir. Bu iki mekanizmanın bazı dezavantajları bulunmaktadır [31]. İstemci tarafının yaklaşımında ölçeklenme yoktur ve aşırı yüklenme bir gateway üzerinde olabilir, eğer ki birçok istemci aynı gateway'i kullanırsa WSP tekrar yönlendirme mekanizması ile yeni seçilen gateway'in adresinin geri dönmesinde ve yeni bağlantının başarılı olmasında başarılı olmaktadır ve ölçeklenmeyi ve uygunluğu ayarlamaktadır. Fakat bu kablosuz ağ trafiğini ve yanıt süresini arttırmaktadır. Çünkü devamlı mesaj değiş tokuşu yapılmaktadır [31].

Küme mimarisi gerçek gateway'lerden oluşmaktadır. Bu küme mimarisi içerisinde genelde yük dengeleyicisi ve mobil istemci isteklerini gerçek gateway'lere yönlendiren front-end-ditributer diye adlandırılan bir mekanizma vardır. Bu mekanizma bir gateway üzerinde bulunmaktadır. Front-end-distributer şeffaftır ve WAP istemciye tek kullanıcı arayüzü sağlamak için kullanılmaktadır [6].

Kannel gateway'ler bu mimarinin temelini oluşturmaktadırlar. Bu mimari gelen talepler için ölçeklenmeyi ve gerçek gateway'ler için yük dengeleme işlemini sağlamaktadır. Fakat bu mimarinin ana problemi front-end-distributer mekanizmasını sunan WAP gateway'in aşırı istek yüklenmesinde tıkanmasıdır [29]. Aşırı istek durumlarında WAP'ta tıkanıklığı çözmek için kullanılan birçok metot vardır. Bu metotlardan önemli olan bazıları şu şekilde adlandırılmaktadır;

- LVS
- KG
- SWG

2.4.1. LVS

LVS gerçek sunucuların oluşturduğu kümede kurulumdan itibaren olan ve yüksek derecede ölçeklenebilir özelliklerine sahip olan bir sunucudur. Aynı zamanda bu sistemde yük dengeleyicisi Linux işletim sisteminde çalışmaktadır. Küme mimarisi son kullanıcı için transparandır [6]. Son kullanıcı sadece tek bir sanal sunucu görmektedir. Gerçek sunucu küme içerisindeki herhangi bir sunucu olabilir örneğin; WAP gateway veya http sunucu gibi. Gerçek sunucular birbirine yüksek hızlı LAN'la veya coğrafik olarak dağıtılan WAN'la bağlı olabilmektedirler. Gerçek sunucuların front-end-distributer'u yük dengeleyicisidir ve Linux Director olarak adlandırılmaktadır. Linux Director gelen istekleri değişik sunuculara dağıtır ve kümeyi tek bir sanal service olarak görünmesi için kümede paralel hizmet mantığını kullanır. Burada ayrıca tek IP adresi kullanılmaktadır [31].

Ölçeklenme ise kümedeki gerçek sunucuların açık, net, anlaşılır olarak eklenip silinmesiyle başarıya ulaşır. LVS WRR'ı kullanır ve WLC(Weighted Least Connection) programlama algoritmasını oturum bağlantılarını gerçek sunucular arasında pay etmek için kullanmaktadır [31].

2.4.2. Kannel gateway

WAP gateway'lerde yapıldığı tarihten itibaren var olan yük dengeleme yeteneğinin açık kod olması bu tip çözümlerin genel özelliğidir. Kümeleme mimarisinin temelini oluşturan Kannel Gateway'ler 2 bileşen içermektedirler [5]. Bunlar;

Taşıyıcı box

Taşıyıcı box bir front-end distribütör'dür ve bu özellik WAP istemcilere birleşmiş bir ara yüz sunmaktadır. Bu bileşen WAP protokol stağında WAP katmanları için uygulanır [5,31].

WAP box

WAP box daha yüksek katmanlarda çalışmaktadır. Örneğin; WTP ve WSP gibi. Taşıyıcı box WAP isteklerini tekrardan yazar ve bunu WAP box'lar içinde ki seçilmiş WAP box'a TCP yolu ile gönderir. Taşıyıcı box rasgele programlama algoritmasını WAP boxes'lara olan bağlantı oturumlarını birleştirmek için kullanır. Kannel gateway'lerin kötü tarafı, yeniden yazılma işleminin uygulama katmanında olmasıdır. Buradaki problem ise bütün istek paketleri ve yanıt paketlerinin taşıyıcı box tarafından yeniden yazılmaya gereksinim duymasıdır [5,31].

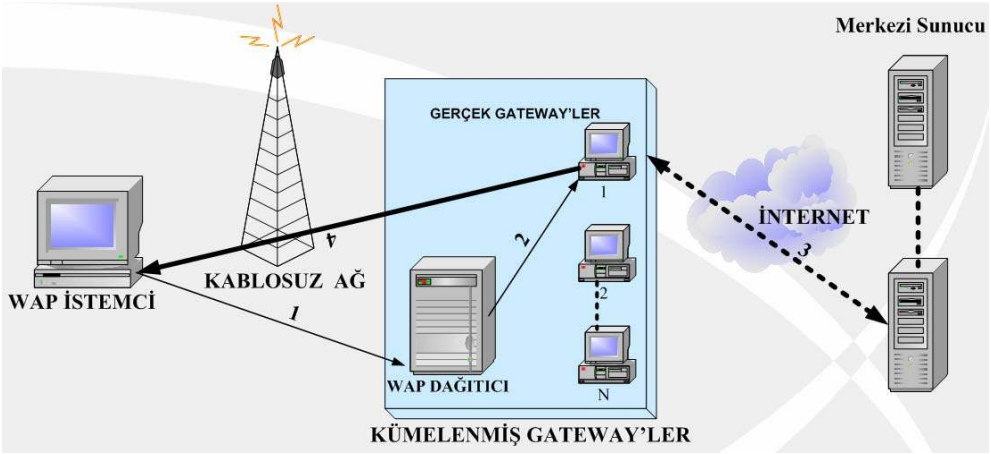
2.4.3. SWG

SWG(Scalable WAP Gateway-Ölçeklenebilir WAP Gateway) bir grup gerçek gateway içermekte ve bunlar birbirine hızlı bir ağda bağlı bulunmaktadırlar. Bu kannel-level front-end distributör'ü bütünleştirmekte ve WAP dispatcher olarak adlandırılan yeni bir mekanizma sunmaktadır. Kannel gateway'de ki dağıtıcı uygulama levelında çalışmakta, gelen istekleri modifiye etmekte ve oluşturulan cevapları talep eden mobil cihaza göndermektedir. WAP dispatcher'ın tıkanmasını önlemek için WAP dispatcher'a gelen talepler kannel alanındaki gerçek gateway'lere yönlendirilmekte ve WAP dispatcher gateway ve client arasındaki

istek-cevap işine karışmamaktadır[31]. WAP Dispatcher 'lar veri miktarı ufak olan istemci isteklerini karakterize ederler ve bu sayede büyük, geniş veri miktarına sahip yanıtlar oluşturabilirler. WAP Dispatcher WRR (weighted round robin) program algoritması kullanmaktadır [31]. Bu algoritma yük dengeleme mekanizması tarafından desteklenen ağırlıkları ve parametreleriyle küme içinde ki gerçek gateway'ler için bağlantıları ayırmak, tahsis etmek için kullanılmaktadır. Ölçeklenebilir WAP gateway'nin sahip olduğu bazı özellikler;

- Potansiyel yüklenmeyi küçük hesaplama zamanı ve iletişim ek yüklemesi olmadan hesaplanması.
- Geri bildirim alarm mekanizmasının gerçek gateway'lerin kullanımı sırasında kritik başlangıç değerini aşması halinde eşzamanlı alarm göndermesi.
- WAP-awareness, WAP istekleri için temel olan yük dengeleme mekanizmasının performansında kullanışlı olması.

Bu metotta ölçeklenebilir WAP gateway'in artan WAP hizmet isteğinde ki ölçeklenebilirliği yük dengeleme yeteneğiyle birleştirilmesi amaçlanmıştır. SWG mimarisi Şekil 2.11.'de gösterilmiştir [31].



Şekil 2.11. Ölçeklenebilir WAP gateway mimarisi

Şekil 2.11'de görüldüğü gibi Garçek gatewayler kümelenmiş ve onların önünde WAP istemciden gelecek bütün talepleri ilk önce karşılayacak ve küme içerisinde

bulunan diğer gerçek gateway'lere eşit şekilde yük dağılımı yapması sağlayacak bir WAP dağıtıcı bulunmaktadır. Gerçek gateway'ler n kadar olabilir.

3. GELİŞTİRİLEN TIKANIKLIK PROTOKOLÜ

Geliştirilen protokol ns-2 benzetim aracı kullanılarak test edilmiştir. Bu bölümde önce ilk önce ns-2 tanıtılacak daha sonra ns-2 kullanılarak geliştirilen tıkanıklık protokolü anlatılacaktır.

3.1. Benzetim Ortamı

WAP' ta geliştirilen tıkanıklık denetiminin performansını görmek için ns-2 2.0 sürümü kullanılmıştır. ns-2 seçiminin öncelikli nedeni çoğu protokolün tanımlı olmasıdır. Diğer nedeni ise akademik çalışmaların çok büyük bir bölümünün ns-2 ile yapılmış olmasıdır.

ns-2 Berkley Üniversitesi tarafından geliştirilmiştir. ns-2 bir kesikli olay benzetim aracıdır. Hem kablosuz ağlar hem de kablolu ağlar için destek vermekte ve birçok LAN altyapısının geliştirilmesine izin vermektedir. Yazılım dili olarak OTCL/Tk kullanılır ve C++ dilinde yazılmıştır. Gerçek ağda bulunan protokolleri ve bunların uygulanabileceği ortamların kullanılmasına izin vermekte ve bu sayede oluşturulan link, node..vb. objenin çalıştırılabileceği bir OTCL scripti yazılabilmektedir. Daha önceden bu benzetim ortamına entegre edilmiş algoritmalar ile farklı ve yeni bir algoritmanın sisteme entegresi için kullanılan yazılım dili C++'tır. ns-2'de bir benzetim çalıştırıldığında üzerinde meydana gelen her olayı çalıştırılan benzetim sonlandırılıncaya kadar iz formatında rapor vermektedir. Buradan elde edilecek veriler daha sonra gerekli ve istenilen istatistiksel bilgileri elde etmek için kullanılmaktadırlar.

Kullanılabilecek düğüm sayısı, veri trafiğinin ne kadar olması, sınırları, paketlerinin boyutu ve bağlantıların kapasitesi gibi durumlar bu kısımda belirlenmektedir. Temel kuyruk desenleri değiştirilerek bu kısımda farklı senaryolar elde edilmektedir. Bu değişiklikler sayesinde düğümlerin kuyruk yapısında değişiklikler yapılarak farklı tıkanıklık denetimleri oluşturulabilmektedir.

3.2. Geliştirilen Protokol

Geliştirilen protokol anlatılmadan önce geliştirilen protokol ile karşılaştırması yapılan drop-tail kuyruk tipi ve red kuyruk tipi açıklanacaktır. Drop-tail kuyruk tipinde FIFO(first in first out-ilk giren ilk çıkar) mantığı uygulanmaktadır. Belirlenen kuyruk doluluk oranının eşik değeri geçildiğinde gelen paketlerin tipine bakılmadan atılmaya başlanmaktadır. Günümüzdeki yönlendirici cihazların geneli bu kuyruk tipini uygulamaktadırlar. ns-2'de bulunan drop-tail kuyruk tipinin class ve OTCL kodu aşağıda verilmiştir.

```
class DropTail : public Queue
{
 protected :
 void enque(Packet*);
 Packet * deque();
 PacketQueue q_;
};
```

Drop-tail kuyruk tipinin türetildiği temel sınıf olan Queue sınıfı en çok kullanılan değişken ve fonksiyonları kendinden isteyen kuyruk yapılarına göndermektedir. Drop-tail kuyruk yapısında bulunan `enque` ve `deque` kodları şu şekildedir.

```
void DropTail :: enque(Packet* p)
{
 q_.enque(p) ;
 if (q_.length() >=qlim_)
 {
  q_.remove(p);
  drop(p);
 }
}
Packet* DropTail :: deque()
{
 Return(q_.deque());
}
```

Burada `enque` fonksiyonu gelen paketi ilk önce iç paket kuyruğunda(kuyruk limit sınırlaması yok) tutmaktadır ve `qlim_` üzerinden kuyruk limitini kontrol etmektedir.

Aşırı yüklenme olduğu durumlarda en son iç paket kuyruğuna alınan paketten atılma olayı gerçekleşmektedir.

Red kuyruk tipi ise aktif bir kuyruk yönetim mekanizması sunmaktadır. Red kuyruk tipinde iletişim sırasında oluşabilecek bir tıkanıklığın önceden belirlenmesi ve buna göre paketlerin atımı sağlanmaktadır[32]. Red kuyruk tipi kuyruk limiti dolmadan gelen paketleri rastgele atmaktadır. Kuyrukta gecikme süresi drop tail'e göre daha azdır. Eşit paket atımı amaçlanmıştır. Red kuyruk tipinde parametreleri konfigure etmek zordur.

Red kuyruk tipi drop-tail kuyruk tipinden performans açısından çok iyi değildir. UDP trafiğinde drop-tail red kuyruk tipinden daha iyidir. Küçük buffer'larda red kuyruk tipinde kullanılan parametrelerin performansa etkisi çok azdır. Büyük buffer'larda red kuyruk tipi performansı arttırabilmektedir ancak parametreleri iyi ayarlanmalıdır. Ancak küçük buffer'lar end2end performansında daha hassastırlar.

Geliştirilen protokol drop-tail kuyruk tipi temel alınarak yapılmıştır. Burada 2 senaryo oluşturulmuş ve sonuçlar diğer kuyruk tipleriyle karşılaştırılmıştır.

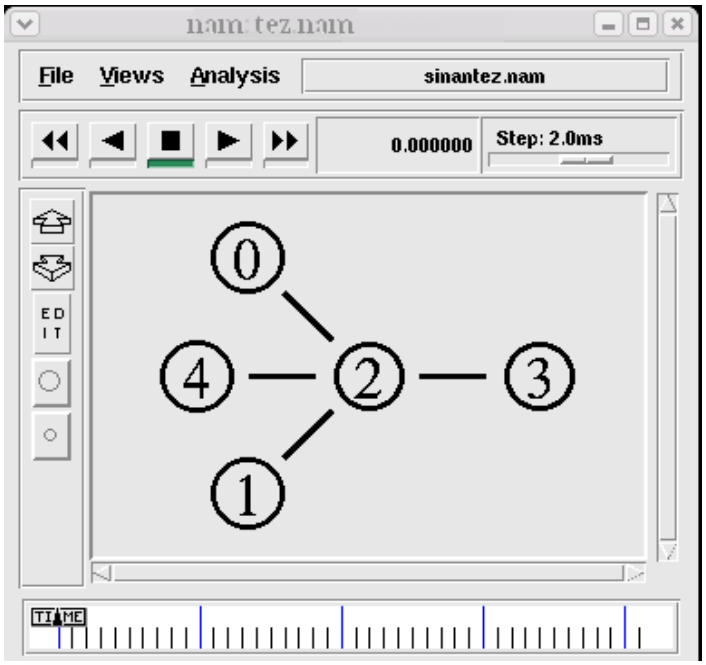
İlk Senaryo: Bu çalışmada benzetim ortamı aracı olan ns-2'de Otcl/tk yazılım dili kullanılarak 5 tane düğüm oluşturulmakta ve hepsi birbirine duplex-link'ler ile bağlanmaktadır. Her bir bağlantının bant genişliği ve kullanacağı kuyruk tipide belirlenmektedir.

```
Set ns [new simulator]
Set n0 [$ns node]
Set n1 [$ns node]
Set n2 [$ns node]
Set n3 [$ns node]
$ns duplex-link $no $n2 2Mb 2ms DropTail
$ns duplex-link $n1 $n2 2Mb 2ms DropTail
$ns duplex-link $n2 $n3 2Mb 2ms DropTail
```

Düğümler ve aralarındaki bağlantılar oluşturulduktan sonra benzetim çıktısının düzgün çıkması için düğümlerin yerleştirilmesi kodla yapılmıştır. Aşağıdaki kod sayesinde düğümler belirlenen yerlerde sabit şekilde elde edilmektedir.

```
$ns duplex-link-op $n0 $n2 orient right-up
$ns duplex-link-op $n1 $n2 orient right-down
$ns duplex-link-op $n2 $n3 orient right
```

Bu işlemler yapıldıktan sonra ns-2 benzetim aracında derleme yapılmış ve çıktı formatı hazırlanmıştır. Kod derlenip çalıştırıldıktan sonra Şekil 3.1 görünümü elde edilmiştir.



Şekil 3.1. Oluşturulan düğümlerin benzetim ekranı

Burada n0, n1 ve n4 istemci görevi, n2 WAP gateway görevi n3 ise WAP sunucu görevi görmektedir. Bu benzetimde n0, n1 ve n4 ile n2 arasındaki bağlantılarda drop-tail, red ve geliştirilen kuyruk tipleri ayrı ayrı çalıştırılmış ve sonuçlar elde edilmiştir.

Yapılan çalışma için hazırlanan ilk senaryo üzerinden 3 kuyruk tipinin ayrı ayrı olarak benzetim sonuçlarının elde edilmesiyle oluşmuştur. Bu senaryo tez.tcl

dosyasında yazılmış ve benzetim ortamında çalıştırılarak sonuçlar elde edilmiştir. tez.nam çıktısının elde edilmesi için tez.tcl dosyasına aşağıdaki kod eklenmiştir.

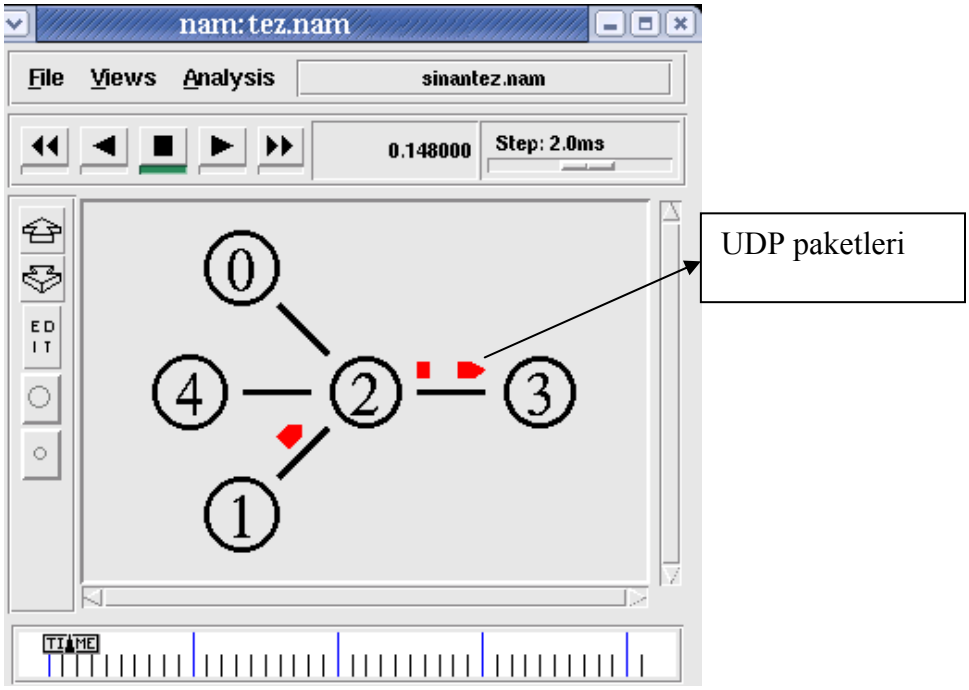
```
[root@localhost bin]# ./ns tez.tcl
ns: finish: couldn't execute "nam": no such file or directory
while executing
"exec nam tez.nam &"
  (procedure "finish" line 7)
  invoked from within
"finish"
```

Başlangıç olarak n1 düğüm yapısı UDP paket yollayacak şekilde, n0 ve n4 düğüm yapısı da TCP paket yollayacak şekilde aşağıdaki kodlar yazılarak hazırlanmıştır. TCP paket yapısı FTP olarak ayarlanmıştır.

```
#TCP bağlantısının oluşturulması
set tcp [new Agent/TCP]
$tcp set class_ 2
$ns attach-agent $n0 $tcp
set sink [new Agent/TCPSink]
$ns attach-agent $n3 $sink
$ns connect $tcp $sink
$tcp set fid_ 1

#TCP bağlantısı üzerinde FTP'nin oluşturulması
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ftp set type_ FTP

#UDP bağlantısının oluşturulması
set udp [new Agent/UDP]
$ns attach-agent $n1 $udp
set null [new Agent/Null]
$ns attach-agent $n3 $null
$ns connect $udp $null
$udp set fid_ 2

#UDP bağlantısı üzerinde CBR'nin oluşturulması
set cbr [new Application/Traffic/CBR]
$cbr attach-agent $udp
$cbr set type_ CBR
$cbr set packet_size_ 1000
$cbr set rate_ 1mb
```

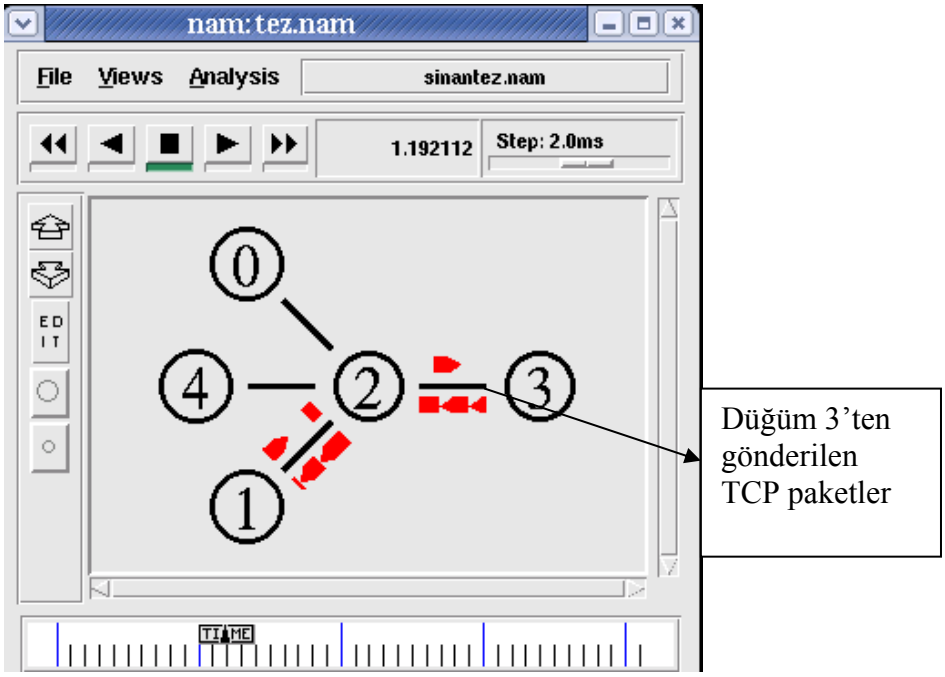
$cbr set random_ false

Bu kodlar yazıldıktan sonra ve hangi düğümlerin hangi bağlantıyı kullanacağı belirlendikten sonra ilk paket gönderim sonucu benzetim aracı da Şekil 3.2'deki gibi alınmıştır. 0,1ms zamanında UDP paket transferi başlamaktadır. TCP paket transferi ise 1,0 ms zamanında başlamaktadır.



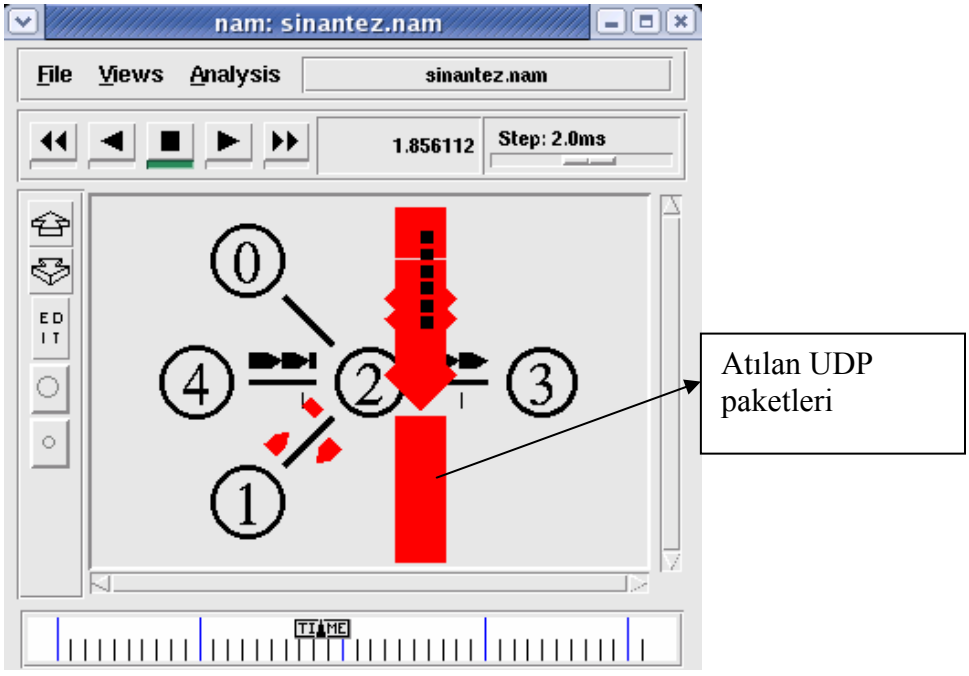
Şekil 3.2. Benzetimde ilk gönderilen paketler

n1 düğümünden n3 düğümüne gelen istekler n3 düğümü tarafından cevaplanmaktadır. n3 düğümü gelen isteğe göre TCP veya UDP paketleri hazırlamakta ve istemciye göndermektedir. Şekil 3.3'te görüldüğü gibi düğüm 1 düğüm 3'e UDP paketleri yollamakta ve düğüm 3'te düğüm 1'e UDP paketleri yollamaktadır.



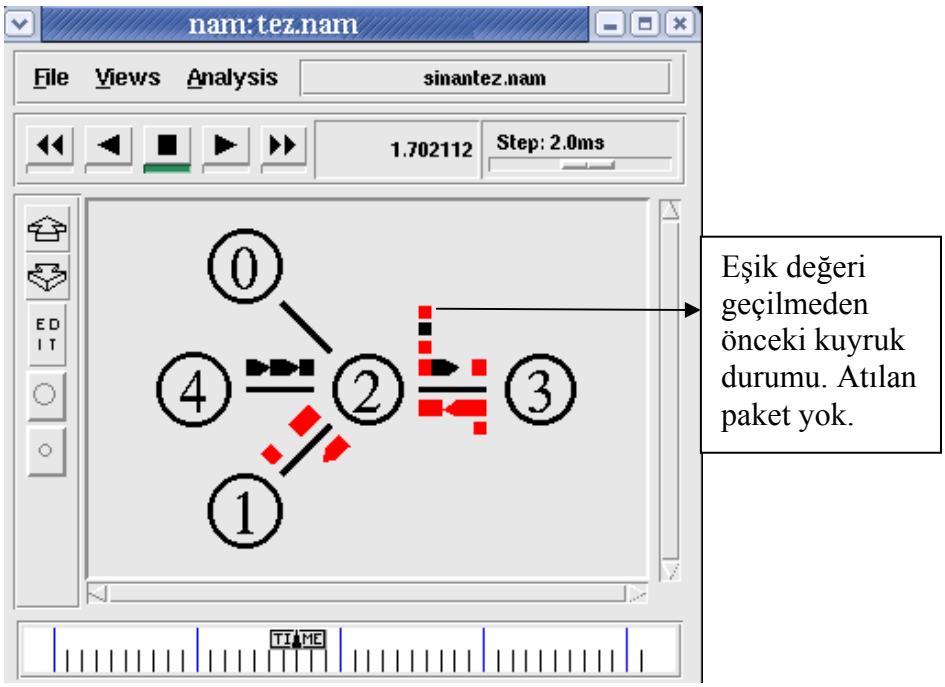
Şekil 3.3. Düğüm 3'e gelen ve giden paket durumu

Şekil 3.4'te düğüm 2'deki düğümünde kullanılan kuyruk yapısının şekli gösterilmektedir.



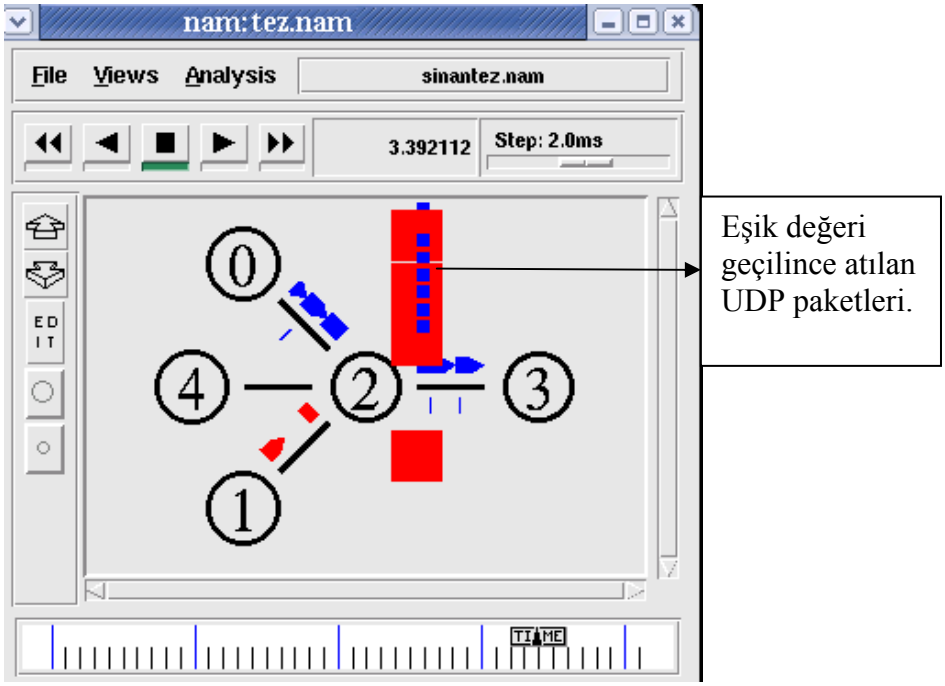
Şekil 3.4. Düğüm 2'de atılan paket ekranı

Kuyruk yapısı gelen paketi önceliğine göre kuyruğa sokmaktadır. Burada öncelik istenilen sayfaya daha önceden verilmiş öncelik numarasıdır. Bu öncelik numarasına göre gelen paketlerin akış numaraları kontrol edilmekte ve önceliğine göre kuyruğa sokulmaktadır. Eğer ki kuyruk boş ise gelen paketlerin önceliklerine bakılmadan kuyruğa sokulmakta ve normal kuyruk işlevi olan ilk giren ilk çıkar mantığıyla işlem yapılmaktadır. Şekil 3.5'te kuyruğun %80 doluluk oranını geçmediği durum gösterilmektedir.



Şekil 3.5. Kuyruğun dolu olmadığı durum

Eğer hat veya ağdaki trafik yoğunsa ve gelen paketlerden dolayı kuyruk doluluk oranının %80'ı dolmuş ise bu durumda öncelik ön plana çıkmakta ve öncelikli olan paket geldiğinde direk kuyruğa sokulmaktadır. Önceliği az olan bir paket geldiğinde kuyruğa alınmadan atılmakta ve böylece öncelikli olan isteğin tıkanıklığa uğraması önlenmektedir. Eğer kuyruk doluluk oranı %80'in altına düşerse bu durumda gelen her paket kuyruğa sokulmakta ve buna göre işlem yapılmaktadır. Kuyruğa alınmayan ve atılan UDP paketleri Şekil 3.4 ve Şekil 3.6'da gösterilmektedir.



Şekil 3.6. Öncelikli paket ve kuyruğun dolu olma durumu

Şekil 3.6' da verilen ekran çıktısında ise 3 s zamanında başlayan ikinci TCP bağlantısının kuyruğa girerken UDP bağlantısına göre öncelikli olmasından dolayı bu zaman aralığında da UDP' den gelen paketler kuyruk tarafından atılmaktadır.

Kuyruk yapısı drop-tail'de yapılan değişiklikler ise paket giriş fonksiyonu olan enque() fonksiyonunda yapılmıştır. Bu fonksiyonda benzetim ortamının sunduğu kütüphanelerden faydalanılarak paketlerin akış numaraları ve kuyruk limiti belirlenmiştir. Eğer kuyruk doluluk oranı %80' in altında ise gelen paketin akış numarasına veya herhangi başka bir özelliğine bakılmadan direk kuyruğa girmesi ve işleminin yapılması sağlanmaktadır. Bu kısımda ilk giren ilk çıkar mantığı uygulanmıştır. Ancak, kuyruk doluluk oranı %80 in üstüne çıktığı anda direkt olarak akış numaraları ve bağlantı tipleri TCP veya UDP değerlendirilmekte ve kuyruğa alınma işlemi bu değerlerin öncelik sırasına göre yapılmaktadır. Burada bağlantının TCP veya UDP olduğu akış numaralarına göre belirlenmektedir. Akış numaraları TCP olan bağlantılar kuyruk doluluk oranı %80 'in altına düşene kadar öncelikli hakka sahip olmaktadırlar. Bağlantı tipi TCP'den farklı olan diğer paketler bu oran eşik değerinin altına düşene kadar kuyruk tarafından kabul edilmemekte ve

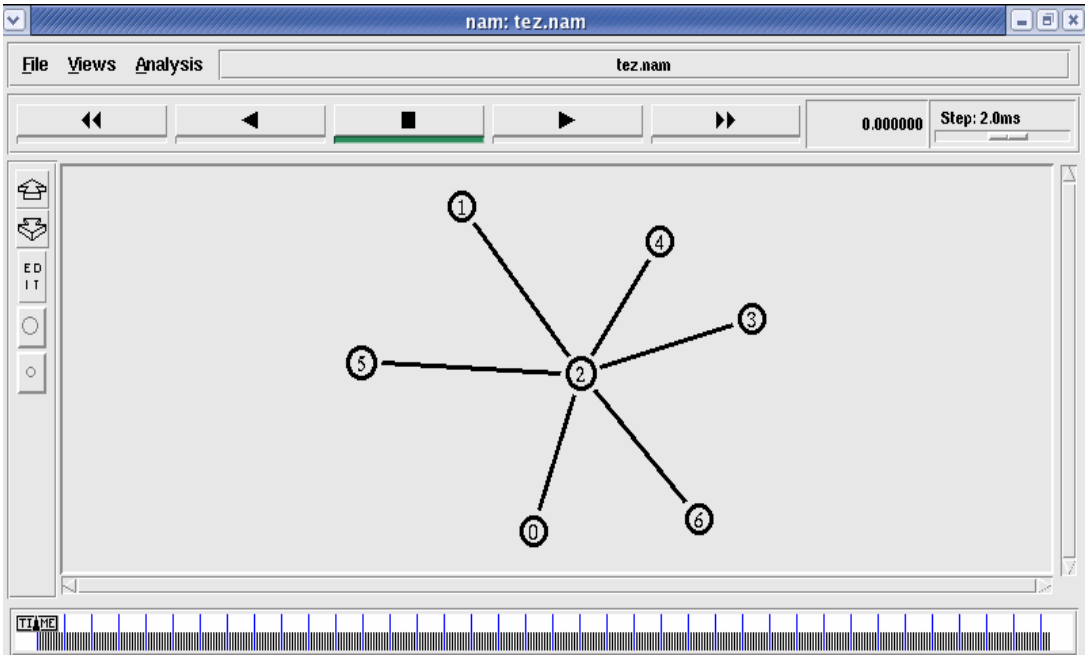
atılmaktadır. Bu sayede öncelikli olan bağlantılarda oluşabilecek tıkanıklık büyük ölçüde önlenmektedir.

Kuyruğa giren paket enque() fonksiyonuyla alınmakta ve deque() fonksiyonuyla hatta verilmekte ve hedefine gönderilmektedir. deque() fonksiyonu alınan paketlerin tekrardan hatta dönmesini sağlamaktadır. Bu fonksiyonlarda kullanılan değişken değerler drop-tail.h'tan alınmaktadır. Ayrıca IP.h dosyasından da bağlantı türlerini ayırt edebildiğimiz akış numaraları elde edilmektedir.

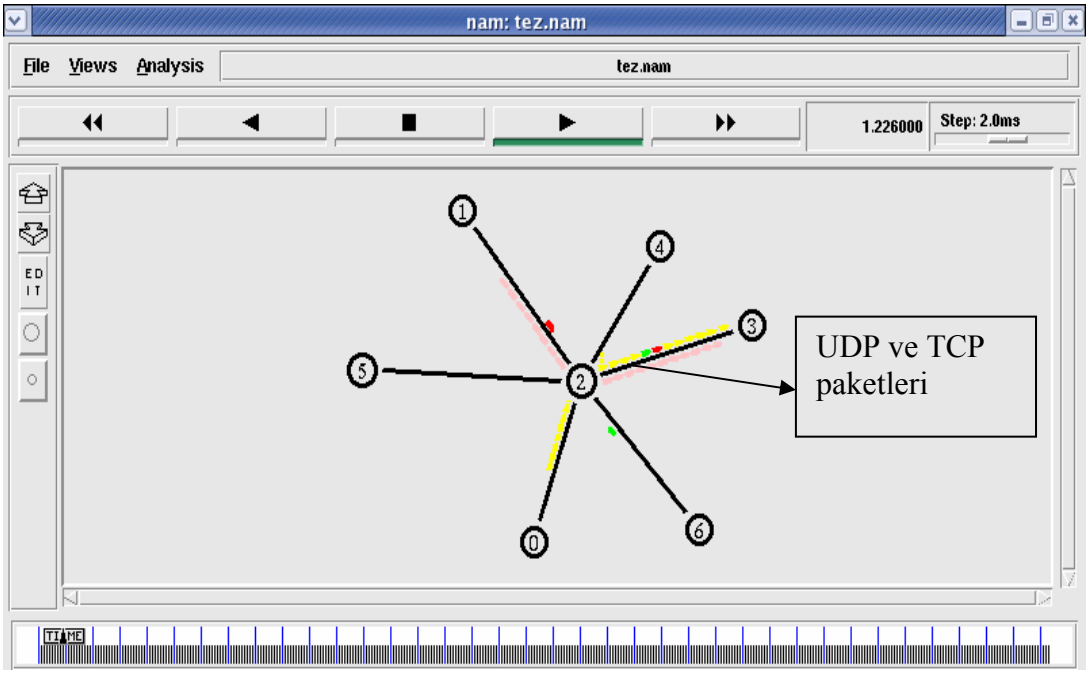
Kuyruk tipinde yapılan değişikliklerin ns-2 benzetim ortamında kabul edilmesi için derlenmesi gerekmektedir. Bunun için ilk önce değişiklik yapılan kuyruk tipinin yolu açılan terminal programına `touch` komutuyla birlikte yazılır. Daha sonra `make` komutuyla benzetim oratmında yeni kuyruk tipinin derlenmesi ve kullanılması sağlanmaktadır.

İkinci senaryo: Bu senaryoda ilk senaryoya ek olarak 2 tane yeni düğüm (Düğümlerden biri TCP bağlantısı kurmakta diğeri ise UDP bağlantısı kurmaktadır) daha eklenmiş ve bağlantıların bant genişliği 2 Mb'tan 10 Mb'a çıkarılmıştır. Ayrıca bütün TCP ve UDP paketlerinin gönderilme işlemi aynı anda olacak şekilde ayarlanmıştır. Zaman aralıkları ise 4 s'den 50 s'ye çıkarılmıştır. Şekil 3.7'de oluşturulan yeni senaryonun benzetim çıktısı görülmektedir.

Şekil 3.7'de gösterildiği gibi yeni senaryoya düğüm 5 ve düğüm 6 eklenmiştir ve benzetimin çalışma zamanı 4 s'den 50 s'ye çıkarılmıştır. Yeni eklenen düğümlerden düğüm 6 düğüm 3 ile UDP bağlantısı, düğüm 5 ise düğüm 3 ile TCP bağlantısı yapmaktadır. Önceki senaryoda olduğu gibi burada da düğüm 2 WAP gateway, düğüm 3'te WAP sunucu görevi görmektedir. Şekil 3.8'de ikinci senaryoda düğüm 3'e gönderilen paketlerin benzetim çıktısı gösterilmektedir.



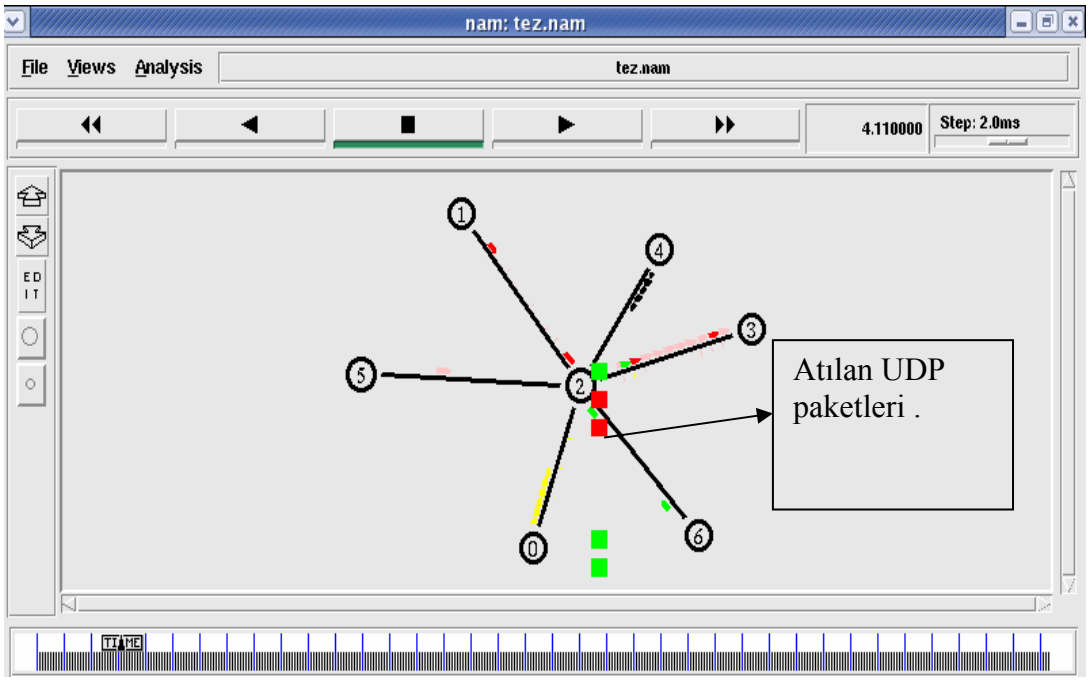
Şekil 3.7. İkinci senaryoda oluşturulan düğümlerin benzetim ekranı



Şekil 3.8. İkinci senaryoda düğüm 3'e gönderilen paketler

Şekil 3.8'de ikinci senaryoda gönderilen paketlerin benzetim çıktısında görüldüğü gibi gönderilen her paketin renkleri farklı olarak tanımlanmıştır. Burada görüldüğü gibi düğüm 2 üzerinden yapılan iletişimde kuyruk doluluk oranının %80 eşik

değerini aşmadığı için bütün paketlerin ilk gelen ilk çıkar mantığıyla gelen bütün paketler düğüm 3'e gönderilmektedir ve herhangi bir paket kaybı yaşanmamaktadır. Şekil 3.9'de ikinci senaryoda atılan paketlerin benzetim çıktısı ekranı görülmektedir.



Şekil 3.9. İkinci senaryoda atılan paketlerin benzetim ekranı

Şekil 3.9'de gösterildiği gibi kuyruk doluluk oranının eşik değeri %80'I aştığı durumlarda düğüm 1 ve düğüm 6'dan gelen UDP paketleri atılmaya başlanmaktadır. Bu rada TCP paketleri kuyruk doluluk oranı %100'e ulaştığında atılmaya başlanmaktadır. 1'den fazla TCP bağlantısı olduğu için ve hepsi aynı anda iletişime geçtiği için TCP paketleride atılabilmektedir. Ancak kuyruk doluluk oranının 580'I aştığı durumlarda atılamaya başlanan UDP paketleri sayesinde TCP paketlerindeki atılma oranı düşürülmüştür.

4. DENEYSEL SONUÇLAR

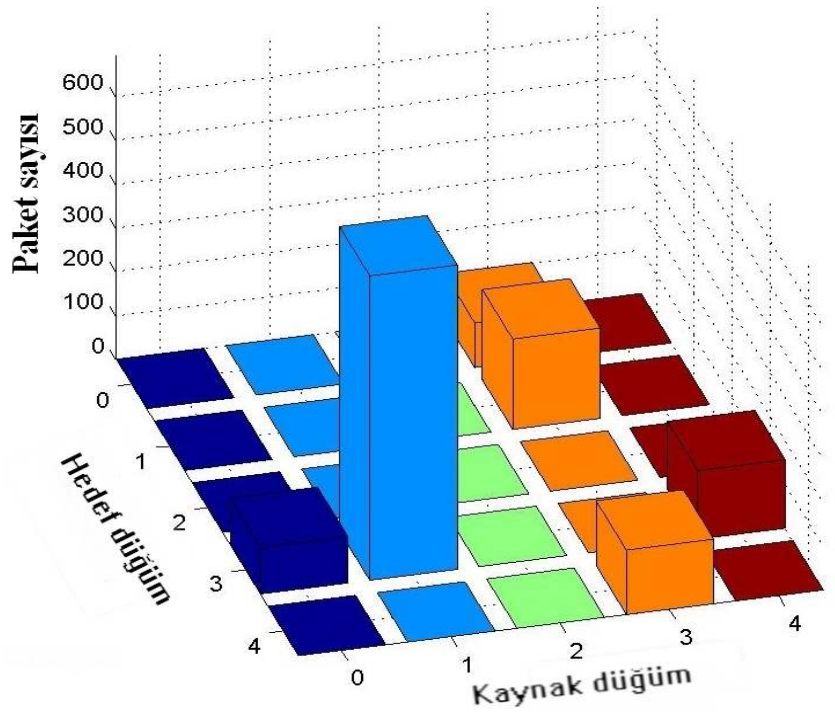
Geliştirilen protokol drop-tail kuyruk yapısı değiştirilerek elde edilmiştir. Elde edilen yeni protokolün benzetim sonuçlarının karşılaştırılması için oluşturulan iki senaryoda hem drop-tail kuyruk yapısının bulunduğu hem de red-tail kuyruk yapısının bulunduğu benzetim sonuçları alınmış ve oluşturulan yeni kuyruk yapısıyla karşılaştırılmıştır.

İlk Senaryonun deneysel sonuçları: İlk senaryoda bağlantıların genişliği 2 Mb, kuyruk uzunluğu 10 ve benzetim işlemi için zaman ayarı ise 4 saniye olarak ayarlanmıştır. Burada ilk önce drop-tail kuyruk yapısının benzetim sonuçları açıklanacak daha sonra red ve yeni oluşturulan kuyruk yapısı benzetim sonuçları açıklanacak ve karşılaştırma yapılacaktır. Benzetim sonuçlarında düğümlerde oluşturulan toplam paket sayısı, gecikme süreleri, atılan, gönderilen ve alınan paket sayılarının bulunduğu sonuçlar elde edilmiş ve karşılaştırma atılan paket sayısına ve gecikme sürelerine göre yapılmıştır. Burada elde edilen sonuçlarda gösterilen düğümlerden düğüm 2 WAP gateway görevi, düğüm 3 ise WAP sunucu görevi yapmaktadır.

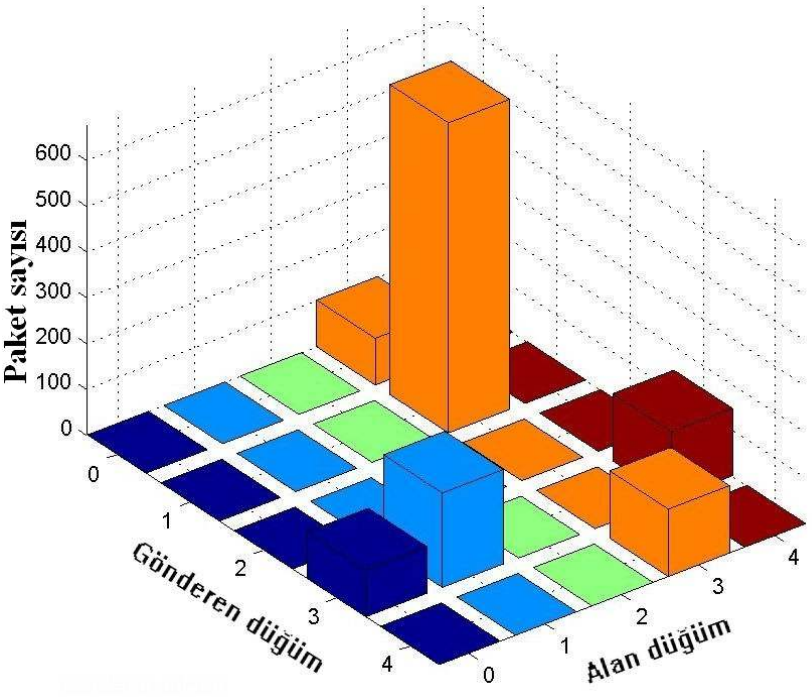
Benzetim sonuçları

Şekil 4.1'de elde edilen sonuçta drop-tail kuyruk tipinin benzetim işleminin başından sonuna kadar kaynak düğüm tarafından oluşturulan ve hedef düğümünün gösterildiği sonuç elde edilmiştir.

Bütün düğümlerde oluşturulan toplam paket sayılarının gösterildiği Şekil 4.1'de görüldüğü gibi drop tail kuyruk tipinde en fazla paket düğüm 1'de üretilmektedir. Düğüm 2 WAP gateway görevi yaptığı için herhangi bir paket üretmemiştir. Şekil 4.2'de bütün düğümlerde alınan paket sayıları gösterilmektedir.

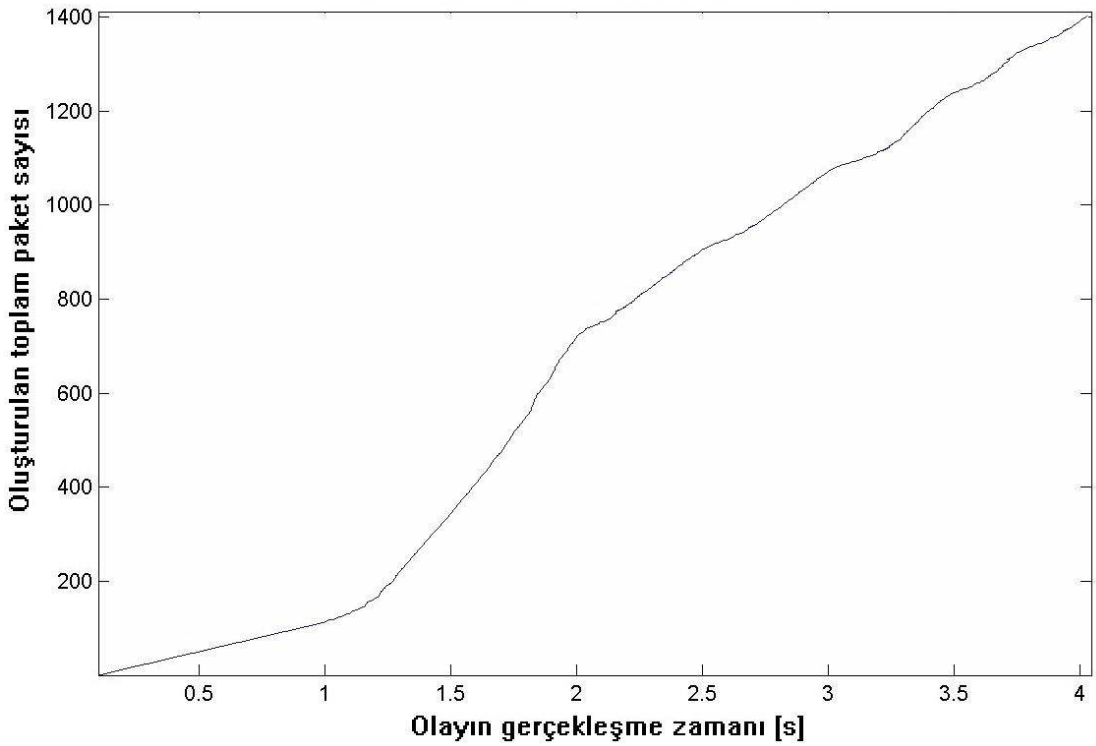


Şekil 4.1. Drop-Tail kuyruk tipinde bütün düğümlerde oluşturulan paket sayıları



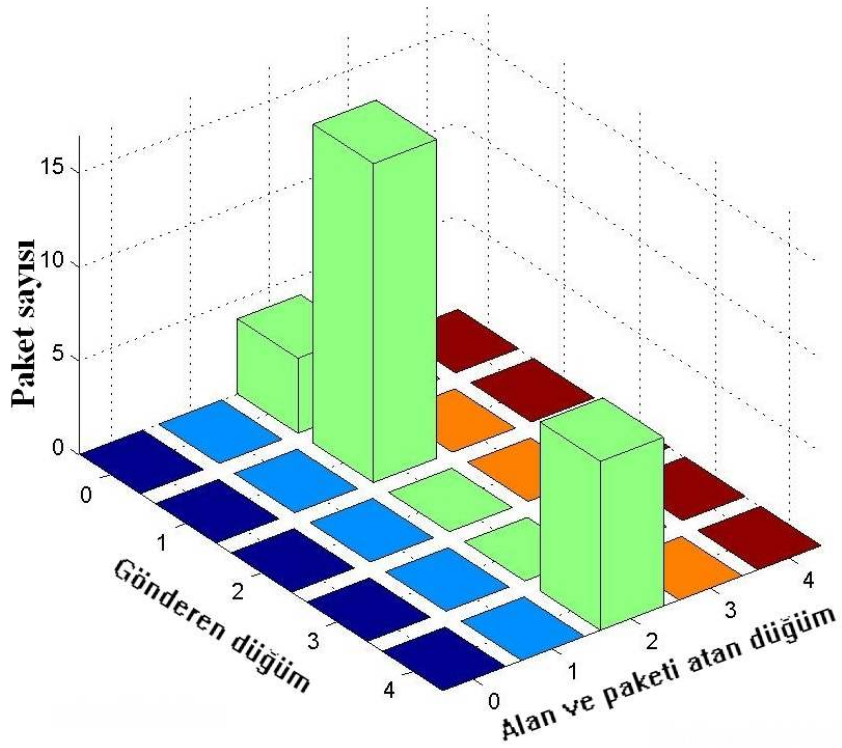
Şekil 4.2. Drop-Tail kuyruk tipinde bütün düğümlerde alınan paket sayıları

Şekil 4.2'de bütün düğümlerde benzetim işlemi sırasında alınan paket sayıları gösterilmektedir. Gönderen ve alan düğüm ekseninde gördüğümüz sayılar düğümleri simgelemektedirler. Burada düğüm 3'ün düğüm 0, düğüm 1 ve düğüm 4'e paket yolladığı görülmektedir. Düğüm 3'ün en fazla paket aldığı düğüm 1 ve düğüm 4 olarak görülmektedir. Düğüm 0, düğüm 1 ve düğüm 4'ün düğüm 3'ten cavap paketleri aldığı görülmektedir. Şekil 4.3'te oluşturulan tüm paketlerin zaman göre dağılımı gösterilmektedir.



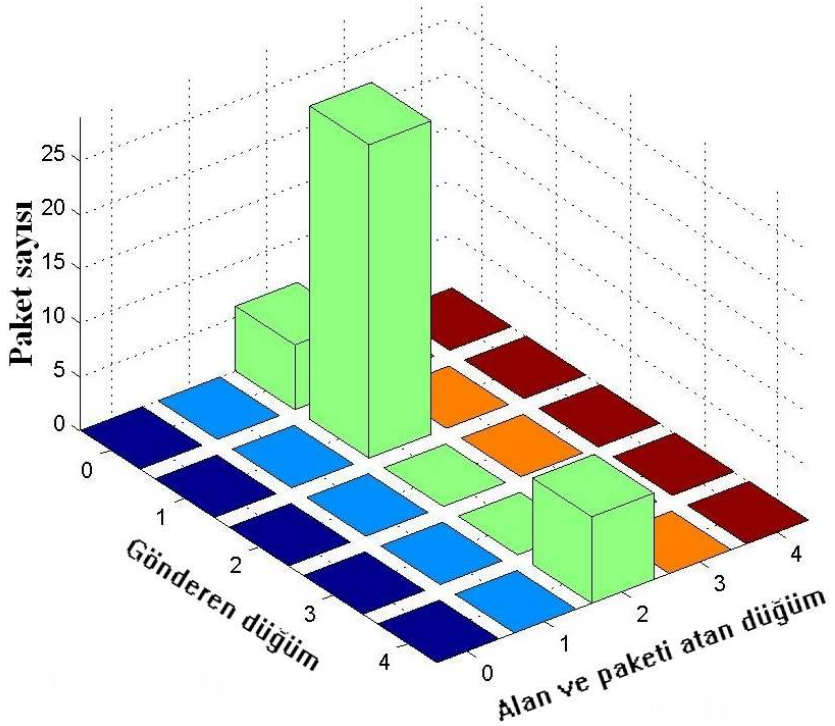
Şekil 4.3. Drop-Tail kuyruk tipinde oluşturulan tüm paketlerin zamana göre dağılımı

Şekil 4.3'te benzetim işleminin başından sonuna kadar saniye bazında oluşturulan toplam paket sayıları gösterilmektedir. Dikkat edilmesi gereken noktalar 1,2 s'den 2,5 s'ye kadar paket üretimi hızla artmıştır. Dolayısıyla 1,2 s'de Düğüm 0'da TCP paketlerininde oluşturulmaya başlandığı anlaşılmıştır. Şekil 4.4'te ise tüm düğümlerde atılan paket sayısı gösterilmektedir.



Şekil 4.4. Drop-Tail kuyruk tipinde bütün düğümlerde atılan paket sayısı

Oluşturulan tüm paketler düğüm 2 üzerinden düğüm 3'e gönderildiği için atılan paketlerin çoğunluğunun düğüm 2'deki kuyruk yapısına uygun olmaması gerekmektedir. Şekil 4.4'te gönderen düğümlerden düğüm 2'deki kuyruk yapısına uygun olmayan paketlerin atıldığı görülmektedir. Burada düğüm 0 ve düğüm 4 düğüm 3 ile TCP bağlantısı kurmuştur. Düğüm 1 ise UDP bağlantısı kurmuştur. Drop- tail kuyruk yapısının kullanıldığı bu benzetim işleminde WAP gateway olarak kullanılan düğüm 2'de atılan paketlerin hem UDP hem de TCP'de oluştuğu Şekil 4.4' te gösterilmektedir. Drop-tail kuyruk yapısında ilk giren ilk çıkar mantığı uygulanmıştır. Diğer kuyruk yapılarında da bu senaryo üzerinde işlem gerçekleştirildiği için aynı sonuçlar elde edilen grafikler tekrar verilmemiştir. Şekil 4.5'te Red kuyruk tipinde atılan paket sayısını gösteren grafik verilmiştir.

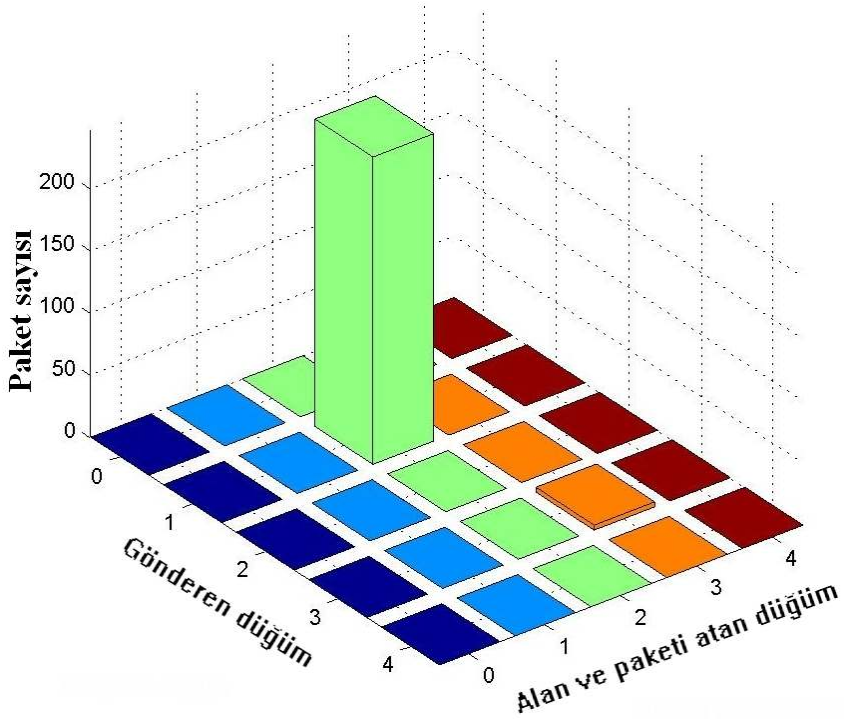


Şekil 4.5. Red kuyruk tipinde atılan paket sayısı

Red kuyruk tipinin kullanıldığı ikinci benzetim işleminde ise atılan paket sayılarının gösterildiği Şekil 4.5'e bakıldığında red kuyruk tipinde atılan paket sayısının drop-tail kuyruk yapısına göre UDP bağlantı içeren düğüm 1 de arttığı, TCP bağlantısı kullanan diğer düğümlerde ise hafif bir azalma olduğu görülmektedir. Ancak TCP bağlantılarının UDP bağlantılarına göre daha önemli olduğu bilinmektedir. Aşırı talep ile karşı karşıya kalındığında bu kuyruk tiplerinde önemli olan paketlerinde kaybolmasını sağlayabilen bir tıkanıklık oluşmaktadır. Şekil 4.6'da geliştirilen yeni protokolde atılan paket sayıları verilmektedir.

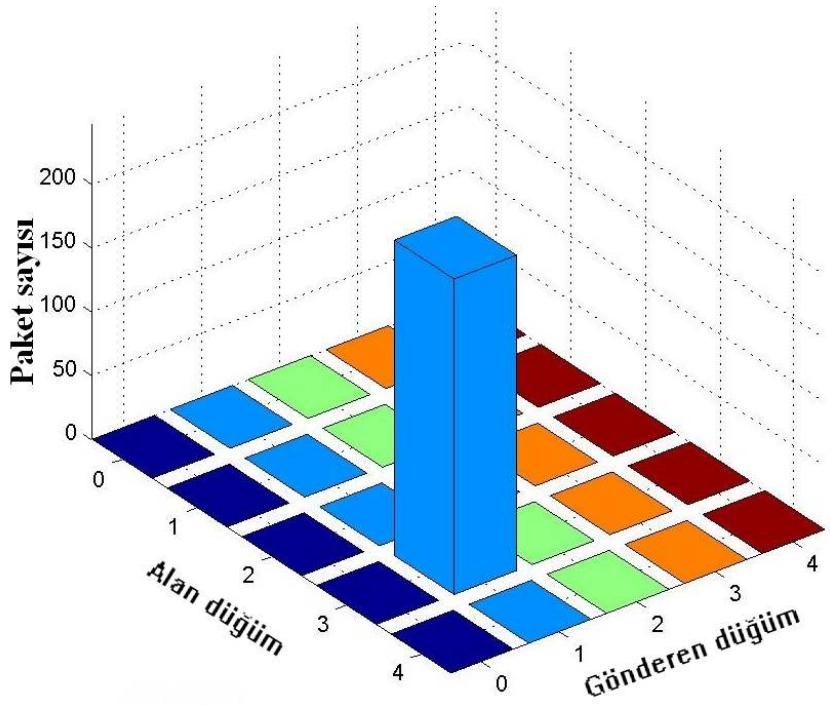
Şekil 4.6'da geliştirilen kuyruk yapısında paketin önceliğine göre WAP gateway olarak kullanılan düğüm 2'nin geçiş önceliğini ayarlaması sağlanmıştır. Burada drop-tail kuyruk yapısı değiştirilmiştir. TCP ve UDP bağlantının olduğu yerlerde kuyruğun doluluk oranına bakılmıştır. Eğer kuyruğun doluluk oranı %80'den az ise gelen paketin TCP veya UDP bağlantısı olup olmadığına bakılmadan ilk giren ilk çıkar mantığıyla paketlerin hedeflerine gidişine izin verilmiştir. Fakat kuyruğun doluluk oranı %80'den fazla olduğu durumlarda TCP bağlantılarına öncelik verilmiş

ve kuyruğun doluluk oranı %80'in altına inene kadar gelen UDP paketleri atılmıştır. Şekil 4.6'da gösterildiği gibi bu senaryoda TCP bağlantısı UDP bağlantısından öncelikli durumda olduğu için tüm benzetim işlemi sırasında TCP'de herhangi bir kayıp yaşanmamış ancak UDP'de kuyruğun doluluk oranının %80'i geçtiği durumlarda atılan paket sayısının olduğu görülmektedir. Şekil 4.7'de düğüm 1 UDP olduğu için önceliklendirmeye göre kaybolan paket sayıları gösterilmektedir.

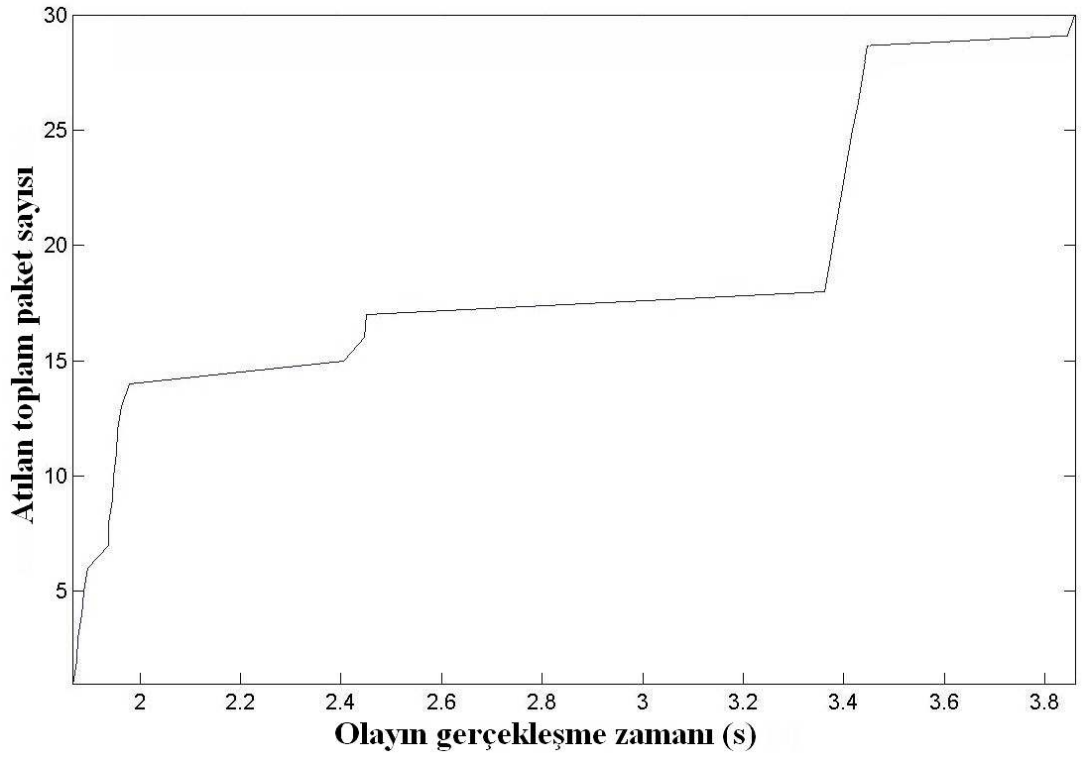


Şekil 4.6. Geliştirilen protokolde atılan paket sayısı

Şekil 4.7'de görüldüğü gibi bütün benzetim işlemi boyunca bu senaryoya göre paketlerde kaybın sadece düğüm 1'de olduğu görülmektedir. Bunun sebebi ise düğüm 1'in düğüm 3 ile UDP bağlantısı kurması ve kuyruğun doluluk oranının %80'i geçtiği durumlarda UDP paketlerinin atılması ile oluşmaktadır. Şekil 4.8'de benzetimde drop tail kuyruk tipinde atılan toplam paket sayılarının zamana göre dağılımları gösterilmektedir.

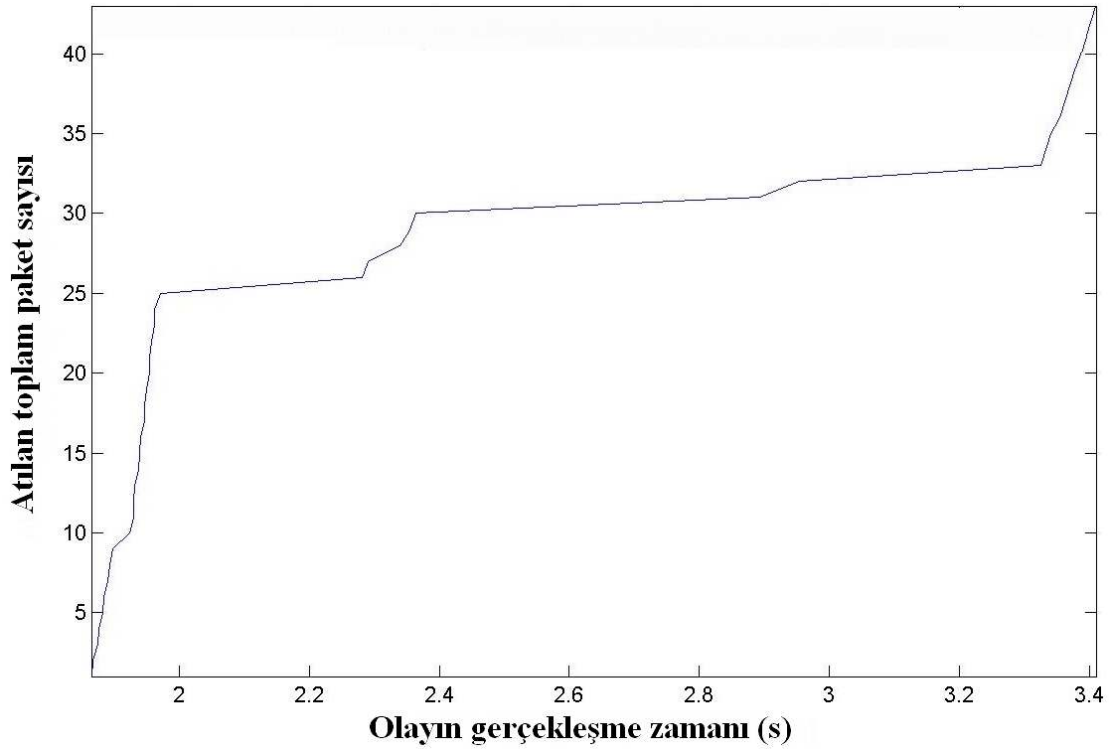


Şekil 4.7. Geliştirilen kuyruk tipinde bütün düğümlerde kaybolan paket sayısı



Şekil 4.8. Drop-tail kuyruk tipinde atılan toplam paketlerin zamana göre dağılımı

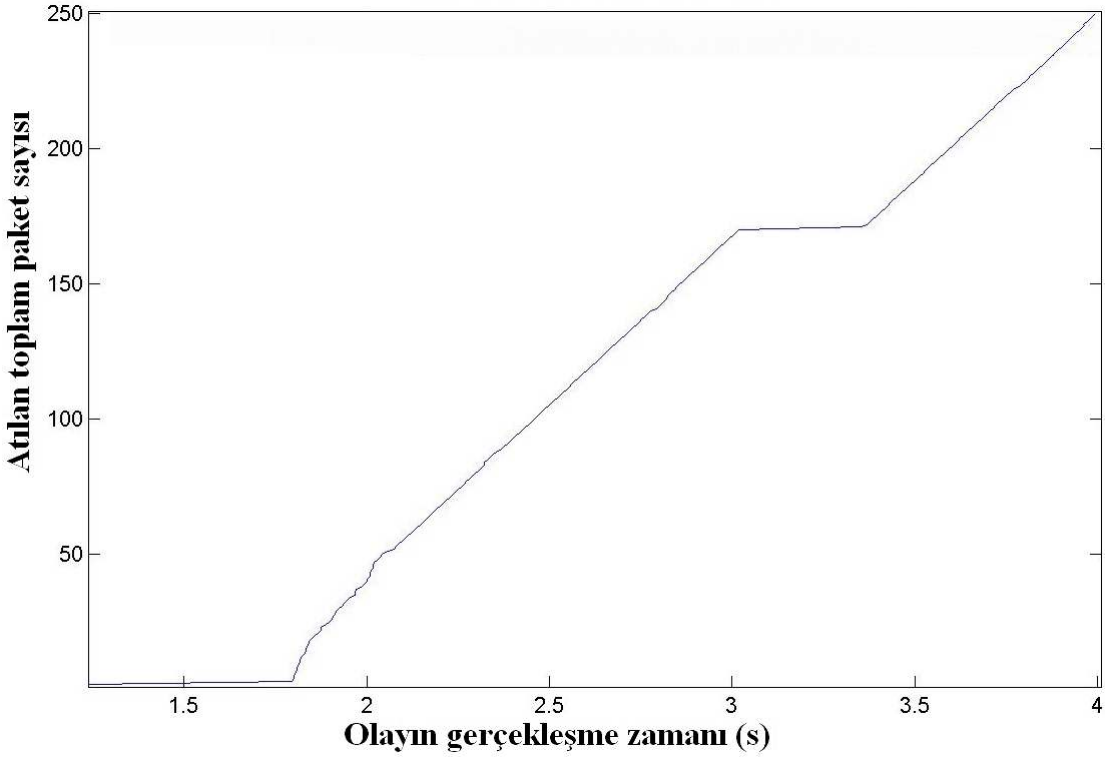
Şekil 4.8'de drop tail benzetimi sonucu elde edilen grafikte 1,7 s'de kuyruk doluluk oranı %100'e ulaştığı için paket atımlarına başlanmaktadır. Burada drop tail ilk giren ilk çıkar mantığını uyguladığı için gelen TCP paketleri de atılmaktadır. Dolayısıyla 2. s'ye kadar paket atımında dik bir ivme görülmektedir. Dikkat edilmesi gereken diğer bir nokta ise 3.3 ve 3.5 s'leri arasında paket atım ivmesi çok yüksektir. Tüm benzetim işleminde toplam paket sayısı diğer kuyruk tiplerine göre daha düşüktür. Ancak, paket atım ivmesi diğer kuyruk tiplerinden daha fazladır. Şekil 4.9'da benzetimde red kuyruk tipinde atılan toplam paket sayılarının zamana göre dağılımları gösterilmektedir.



Şekil 4.9. Red kuyruk tipinde atılan toplam paketlerin zamana göre dağılımı

Şekil 4.9'da gösterilen grafikte red kuyruk tipinde atılan toplama paket sayılarının zamana göre dağılımı gösterilmektedir. Burada 1,7 s'de başlayan paket atımları hemen hemen dik bir ivme çizmektedir. Ancak 2. s'den sonra paket atımındaki ivme azalmaktadır. Dolayısıyla atılan paket sayısı drop tail kuyruk tipine göre fazla olabilir ancak paket iletim sürekliliği bu kuyruk tipinde daha fazladır. Şekil 4.10'da

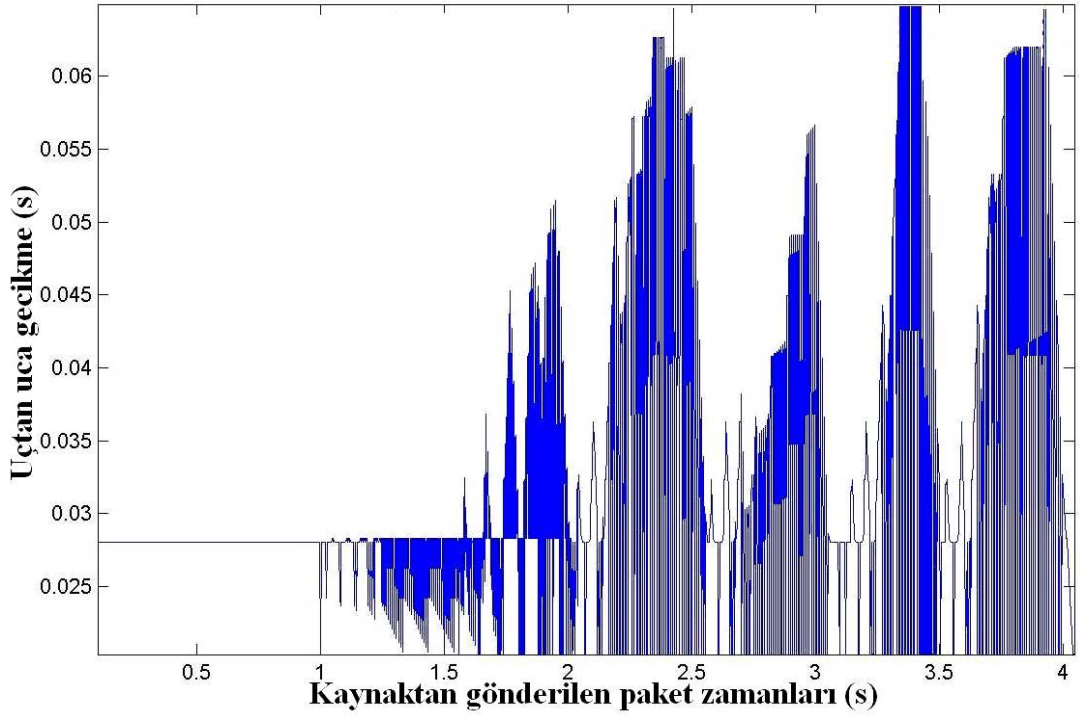
geliştirilen kuyruk tipinde atılan toplam paket sayısının zamana göre dağılımı verilmiştir.



Şekil 4.10. Geliştirilen kuyruk tipinde atılan toplam paketlerin zaman göre dağılımı

Şekil 4.10'da görüldüğü gibi 1,7 s'ye kadar benzetimde paketin atılmadığı görülmektedir. 1,7 s'den sonra TCP paketlerinin oluşturulması ve kuyruk doluluk oranının %80'i aştığı zaman aralıklarında UDP paketlerinin atılmasından dolayı toplam atılan paket sayısı artmaktadır. 3 s ile 3,3 s arası paket atılmamasının nedeni ise bir TCP bağlantısının bitip diğer TCP bağlantısının başlaması için geçen sürede kuyruk doluluk oranının %80'in altına düşmesi ve paket atılmamasından dolayıdır. Son açıklanan 3 şekilden de anlaşılacağı gibi paket atma ivmesi en az yeni geliştirilen protokolde artmaktadır. Paket atımının fazla olmasının nedeni ise diğer kuyruk tiplerinin paket atımı için kuyruk doluluk oranının %100'e ulaşması beklenmekte ancak yeni geliştirilen protokolde paket atımı için kuyruk doluluk oranının %80 olmasından dolayıdır.

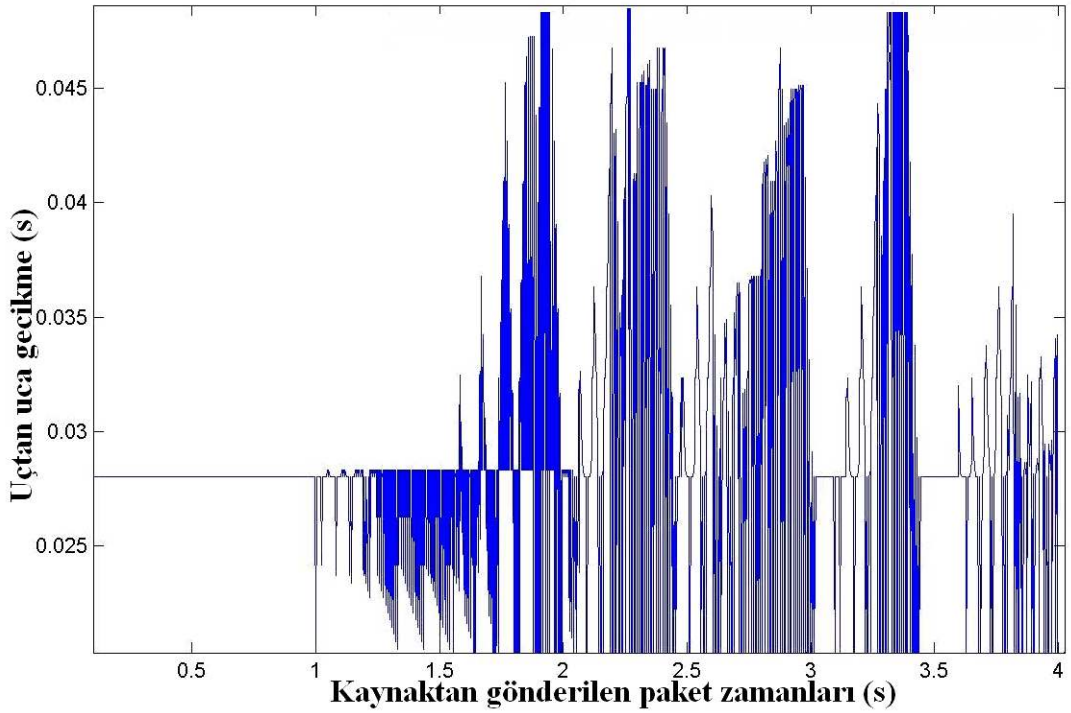
Diğer bir karşılaştırma işlemi ise gecikme süreleri üzerinden yapılmıştır. Bunun için ilk önce drop-tail kuyruk yapısı daha sonra ise red ve geliştirilen kuyruk yapısının benzetim sonuçları elde edilerek yapılmıştır. Drop-tail kuyruk yapısında gecikme sürelerini ve iletişimin devamlılığını gösteren benzetim sonucu Şekil 4.11'da gösterilmektedir.



Şekil 4.11. Drop tail kuyruk yapısında elde edilen gecikme süreleri

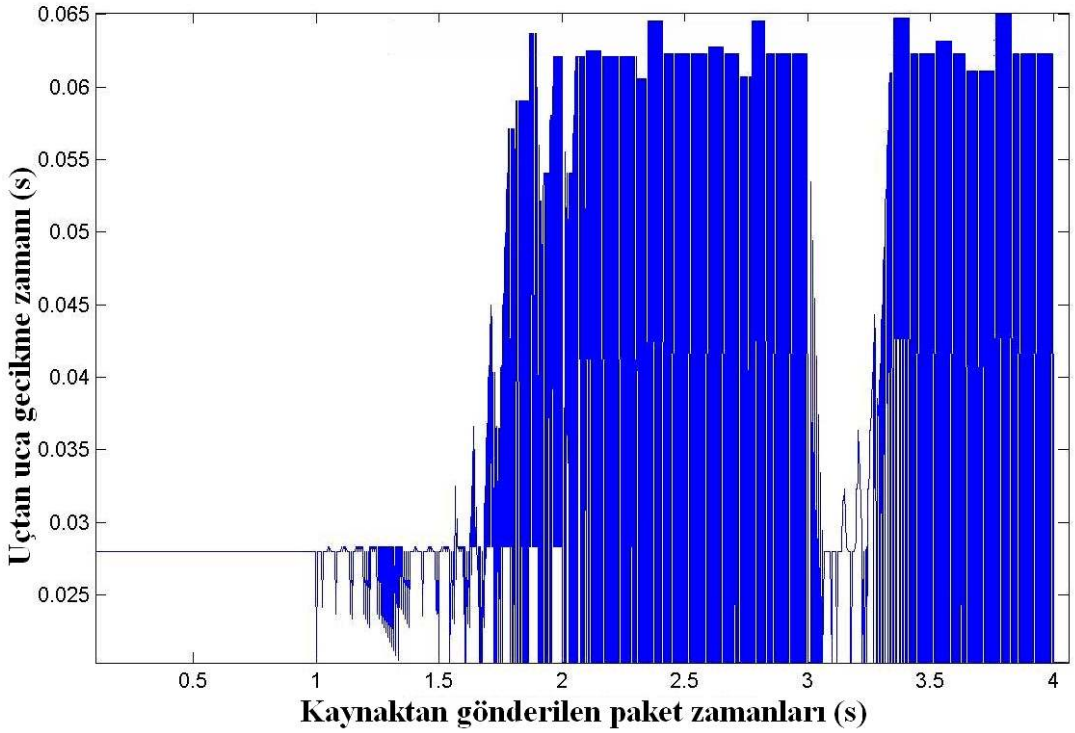
Senaryodaki benzetim çalıştırılmaya başlandığında benzetimde paket gönderme işlemi 1sn'de başlamaktadır. İlk gönderme işlemini yapan düğüm 1 UDP bağlantısı ile paketleri yollamaktadır. İlk başta kuyruk doluluk oranı fazla olmadığı için gecikme süresi azdır. TCP bağlantısı başladığı andan itibaren yani 1.7 saniyeden sonra gecikme sürelerinin arttığı gözükmektedir. 2, 2,5 ve 3,5'inci saniyelerde kuyruk doluluk oranı arttığı için atılan paketler sayesinde gecikme süreleri çok az bir zaman aralığında düştüğü görülmektedir. 3,5'inci saniyede ise bir TCP bağlantısının bittiği ikinci TCP bağlantısının başladığı zaman aralığı olduğu için bu zaman aralığında da gecikme süresinin azaldığı gözlenmektedir. Gecikme süresi bu kuyruk tipinde 0,06'nın üzerine çıkmaktadır. Bu grafikte ve diğer grafiklerde gecikme

süreleri TCP bağlantıları baz alınarak yapılmaktadır. Şekil 4.12'da gösterilen ikinci benzetim sonucu red kuyruk tipi kullanılarak yapılan benzetim sonucunda elde edilmiştir.



Şekil 4.12. Red kuyruk yapısında elde edilen gecikme süreleri

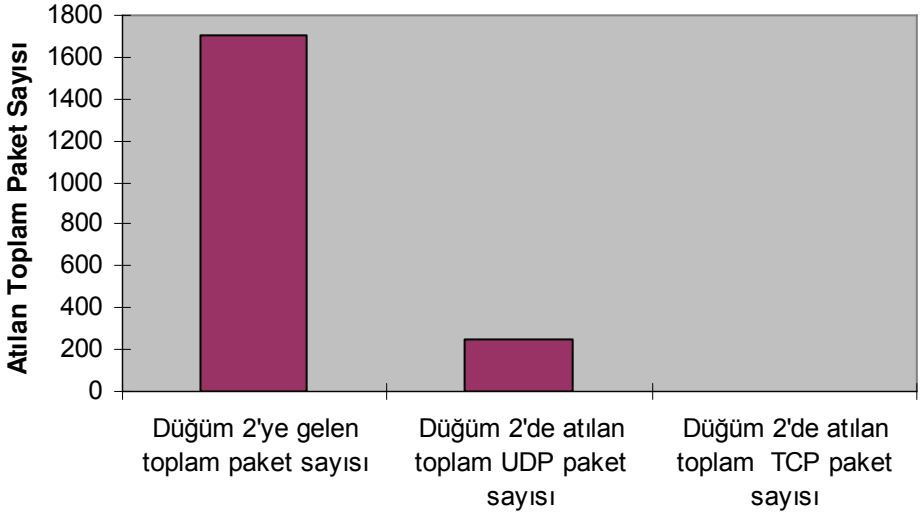
Bu benzetim işleminde de ilk paket 1'inci saniyede gönderilmeye başlanmaktadır. Bağlantı tipi ise UDP'dir. 1,7'inci saniyede başlayan TCP tipi paket gönderiminin başladığı anda gecikme sürelerinin drop tail ile aynı zaman aralıklarında meydana geldiği gözükmektedir. Ancak grafiğe bakıldığında red kuyruk tipinde drop tail kuyruk tipine oranla daha az TCP paketi atıldığı görülmekte ve gecikme süresinin 0,045 saniye ile drop tail kuyruk tipinden daha az olduğu görülmektedir. Red kuyruk tipinde de TCP paketlerinin atıldığı görülmektedir. Geliştirilen kuyruk tipinin benzetimi sonucunda elde edilen sonuç grafik Şekil 4.13'de gösterilmektedir.



Şekil 4.13. Geliştirilen kuyruk yapısında elde edilen gecikme süreleri

Burada 1.7'inci saniyede gönderilmeye başlanan TCP paketlerinin gecikme süreleri diğer kuyruk tipine göre daha fazla olmasına karşın TCP paketlerinin atılmadığı ve paket iletişiminin diğer kuyruk tiplerine oranla çok daha iyi olduğu gözlenmektedir. 3.1'inci saniyede başlayan ve gecikme süresinin çok düştüğü zaman aralığında ise birinci TCP bağlantısının bitip ikincisinin başladığı ana denk geldiği görülmektedir. Bu grafikte de görüldüğü gibi önceliği bulunan TCP paketlerinin kuyruk doluluk oranı %80'i aştığı zaman atılmadığı ve geliştirilen kuyruk tipinde TCP paketlerinin daha başarılı bir şekilde hedefine gönderildiği görülmektedir.

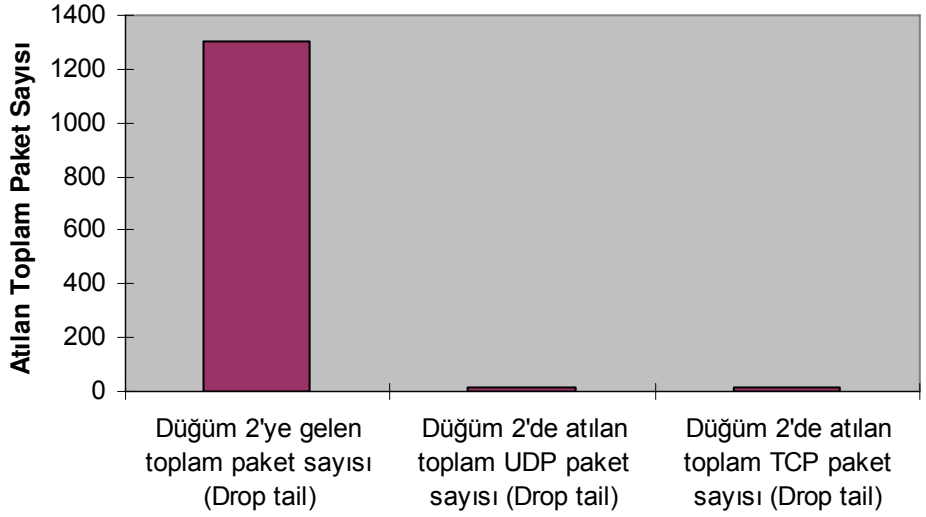
Geliştirilen protokolün benzetim işlemi sonucunda elde edilen diğer bir istatiksel bilgi Şekil 4.14'te gösterilen düğüm 2 tarafından kuyruk doluluk oranı dolunca atılan TCP ve UDP paket oranlarıdır.



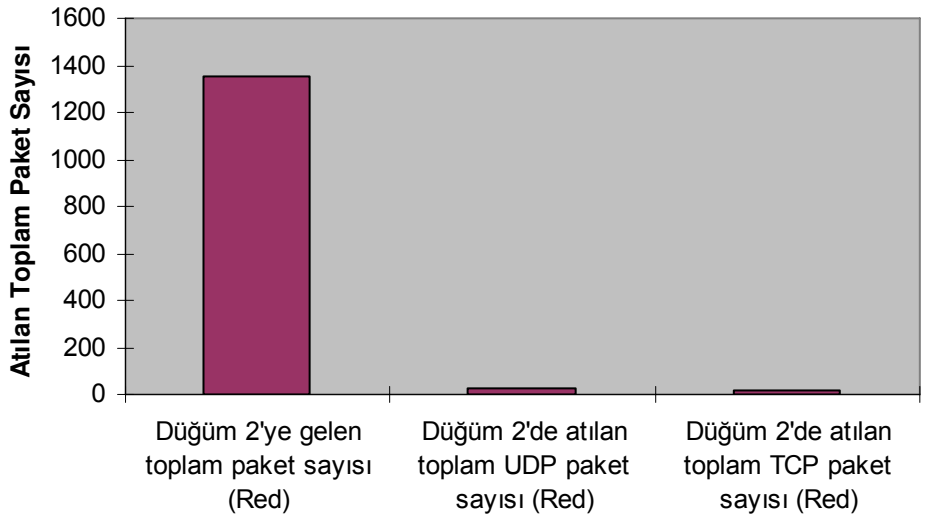
Şekil 4.14. Düğüm 2'de atılan TCP ve UDP paket oranları(geliştirilen)

Şekil 4.14'tede gösterildiği gibi düğüm 2 'ye toplam 1700 tane paket gelmiştir ve bu gelen paketlerden TCP paketlerinde herhangi bir kayıp olmadan düğüm 3'e gönderilmiştir. Ancak UDP paketlerinden 251 tanesi atılmıştır. Bu oranlar kuyruk doluluk oranının %80'i aştığı durumlarda elde edilmiştir. Şekil 4.15'te ise düğüm 2'de atılan TCP ve UDP paket oranları için drop tail kuyruk tipinde elde edilen sonuçlar verilmiştir.

Şekil 4.15'te gösterildiği gibi benzetimde drop tail kuyruk tipi kullanıldığında düğüm 2'ye gelen toplam paket sayısı 1300 olarak tespit edilmiştir. Atılan TCP paket sayısı 13 ve UDP paket sayısı da 17 olarak bulunmuştur. Bu grafik kuyruk doluluk oranının %100'ü aştığı durumlarda elde edilen değerler için oluşturulmuştur. Şekil 4.16'da ise red kuyruk tipi kullanılarak gerçekleştirilen benzetimde düğüm 2'de atılan TCP ve UDP paket oranları verilmektedir.

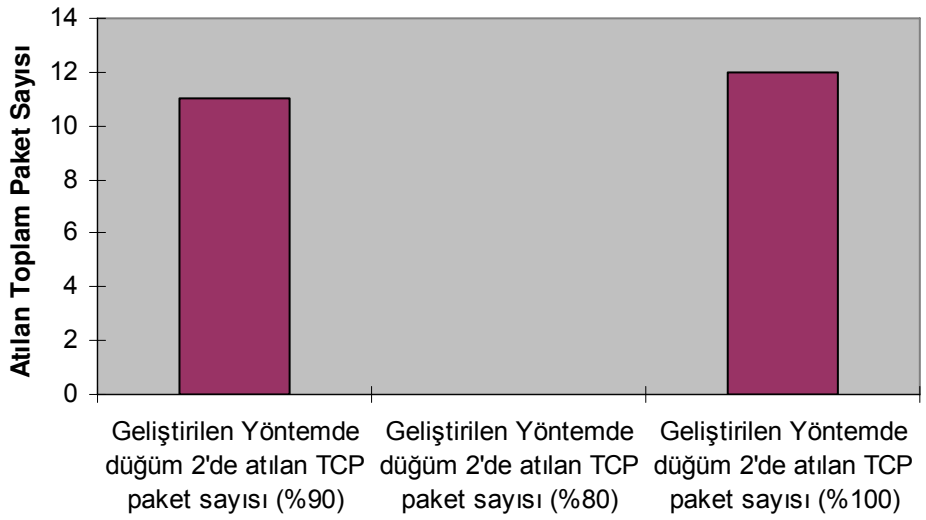


Şekil 4.15. Düğüm 2'de atılan TCP ve UDP paket oranları (drop-tail)



Şekil 4.16. Düğüm 2'de atılan TCP ve UDP paket oranları (red)

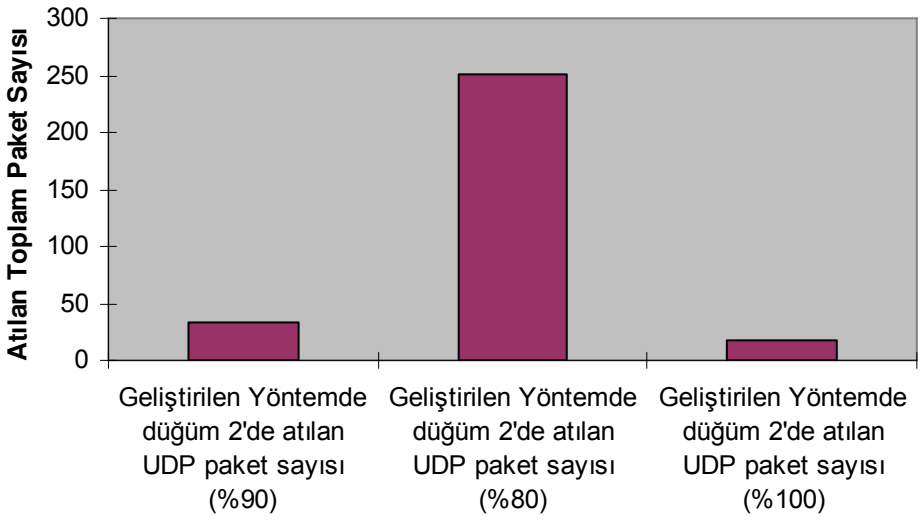
Şekil 4.16'te gösterildiği gibi benzetimde red kuyruk tipi kullanıldığında düğüm 2'ye gelen toplam paket sayısı 1350 olarak tespit edilmiştir. Atılan TCP paket sayısı 14 ve UDP paket sayısı da 29 olarak bulunmuştur. Bu grafik kuyruk doluluk oranının %100'ü aştığı durumlarda elde edilen değerler için oluşturulmuştur. Şekil 4.17'de geliştirilen protokolde kuyruk doluluk oranlarına göre TCP paket atım oranları verilmiştir.



Şekil 4.17. Geliştirilen protokolde kuyruk doluluk oranlarına göre TCP paket atım oranları

Şekil 4.17'de gösterildiği gibi kuyruk doluluk oranı %100'ü aştığında paket atılma olayını gerçekleştiren benzetim işleminde düğüm 2'de atılan toplam TCP paket sayısı 12 olarak bulunmuştur. Kuyruk doluluk oranının eşik değeri %90 yapıldığından düğüm 2'de 11 TCP paketinin atıldığı görülmüştür. Kuyruk doluluk oranının eşik değeri %80'e çekildiğinde herhangi bir TCP paket kaybının olmadığı görülmüştür. Şekil 4.18'de geliştirilen protokolde kuyruk doluluk oranlarına göre UDP paket atım oranı verilmiştir.

Şekil 4.18'de gösterildiği gibi kuyruk doluluk oranı %100'ü aştığında paket atılma olayını gerçekleştiren benzetim işleminde düğüm 2'de atılan toplam UDP paket sayısı 17 olarak bulunmuştur. Kuyruk doluluk oranının eşik değeri %90 yapıldığından düğüm 2'de 34 UDP paketinin atıldığı görülmüştür. Kuyruk doluluk oranının eşik değeri %80'e çekildiğinde ise toplam 251 tane UDP paketin atıldığı görülmüştür.

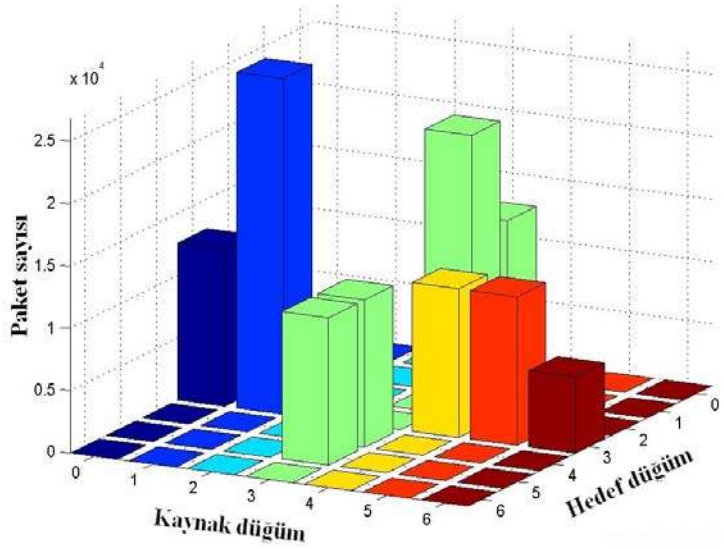


Şekil 4.18. Geliştirilen protokolde kuyruk doluluk oranlarına göre UDP paket atım oranları

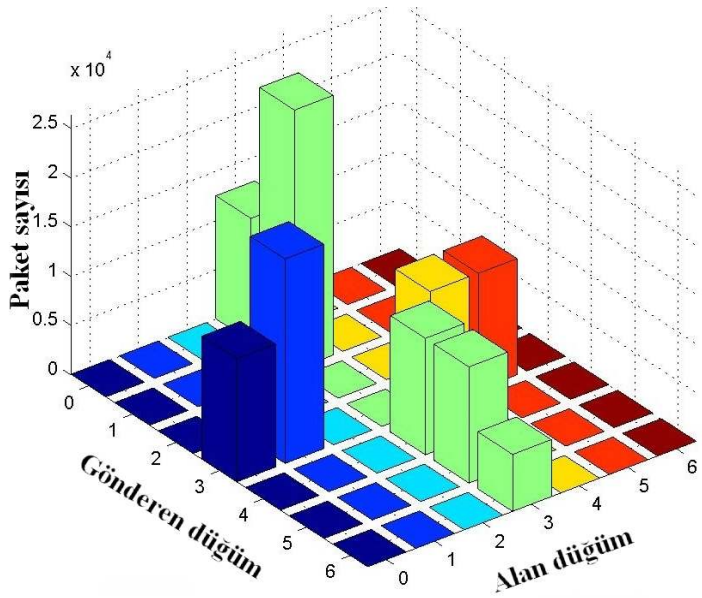
İkinci senaryonun deneysel sonuçları: İkinci senaryoda bağlantıların bant genişliği 10 mb, kuyruk uzunluğu 10 ve benzetim işlemi için zaman ayarı ise 50 saniye olarak belirlenmiştir. Burada ilk önce drop-tail kuyruk yapısının benzetim sonuçları açıklanacak daha sonra red ve yeni oluşturulan kuyruk yapısı benzetim sonuçları açıklanacak ve karşılaştırma yapılacaktır. Benzetim sonuçlarında düğümlerde oluşturulan toplam paket sayısı, gecikme süreleri, atılan, gönderilen ve alınan paket sayılarının bulunduğu sonuçlar elde edilmiş ve karşılaştırma atılan paket sayısına ve gecikme sürelerine göre yapılmıştır. Burada elde edilen sonuçlarda gösterilen düğümlerden düğüm 2 WAP gateway görevi, düğüm 3 ise WAP sunucu görevi yapmaktadır. Şekil 4.19'da ikinci senaryoda düğümlerde oluşturulan paket sayıları gösterilmektedir.

Şekil 4.19'da görüldüğü gibi bütün düğümlerde oluşturulan paketler farklı renklerde gösterilmişlerdir. Düğüm 1 ve düğüm 6 UDP paket üretmiştir. Diğer düğülerin ürettikleri paketlerin tipi ise TCP'dir. Düğüm 2 de paket üretilmemiştir. Çünkü, düğüm 2 WAP gateway görevi gördüğü için sadece gelen paketleri kuyruk doluluk oranı ve önceliklendirme koşullarına göre düğüm 3'e göndermektedir. Şekil 4.20'de drop tail kuyruk tipinde paketlerin düğümler tarafından alım oranları

gösterilmektedir. Şekil 4.20'de drop tail kuyruk tipinde paketlerin düğümler tarafından alım oranı gösterilmektedir.



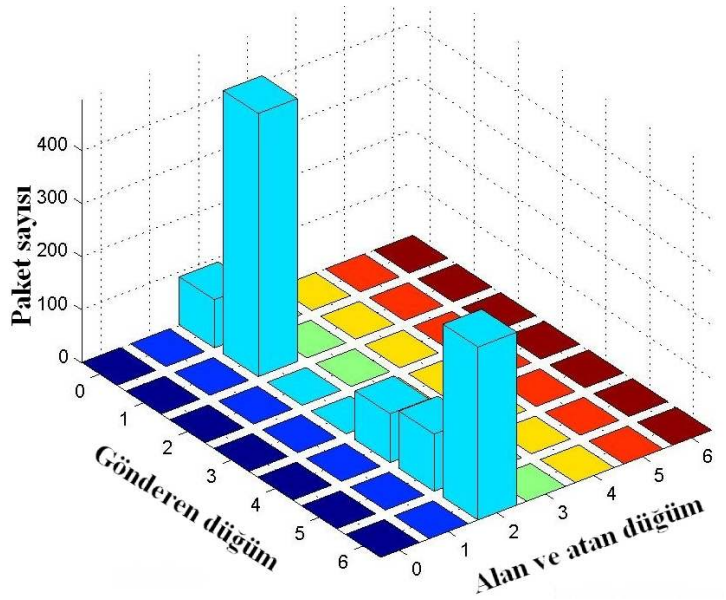
Şekil 4.19. Drop-tail'de bütün düğümlerde oluşturulan paket sayıları



Şekil 4.20. Drop-tail kuyruk tipinde paketlerin düğümler tarafından alım oranı

Şekil 4.20'de gösterildiği gibi düğüm 2 hiç paket almamıştır. Bunun sebebi düğüm 2'nin WAP gateway görevi görmesidir. En fazla paketi alan düğüm olan düğüm 3 ise

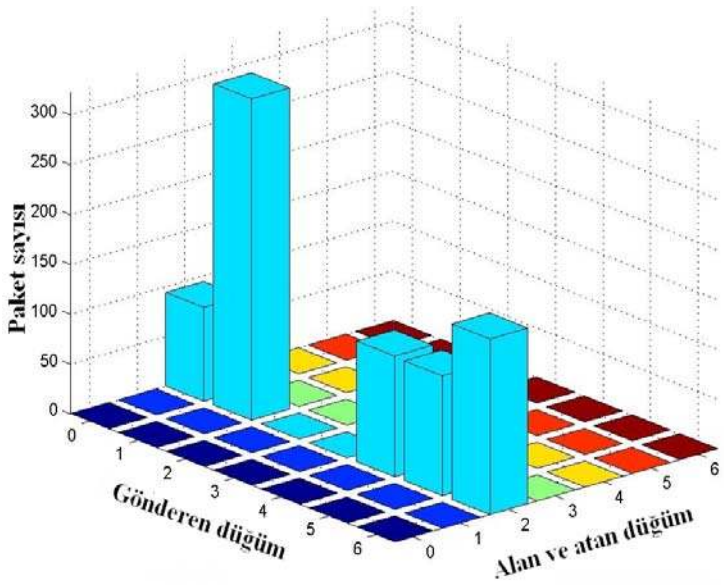
WAP sunucu görevi gördüğü için diğer düğümlerden gelen paketleri kuyruk tipi drop tail olduğu için ilk giren ilk çıkar mantığına göre almıştır.



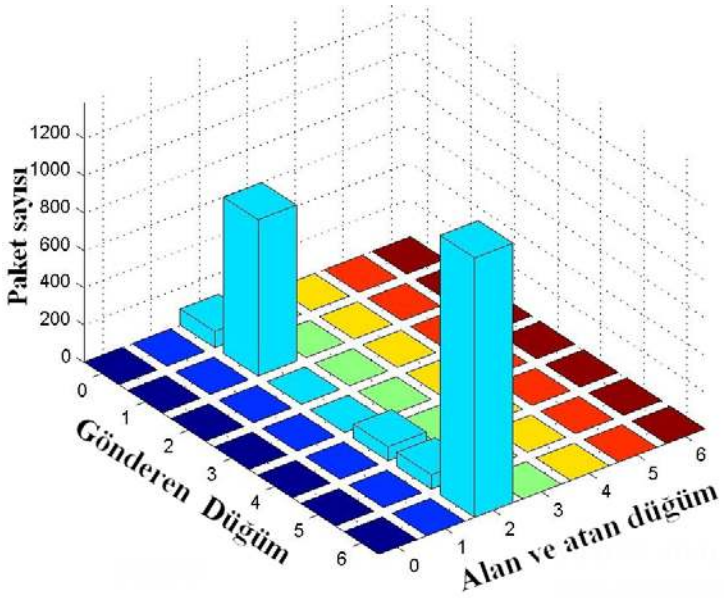
Şekil 4.21. Drop-tail kuyruk tipinde atılan paket sayılarının düğümlere göre oranı

Şekil 4.21'de gösterildiği gibi benzetimde drop tail kuyruk tipi kullanıldığında düğümlerden gelen paketlerin atılma oranları verilmiştir. Drop tail'de ilk giren ilk çıkar mantığı uygulandığı için kuyruk doluluk oranının %100'ü aştığı durumlarda düğümlere göre atılan paket oranları verilmiştir. Şekil 4.22'de red kuyruk tipinde atılan paket sayılarının düğümlere göre oranı verilmiştir.

Şekil 4.22'de benzetimde red kuyruk tipi uygulandığında elde edilen sonuçlar gözükmektedir. Red kuyruk tipi benzetimde kuyruk doluluk oranı %100'e ulaştığında paket atımlarının başlaması şeklinde çalıştırılmıştır. UDP paketlerinde daha başarılı olduğu görülmekle birlikte TCP paketlerinde drop-tail kuyruk tipinden biraz daha başarısız olduğu görülmüştür. Şekil 4.23'te geliştirilen protokolde atılan paket sayılarının düğümlere göre oranı verilmektedir.



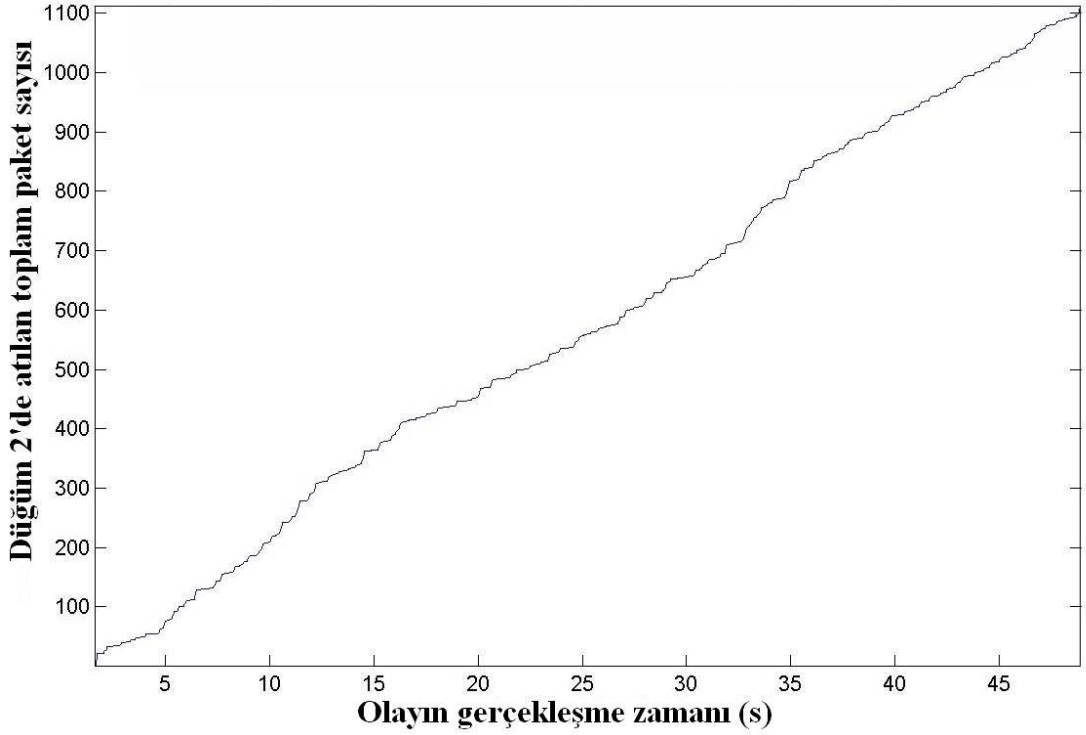
Şekil 4.22. Red kuyruk tipinde atılan paket sayılarının düğümlere göre oranı



Şekil 4.23. Geliştirilen protokolde atılan paket sayılarının düğümlere göre oranı

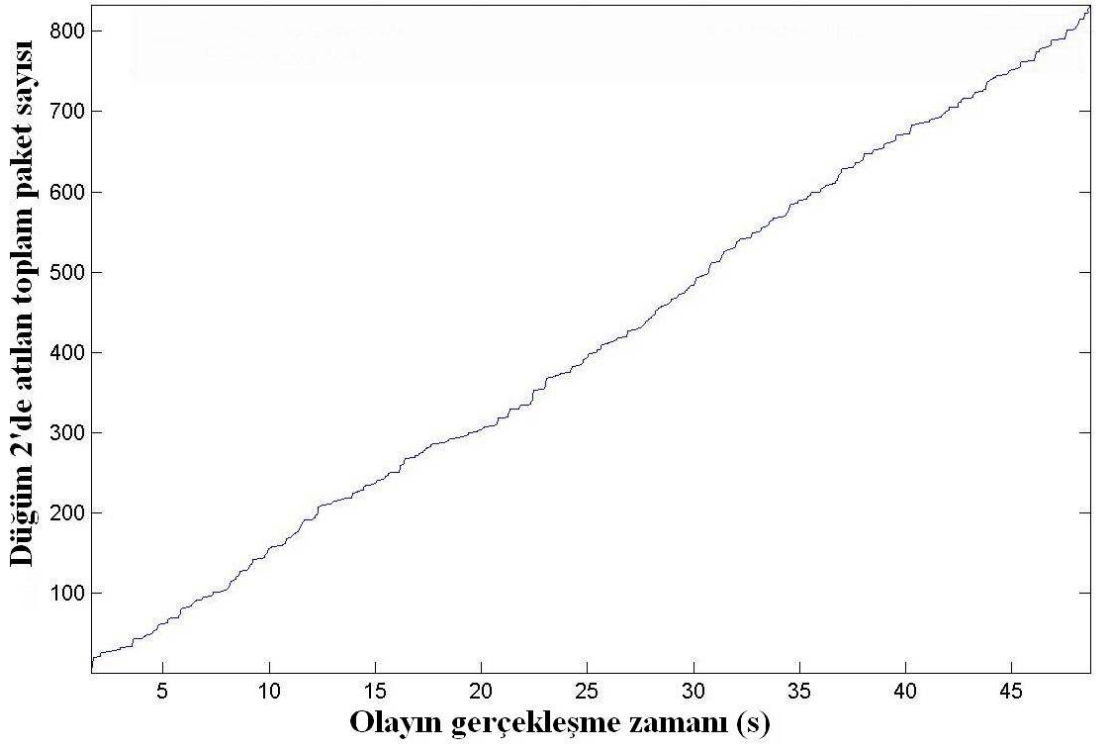
UDP paketlerinin fazla atılmış olmasına rağmen atılan TCP paketlerinde paket kayıplarının diğer kuyruk tiplerine göre daha az olduğu Şekil 4.23'te görülmüştür. Geliştirilen protokolde kuyruk doluluk oranı %80 olarak belirlenmiştir. Bunun sebebi ise öncelikli olan TCP paketlerinin atılmasını önlemektir. TCP paketlerinde azda olsa kaybın yaşanmasının sebebi ise aynı anda 5 düğümün birden düğüm 3 ile iletişim kurmalarından kaynaklanmaktadır. Ancak diğer düğümlere göre TCP paketlerinde

daha başarılı olduğu görülmüştür. Şekil 4.24'te drop tail kuyruk tipinde düğüm 2'de atılan paketlerin zamana göre dağılımı gösterilmektedir.



Şekil 4.24. Drop-tail kuyruk tipinde düğüm 2'de atılan paketlerin zamana göre dağılımı

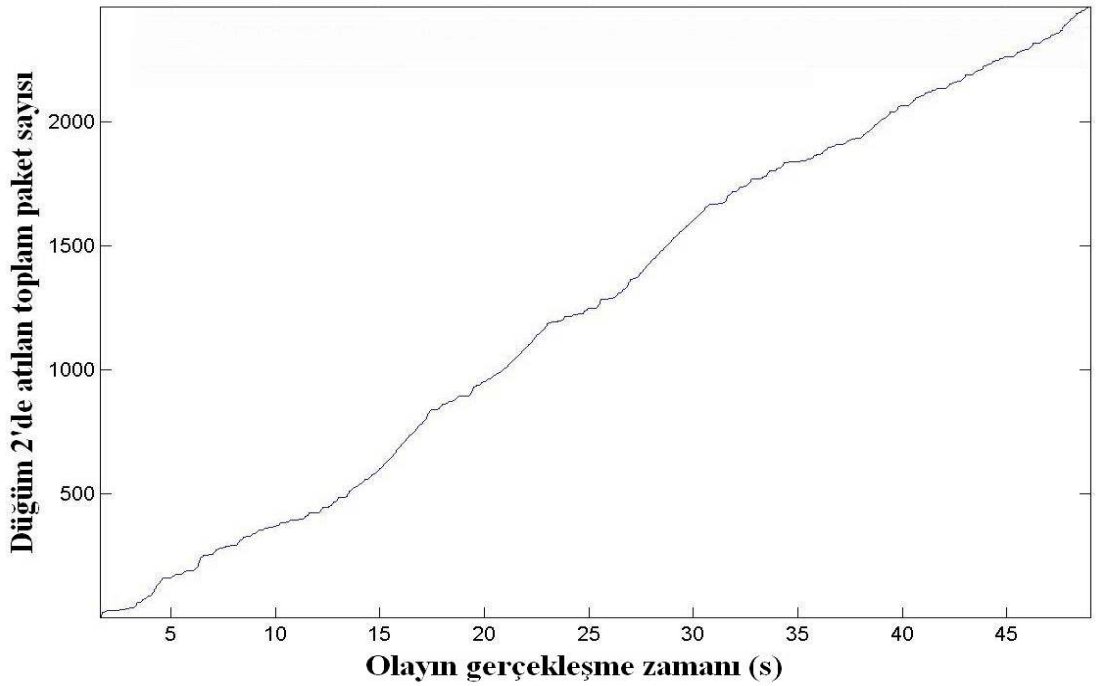
Şekil 4.24'te gösterildiği gibi drop tail kuyruk tipinde düğüm 2'de drop tail kuyruk tipinde atılan toplam paketlerin sayısı 1100'dür. Ancak bu atılan paketler ilk giren ilk çıkar mantığına göre atıldığı için atılan TCP paket sayısı geliştirilen protokole göre daha fazladır. 15 s ile 20 s ve 35 s ile 40 s arasında paket atımının arttığı görülmektedir. Kuyruk doluluk oranı %100 olarak alınmıştır. Şekil 4.25'te red kuyruk tipinde düğüm 2'de atılan toplam paket sayıları gösterilmektedir.



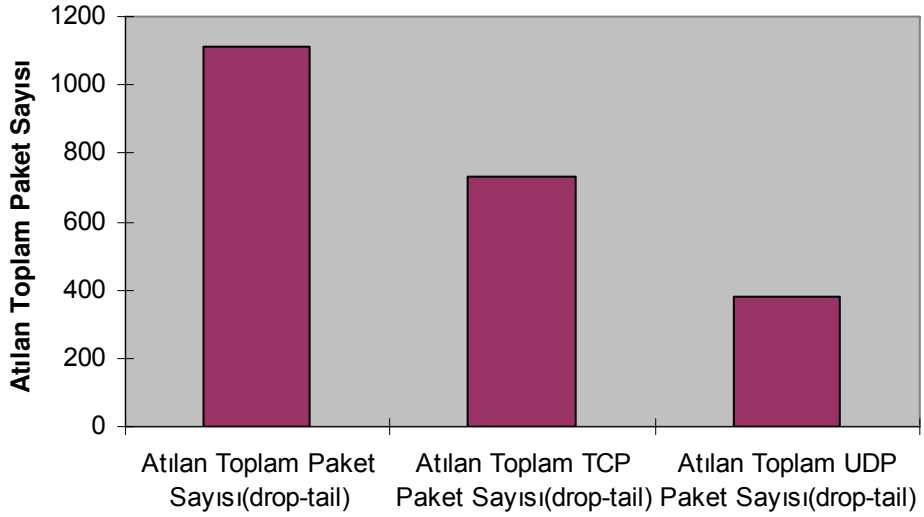
Şekil 4.25. Red kuyruk tipinde düğüm 2'de atılan toplam paketlerin zamana göre dağılımı

Şekil 4.25'te gösterildiği gibi red kuyruk tipinde düğüm 2'de atılan toplam paket sayısı 820 olarak belirlenmiştir. Burada paket atımı az olmasına rağmen atılan TCP paketleri daha fazla olduğu belirlenmiştir. Kuyruk doluluk oranı %100 olarak belirlenmiştir.

Şekil 4.26'da gösterildiği gibi geliştirilen protokolde atılan paket sayısının 2500 olduğu görülmektedir. Bunun sebebi kuyruk doluluk oranının eşik değerinin %80 olarak belirlenmiş olmasıdır. Burada gösterilen grafik benzetimde red kuyruk tipi kullanıldığında elde edilen grafiğe göre daha kötü gibi görünmesine rağmen öncelikli paketlerin iletişime devam etmesi açısından yukarıda verilen atılan paket sayı oranlarına bakıldığında daha iyi olduğu görülmektedir. Şekil 4.27'de drop-tail kuyruk tipinde atılan paket sayılarının oranları gösterilmektedir.



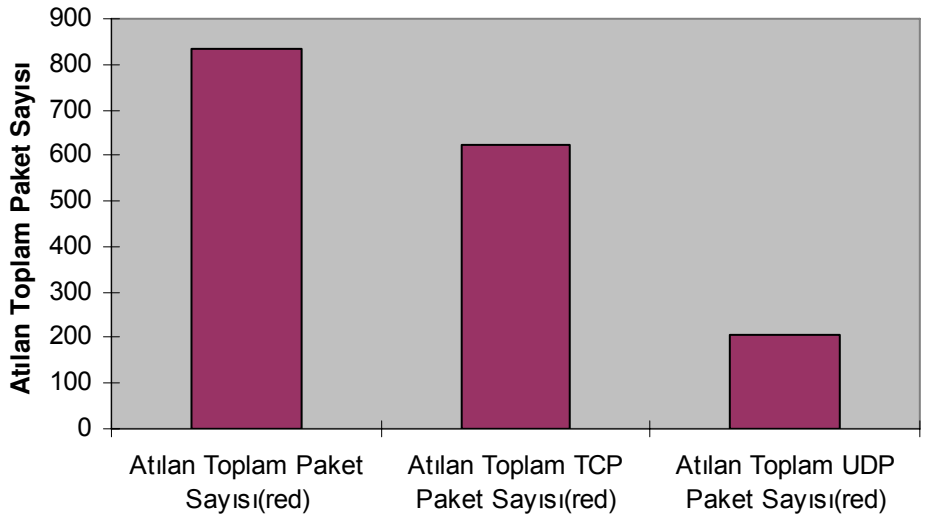
Şekil 4.26. Geliştirilen protokolde düğüm 2'de atılan paketlerin zaman göre dağılımı



Şekil 4.27. Drop-tail kuyruk tipinde atılan paket sayılarının oranları

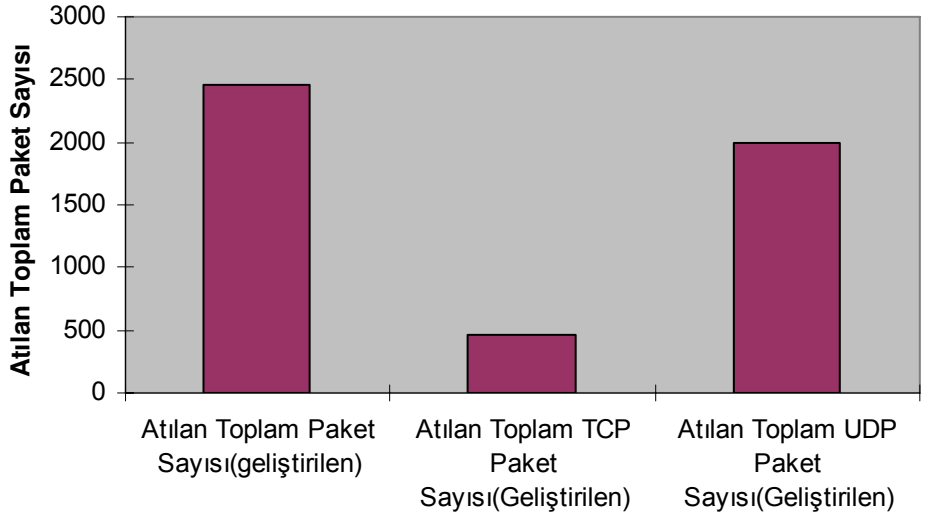
Şekil 4.27'de drop-tail kuyruk yapısında WAP gateway dörevi gören düğüm 2'de atılan toplam paket sayıları ve atılan toplam TCP ve UDP paket sayıları verilmektedir. Düğüm 2'de atılan toplam paket sayısı 1113'tür. Atılan TCP paket sayısı 731, UDP paket sayısı ise 382'dir. Bu benzetimde kuyruk doluluk oranı %100

olarak alınmıştır. Şekil 4.28'de red kuyruk tipinde atılan paket sayılarının oranları gösterilmektedir.



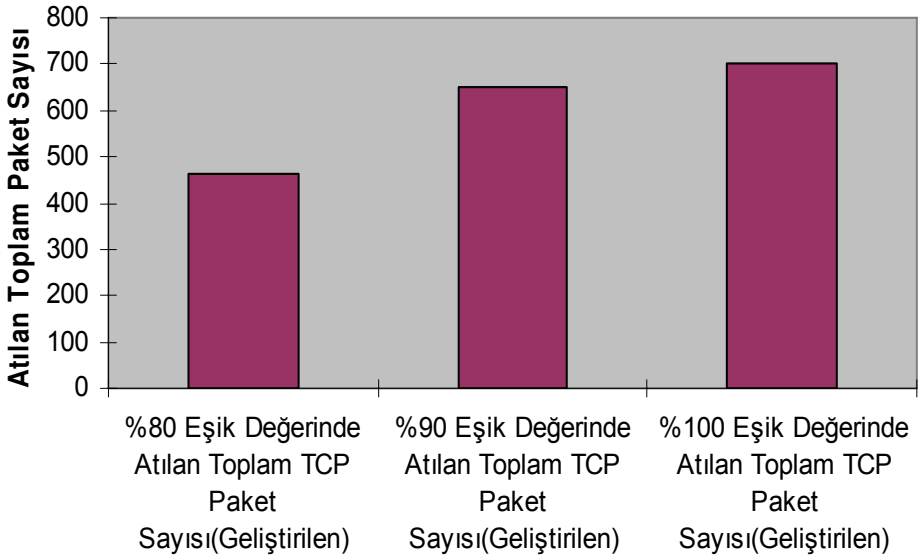
Şekil 4.28. Red kuyruk tipinde atılan paket sayılarının oranları

Şekil 4.28'de benzetimde kullanılan red kuyruk tipinde düğüm 2'de atılan toplam paket sayısının 833, atılan TCP paket sayısı 625, UDP paket sayısı ise 208 olarak elde edilmiştir. Şekil 4.29'da geliştirilen protokolde atılan paket sayılarının oranları verilmiştir.



Şekil 4.29. Geliştirilen protokolde atılan paket sayılarının oranları

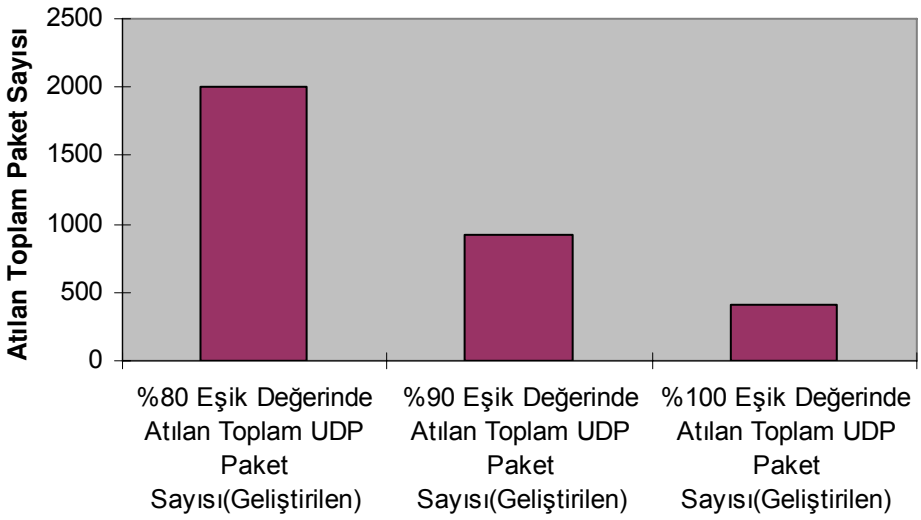
Şekil 4.29'da benzetimde kuyruk tipi olarak geliştirilen protokol uygulanmış ve düğüm 2'de atılan toplam paket sayısının 2465, atılan TCP paket sayısı 463, UDP paket sayısı ise 2002 olarak elde edilmiştir. Geliştirilen protokolün uygulandığı benzetim sonuçlarının diğer kuyruk tiplerine göre daha iyi olduğu görülmektedir. Elde edilen sonuçlara bakıldığında geliştirilen protokolde öncelikli hakka sahip olan TCP paketlerindeki atılma oranı diğer kuyruk tiplerine göre daha başarılıdır. UDP paketlerinde atılma oranının diğer kuyruk tiplerine göre daha fazla olmasının nedeni ise kuyruk doluluk oranının eşik değerinin %80 olarak alınmış olmasından kaynaklanmaktadır. Şekil 4.30'da geliştirilen protokolde kuyruk doluluk oranlarına göre atılan TCP paket sayılarının oranları gösterilmektedir.



Şekil 4.30. Geliştirilen protokolde kuyruk doluluk oranlarına göre atılan TCP paket sayılarının oranları

Şekil 4.30'da geliştirilen protokolün kuyruk doluluk oranlarına göre benzetim işlemi sırasında düğüm 2 tarafından atılan TCP paketleri kuyruk doluluk oranının eşik değerine göre değişmektedir. Kuyruk doluluk oranının eşik değeri %80 olarak alındığında düğüm 2 tarafından atılan toplam paket sayısı 463, %90 olarak belirlendiğinde 650 ve %100 olarak belirlendiğinde ise 700 olarak elde edilmiştir. Bütün benzetimler için eşik değerinin %80 olarak belirlenmesinin sebebi ise gelen TCP paketlerinin atılmasını önlemek için kuyruk limitinde yer bırakmak içindir. Bu TCP paketlerinin iletişimi devam ettirebilmeleri için gerekli görülmüş ve sonuçlarda

da görüldüğü gibi başarılı olunmuştur. Şekil 4.31'de geliştirilen protokolde kuyruk doluluk oranlarına göre atılan UDP paket sayılarının oranları gösterilmektedir.



Şekil 4.31. Geliştirilen protokolde kuyruk doluluk oranlarına göre atılan UDP paket sayılarının oranları

Şekil 4.31'de düğüm 2 tarafından atılan toplam UDP paket sayıları için kuyruk doluluk oranının eşik değeri %80 olarak belirlendiğinde toplam 2002, eşik değeri %90 olarak belirlendiğinde toplam 923 ve eşik değeri %100 olarak belirlendiğinde ise 413 paketin atıldığı görülmüştür. Burada eşik değerinin %80 olarak belirlendiğinde benzetimde fazla paketin atılmasının sebebi düğüm 2'nin kuyruk doluluk oranının eşik değeri %80 olunca paket atmaya başlamasından kaynaklanmaktadır. Eşik değeri %100 olarak belirlendiğinde fazla olmasada geliştirilen protokolün diğer kuyruk tiplerine göre daha başarılı olduğu elde edilen sonuçlarda görülmektedir.

Sonuç olarak elde edilen sonuçlardan da görüldüğü gibi değiştirilen kuyruk yapısı sayesinde tıkanıklık oluşma durumlarının meydana geldiği anda önceliklendirme sayesinde ve önemsiz paketler atılarak tıkanıklık aşılmaktadır. Elde edilen sonuçlarda görüldüğü gibi geliştirilen kuyruk yapısı diğer kuyruk yapılarına göre daha iyi sonuç vermektedir.

5. SONUÇLAR VE ÖNERİLER

Bu çalışmada WAP trafiğinde meydana gelen tıkanıklığın çözümü için bir protokol geliştirilmiştir. Geliştirilen Protokol ns-2 benzetim aracı kullanılarak test edilmiştir. Kuyruk tipleri incelenerek tıkanıklığın kuyruk yapısında değişiklikler ile çözümü amaçlanmış ve başarılı sonuçlar alınmıştır.

Bugüne kadar yapılan çalışmalar genelde donanım üzerinde olmuş ve cihazların kümelenmesi ve yük dağılımı esas alınarak tıkanıklık probleminin çözülmesi amaçlanmıştır. Ancak bu sistem hem pahalı hem de uygulanması zor olan bir yöntemdir. Genele yönelik önceliklendirme parametreleri arttırılarak geliştirilecek bir yöntemin hem uygulanması hem de fiyatı daha uygun olacaktır. Bu çalışmada geliştirilen protokol WAP gateway ile WAP sunucu arasındaki trafiği kontol etmektedir.

Bu çalışmada WAP sunucu ile kullanıcı arasındaki hizmet alışverişi sırasında oluşabilecek tıkanıklığı çözmek için kurulan bağlantılar üzerinde önceliklendirme yapılmıştır. Bu çalışmada TCP ve UDP arasında bir önceliklendirme yapılmıştır. Bu önceliklendirme yapıldıktan sonra kuyruk yapısı, gelen paketlerin öncelik sırasına göre kuyruğa alınmasını sağlamıştır. Bu sayede kuyruk yapısında çalışan ilk giren ilk çıkar mantığı ile önem sırası büyük olan bir isteğin kuyruğa ilk girmesi amaçlanmıştır. Bu sayede genelde önemli iletişimlerin yapıldığı TCP bağlantısında paket kaybının ve gecikmelerin azaltılması buna bağlı olarak tıkanıklığın çözümünde başarılı olunmuştur.

Bu çalışma sonucunda kuyruk tiplerinde yapılacak değişiklikler veya yeni kuyruk tiplerinin oluşturulmasıyla daha verimli bir hizmet verilebileceği görülmüştür. Tıkanık probleminin çözümü için daha kapsamlı çalışmalar yapılarak özellikle ulaşım katmanında ağlar için yeni ve daha kapsamlı kuyruk tiplerinin oluşturulması mümkündür. Her ağda istenilen veya talep edilen hizmetler değişeceğinden bu önceliklendirme genele uyabilecek bir şekilde yapılmasıyla tıkanıklığa genel bir çözüm sağlanabilir.

KAYNAKLAR

1. Arehart C., Chidambaram N., ”Professional WAP”, ***Wrox Press*****,** NewYork, 1861004044 (2000).

2. Toksoy Y., “Kablosuz Uygulama Protokolü Mimarisi”, Yüksek Lisans., ***İTU Fen Bilimleri Enstitüsü***, İstanbul, 1-45 (2002)

3. Lindberg T., Uneman J., “WAP Research”, M.sc., ***Chalmers University of Technology,*** Göteborg , 1-69 (2001).

4. Goldszmidt G., Hunt G., “NetDispatcher: a TCP Connection Router*”, **IBM Research Technical Report,*** 1-32 (1997).

5. İnternet: Kannel Gateway, “Architecture Document”, http://www. Kannel . org/arch.shtml (2003).

6. V. Cardellini, M. Colajanni, P.S. Yu, “Efficient State Estimators for Load Control Policies in Scalable Web Server Clusters”, ***Proceedings of COMPSAC ’98***, 449–455 (1998).

7. Shorten R., King C., Wirth F., Leith D., “Modelling TCP congestion control dynamics in drop-tail environments”, ***Automatica*****,** 43(3): 441-449 (2007).

8. Hur K., Eom Seop D.-, Lee Woo Y.-, Lee Ho J., Kang S., “Priority forwarding for improving the TCP performance in mobile IP based networks with packet buffering”, ***Computer Communications*****,** 30(6): 1337-1349 (2007).

9. Sun J., Zukerman M., “RaQ: A robust active queue management scheme based on rate and queue length”, ***Computer Communications*****,** 30(8): 1731-1741 (2007).

10. Chen S., Bensaou B., “Can high-speed networks survive with DropTail queues management?”, ***Computer Networks*****,** 51(7): 1763-1776 (2007)

11. Guo S., Liao X., Li C., Yang D., “Stability analysis of a novel exponential-RED model with heterogeneous delays”, ***Computer Communications*****,** 30(5): 1058-1074 (2007).

12. Ren F., Lin C., Yin X., “Design a congestion controller based on sliding mode variable structure control”, ***Computer Communications*****,** 28(9): 1050-1061(2005).

13. Şimşek M, “Kablosuz Ağlarda Yönlendirme Protokolü ve Tıkanıklık Denetimi”, Yüksek Lisans, ***Gazi Üniversitesi Fen Bilimleri Enstitüsü*****,** Ankara, 1-10 (2006).

14. Doğru İ.A., "Ad-Hoc Ağlarda Hareketlilik Yönetim Protokolü", Yüksek Lisans., ***Gazi Üniversitesi Fen Bilimleri Enstitüsü*****,** Ankara, 1-10 (2007).

15. İnceoğlu M.M,Kılınç E., "Wap Ağ Geçidinde Ortaya Çıkan Boşlukların kapatılmasına İlişkin Öneriler", ***İnet-tr'03*** **,** İzmir, 1-6 (2003).

16. B.A. Julstrom, D.H. Robinson, "Simulating Exponential Normalization with Weighted k-Tournaments", ***Proceedings of 2000 Congress on Evolutionary Computation*****,** 227–231(2000).

17. İnternet: WAP Forum, "WAP WAE Specification", http://www1. wapforum.org/ what/technical/SPEC-WAESpec-19990524.pdf (1999).

18. İnternet: Linux Virtual Server Project, "Documentation", http://www.linux virtualserver. org/Documents.html, (2003).

19. İnternet: WAP Forum, "WAP WTAI(PDC)", http://www1.wmlclub.com/ docs/especwap1.2/SPEC-WTAIPDC-19991108.pdf (1999).

20. İnternet: WAP Forum, "WTA", http://www1.wapforum.org /tech/terms.asp? doc=WAP-266-WTA-20010908-a.pdf (2001).

21. Sicilia J.M.R, Alonso J.M., "A Transport protocol to improve Access to data networks from GSM using a proxy", ***School of Electrical and Computer Engineering Purdue University***, 1-9 (2000).

22. Hellmund M., " Smart Personalization for Wireless Applications", ***University of Applied Sciences***, 1-100 (2003).

23. Badra M., Serhrouchni A., Urien P., "A lightweight identity authentication protocol for wireless networks", ***Computer Communications***, 27(17): 1738-1745 (2004).

24. İnternet: Free Protocols Foundation , "WAP is a Trap", http://www.freeprotocols. org/content/generated/doc.free/leapForum/leap/manifesto/WapTrap/main.pdf (2000).

25. İnternet: WAP Forum, "WML", http://www1.wapforum. org/tech/ terms.asp ? doc=WAP-191_102-WML-20001213-a.pdf (2000).

26. V. Jaconson, "Congestion Avoidance and Control," ***Proceedings of ACM SIG-COMM 1988*****,** 314-329, (1988).

27. W. R. Stevens, "TCP Slow-Start, Congestion Avoidance, Fast Retransmission, and Fast Recovery Algorithms," ***IETF RFC 2001*****,** Virginia, 3-5 (1997).

28. Keshav S., "Congestion Control in Computer Networks," Phd. in Computer Science, ***University of California at Berkeley*****,** California, 20-24 (1991).

29. K. Shen, T. Yang, L. Chu, "Cluster Load Balancing for Fine-Grain Network Services", ***Proceedings of IPDPS*****,** 493–500 (2002).

30. V. Cardellini,M. Colajanni, P.S. Yu, "Redirection Algorithms for Load Sharing in Distributed Web-server Systems", ***Proceedings of 19th IEEE International Conference on Distributed Computing Systems***, 528–535 (1999).

31. Lin T.H., Wang K., Liu A.Y., "An efficient load balancing strategy for scalable WAP gateways", ***Computer Communications*****,** 28(9): 1028-1037 (2005).

32. Feng W, Shin Kang G., Kandlur Dilip D., Debanjan S., "The BLUE Active Queue Management Algorithms", ***IEEE/ACM Transactions on Networking (TON),*** 10(4): 1-18 (2002).

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: TOKLU, Sinan
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 01.05.1979 Ceyhan
Medeni hali	: Bekâr
Telefon	: 0 (312) 4793820
Faks	: 0 (312) 230 8434
e-mail	: s.toklu@gazi.edu.tr

Eğitim

Derece	Eğitim Birimi	Mezuniyet tarihi
Lisans	Doğu Akdeniz Üniversitesi / Bilgisayar Mühendisliği	2004
Lise	Kurttepe Adana Anadolu Lisesi	1997

İş Deneyimi

Yıl	Yer	Görev
2006-2007	Başbakanlık Özürlüler İdaresi Başkanlığı	Çözümleyici
2007-	Gazi Üniveristesi	Araştırma Görevlisi

Yabancı Dil

İngilizce

Yayınlar

Akcayol M.A., Şimşek, M., Bay, İ., Toklu, S., Doğru, İ.A., "Genetic Algorithm Based Location Optimization of Emergency Service Units", 4th FAE International Symposium, European University of Lefke