PAKET ANAHTARLAMALI AĞLARDA HİZMET KALİTESİ TABANLI ULAŞTIRMA PROTOKOLÜ VE KUYRUK YAPISI GELİŞTİRİLMESİ VE UYGULAMASI

MEHMET ŞİMŞEK

DOKTORA TEZİ
ELEKTRONİK VE BİLGİSAYAR EĞİTİMİ

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

NİSAN 2012
ANKARA

Mehmet Şimşek tarafından hazırlanan PAKET ANAHTARLAMALI AĞLARDA HİZMET KALİTESİ TABANLI ULAŞTIRMA PROTOKOLÜ VE KUYRUK YAPISI GELİŞTİRİLMESİ VE UYGULAMASI adlı bu tezin Doktora tezi olarak uygun olduğunu onaylarım.

Prof. Dr. M. Ali Akcayol
İkinci Tez Yöneticisi

Yrd. Doç. Dr. Nurettin Doğan
Birinci Tez Yöneticisi

Bu çalışma, jürimiz tarafından oy birliği / oy çokluğu ile Elektronik ve Bilgisayar Eğitimi Anabilim Dalında Doktora tezi olarak kabul edilmiştir.

Başkan: : Prof. Dr. Hüseyin Ekiz

Üye : Prof. Dr. Ömer Faruk Bay

Üye : Prof. Dr. İsmail Ertürk

Üye : Yrd. Doç. Dr. Nurettin Doğan

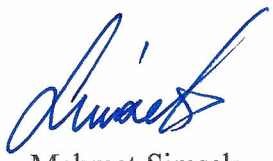
Üye : Yrd. Doç. Dr. Hayrettin Evirgen

Tarih : 13/04/2012

Bu tez, Gazi Üniversitesi Bilişim 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.



Mehmet Şimşek

PAKET ANAHTARLAMALI AĞLARDA HİZMET KALİTESİ TABANLI ULAŞTIRMA PROTOKOLÜ VE KUYRUK YAPISI GELİŞTİRİLMESİ VE UYGULAMASI

(Doktora Tezi)

Mehmet ŞİMŞEK

GAZİ ÜNİVERSİTESİ
BİLİŞİM ENSTİTÜSÜ

Nisan 2012

ÖZET

Bu tezde, Gerçek Zamanlı Çoklu Ortam (GZÇO) verilerinin paket anahtarlamalı ağlarda iletilmesinde yaşanan en büyük sorunlar olan gecikme ve jitter parametrelerini kontrol altında tutan bir ulaştırma katmanı protokolü ve kuyruk modeli geliştirilmiştir. Geliştirilen yöntem Tam Zamanında Ulaştırma (Just In Time Transport - JITT) olarak adlandırılmıştır. JITT, GZÇO verilerini taşıyan paketlerin geçecekleri yönlendiriciler üzerinde zamanlama ve önceliklendirme yapılması temeline dayanmaktadır. Böylece, gecikme ve jitter kontrol altında tutularak sonuçları iyileştirilmiştir. JITT'in ilk olarak matematiksel olarak analizi yapılmış; daha sonra ns2 benzetim aracı ile kapsamlı bir şekilde test edilmiş ve laboratuar ortamında uygulaması gerçekleştirilmiştir. Ayrıca, yaygın olarak kullanılan ulaştırma katmanı protokolleri ve kuyruk modelleri ile JITT karşılaştırılmıştır. Elde edilen sonuçlara göre JITT gecikme, jitter ve paket kayıp oranları parametrelerine göre çok daha başarılı sonuçlar vermiştir.

Bilim Kodu : **702.1.014**
Anahtar Kelime : **gerçek zamanlı iletişim, çoklu ortam, hizmet kalitesi, ulaştırma protokolü, kuyruk modeli, zamanlama**
Sayfa Adedi : **100**
Tez Yöneticisi : **Yrd. Doç. Dr. Nurettin DOĞAN**

DEVELOPMENT AND APPLICATON OF A QUALITY OF SERVICE BASED TRANSPORT LAYER PROTOCOL AND QUEUE MODEL FOR PACKET-SWITCHED NETWORKS

(Ph.D. Thesis)

Mehmet ŞİMŞEK

GAZİ UNIVERSITY
INFORMATICS INSTITUTE

April 2012

ABSTRACT

In this thesis a new method has been developed to control delay and jitter which are the biggest problems to transmission of real-time multimedia data in packet switched networks. The developed method (called Just in Time Transport (JITT)) consists of a transport protocol and a queue model. JITT makes timing and prioritization on the routers. So that, delay and jitter are kept under control and improved. Firstly, JITT behaviors have been analyzed mathematically. After that, JITT has been tested by using the network simulator 2 (ns2) in different simulation model and networking conditions as well as in a real environment and compared to other commonly used transport layer protocols and queue models. According to experimental studies, JITT has given highly successful results in terms of delay, jitter and packet loss ratios.

Science Code : **702.1.014**
Key Words : **real-time communication, multimedia, quality of service, transport protocol, queue model, scheduling**
Page Number : **100**
Adviser : **Assist. Prof. Dr. Nurettin DOĞAN**

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren Hocalarım Yrd. Doç. Dr. Nurettin Doğan ve Prof. Dr. M. Ali Akcayol'a; kıymetli tecrübelerinden faydalandığım Hocalarım Prof. Dr. Ömer Faruk Bay, Prof. Dr. İsmail Ertürk ve Doç. Dr. O. Ayhan Erdem'e; her türlü destekle beni hiçbir zaman yalnız bırakmayan çok değerli eşim Aybike Şimşek'e; neşe ve moral kaynağım oğlum Davut Mete Şimşek'e ve Aileme teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

Şekil **Sayfa**

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

Bu çalışmada kullanılmış bazı simgeler ve kısaltmalar, açıklamaları ile birlikte aşağıda sunulmuştur.

Simgeler	**Açıklama**
t	Süre
$p_s(t)$	t zamanda oluşturulan çoklu ortam verisi miktarı
e(t)	t zamanda kodlayıcı tarafından oluşturulan bit akışının uzunluğunu
$b_s(t)$	t zamanda ağa enjekte edilen paket miktarı
$b_r(t)$	t zamanda paketlerden alınan veri miktarı
r(t)	Paketlerden alınan ve kod çözücü tarafından okunacak olan giriş
d(t)	Kod çözücü tarafından oluşturulan çıkış
$p_r(t)$	t zamanda alıcı tarafından oynatılan veri miktarı
Q	JITT kuyruğu
P_i	Kuyrukta bulunan paketler
τ_i	P_i paketinin MWT değeri
τ_{max}	Maximum τ_i
P_{max}	MWT değeri τ_{max} olan paket
Φ	$Q - P_{max}$
φ	Bir paketin bit cinsinden boyutu
Θ	Φ alt kümesinin her bir elemanının bit cinsinden boyut toplamı
B_{out}	Bir yönlendiricinin çıkış bant genişliği
λ	Bir yönlendiricinin girişindeki bps cinsinden veri oranı
μ	Bir yönlendiricinin bps cinsinden çıkış bant genişliği
σ	Bir yönlendiricideki tam zamanında iletim sınırı içerisindeki boş yer
W	Kuyrukta bekleme süresi ile hizmet süresi toplamı

Simgeler	Açıklama
W_q	Kuyrukta bekleme süresi
ρ	Fayda faktörü
C	Kuyruk kapasitesi
F	Bir JITT akışı
J	Bir JITT akışının jitteri
δ	Paketlerin hattan alınma zamanları farkı
ω	Paketlerin kuyruktan alınma zamanları farkı
η	Bir paketin hizmet süresi
α	Birim zamanda JITT akışlarının ürettiği veri miktarı
b_j	Anlık JITT trafik oranı
B_j	Maksimum JITT trafik oranı
β	Anlık JITT harici trafik oranı
B_d	Maksimum JITT harici trafik oranı

Kısaltmalar	Açıklama
ACM	Hesaplama Makineleri Derneği (Association for Computing Machinery)
AMP	Uyarlanabilir Ortam Oynatma (Adaptive Media Playout)
ARTP	Uyarlanabilir Gerçek-Zamanlı Ulaştırma Protokolü (Adaptive Real-Time Transport Protocol)
CBQ	Sınıf Tabanlı Kuyruklama (Class Based Queueing)
CBR	Sabit Bit Oranı (Constant Bit Rate)
CBT	Sınıf Tabanlı Eşikler (Class-Based Thresholds)
CT	Şimdiki Zaman (Current Time)
DCCM	Dinamik Tıkanıklık Kontrol Mekanizması (Dynamic Congestion Control Mechanism)
DiffServ	Ayrıştırılmış Hizmetler (Differentiated Services)

Kısaltmalar	Açıklama
DRR	Eksik Round-Robin (Deficit Round-Robin)
ET	Geçen Süre (Elapsed Time)
GPS	Genelleştirilmiş İşlemci Paylaşımı (Generalized Processor Sharing)
GZÇO	Gerçek Zamanlı Çoklu Ortam
IBPS	Araya Sokma Tabanlı Paket Zamanlama (Insertion Based Packets Scheduling)
IEEE	Elektrik ve Elektronik Mühendisleri Enstitüsü (Institute of Electrical and Electronics Engineers)
IETF	İnternet Mühendisliği Görev Gücü (Internet Engineering Task Force)
IntServ	Bütünleştirilmiş Hizmetler (Integrated Services)
IP	İnternet Protokolü (Internet Protocol)
ITU	Uluslararası Telekomünikasyon Birliği (International Telecommunication Union)
JITT	Tam Zamanında Ulaştırma (Just In Time Transport)
LLEPS	Düşük Gecikmeli Etkin Paket Zamanlama (Low Latency Efficient Packet Scheduling)
ML-AVSS	Çok Katmanlı Ses ve Görüntü Akış Sistemi (MultiLayer-AudioVisual Streaming System)
MSF-RS	En Az Hizmet İlk Zamanlama (Minimum-Service First Request Scheduling)
MWT	Azami Bekleme Süresi (Maximum Waiting Time)
NIC	Ağ Arayüz Kartı (Network Interface Card)
ns	Ağ Benzetim Aracı (Network Simulator)
OSI	Açık Sistemler Bağlantısı (Open Systems Interconnection)
OTCL	Nesne Aracı Komut Dili (Object Tool Command Language)

Kısaltmalar	Açıklama
PUNSI	Kayıtsız Bilgili Tepkisiz Akışları Cezalandırma (Penalizing UNresponsive Flows with Stateless Information)
QoS	Hizmet Kalitesi (Quality of Service)
RED	Rastgele Erken Tespit (Random Early Detection)
REM	Rastgele Üssel İşaretleme (Random Exponential Marking)
RFC	Açıklama İsteği (Request for Comment)
ROS	Oran Garantili Fırsatçı Zamanlama (Rate-Guaranteed Opportunistic Scheduling)
RTP	Gerçek-Zamanlı Ulaştırma Protokolü (Real-Time Transport Protocol)
RTT	Gidiş Dönüş Zamanı (Round Trip Time)
RTTMON	Gidiş Dönüş Zamanı Gözlemcisi (Round Trip Time Monitor)
SACK	Seçmeli Alındı (Selective Acknowledgment)
SCTP	Akış Kontrol Aktarım Protokolü (Stream Control Transmission Protocol)
SSA	Servis Güvencesi Ajanı (Service Assurance Agent)
SSVP	Ölçeklenebilir Akış Vidyo Protokolü (Scalable Streaming Video Protocol)
TCP	Aktarım Kontrol Protokolü (Transmission Control Protocol)
TSN	Aktarım Sıra Numarası (Transmission Sequence Number)
TTL	Yaşam Süresi (Time to Live)
UDP	Kullanıcı Verikatarı Protokolü (User Datagram Protocol)
VBR	Değişken Bit Oranı (Variable Bit Rate)
WFQ	Ağırlıklı Adil Kuyruk (Weighted Fair Queuing)

1. GİRİŞ

Hizmet kalitesi, International Telecommunication Union (ITU) tarafından şu şekilde tanımlanmaktadır:

"Bir servisin kullanıcısının memnuniyet derecesini belirleyen hizmet performansının toplam etkisi" [1].

Bu bağlamda, bir servisin kullanıcısının memnuniyetini arttırmaya yönelik her türlü çalışma hizmet kalitesinin arttırılmasına yönelik çalışma olarak değerlendirilebilir. Bilgisayar ağları açısından bakılacak olursa hizmet kalitesini belirlemede göz önünde bulundurulacak olan parametreler bant genişliği, paket kayıp oranı, gecikme ve jitter olarak verilebilir [2]. İletim ortamının bant genişliği, fiziksel olarak sağlanan frekans aralığı ve modülasyon teknikleri ile ilgili ve Open Systems Interconnection (OSI) katmanlarından biri olan fiziksel katmanı ilgilendiren bir kavramdır. Paket kayıp oranı, iletim ortamında meydana gelen çakışmalar sonucunda kaybolan paketler ve tıkanıklık durumunda yaşanan paket kayıpları ile ilgili bir kavramdır. Gecikme, bir veri paketinin kaynaktan hedefe varış süresi, jitter (paket gecikme değişimi) hedefe ard arda varan ve aynı akışa ait olan paketlerin gecikmeleri arasındaki fark olarak tanımlanabilir [3,4,5].

Paket anahtarlama fikri ilk olarak 1960'larda ortaya atılmıştır. Paket anahtarlama, sabit veya değişken veri oranına sahip kaynaklar tarafından üretilen verilerin paylaşımlı bir ortam üzerinden gruplar halinde iletilmesi için, grup halinde iletilen verilerin geçtikleri her cihaz üzerinden bir diğerine grup halinde iletilmesi demektir ve grup halinde iletilen veri bloklarına paket ismi verilmektedir. Paket anahtarlamalı ortamlar paylaşımlı ortamlar olduklarından ve paket boyutları da sabit olmadığından, bir kaynaktan çıkan paketlerin aynı sürelerde hedefe ulaşıp ulaşamadıkları GZÇO akışları için önem arz etmektedir. GZÇO iletişim kalitesini arttırmaya yönelik yapılan çalışmaların hedefinde de gecikme ve jitter'in kontrol altında tutulması yer almaktadır.

1990'larda tanımlanan Differentiated Services (DiffServ) ve Intergrated Services (IntServ) paket anahtarlamalı ortamlarda Quality of Service (QoS) bilinçli hizmet vermek için tanımlanmış kavramlardır ve o günden bu yana yapılan hemen hemen bütün çalışmalar bu kategorilerde yer almaktadır [6,7,8].

Bu çalışmada, GZÇO verilerinin paket anahtarlamalı ağlarda hizmet kalitesi tabanlı olarak iletilmesi için bir aktarım protokolü ve kuyruk modelinden oluşan bir mimari geliştirilmiştir. Protokol ve kuyruk yapısı, ns2 benzetim aracı üzerinde kodlanmış, deneysel sonuçlar elde edilmiş, uygulaması gerçekleştirilmiş, matematiksel analizi yapılmış ve ölçeklenebilirliğini göstermek için benzetimler gerçekleştirilmiştir.

2. HİZMET KALİTESİ ÜZERİNE YAPILMIŞ OLAN ÇALIŞMALAR

Bu bölümde paket anahtarlamalı ağlarda hizmet kalitesini arttırmak için yapılmış olan ve literatürde yer alan çalışmalar verilmiştir.

2.1. Ulaştırma Katmanı ve Tamponlama Üzerine Yapılmış Çalışmalar

2000 yılında Stewart R. ve ark. Stream Control Transmission Protocol (SCTP) isimli bir Request For Comment (RFC) dokümanı yayınladılar. Çalışmada, TCP'nin, IP ağlarında güvenilir veri transferi sağladığını ancak yeni gelişen uygulamalarda TCP'nin yetersiz kaldığını ve TCP'nin güvenilir veri transferi özelliğini UDP ile birleştirme ihtiyacının doğduğunu belirtmişlerdir. Kullanıcıların kullanmak istemedikleri TCP özellikleri şu şekilde sıralanabilir [9]:

- TCP, veri paketlerini numaralandırır ve bu paketleri hedef düğümde yeniden sıralar. Bazı uygulamalar güvenilir aktarımla birlikte yeniden sıralama istemeyebilir.
- TCP'nin akış tabanlı doğası bazı uygulamalarda sıkıntı yaratmaktadır. TCP, verileri byte bazında paketlere bölerek aktarır. Uygulamalar, mesajlarına kendi işaretlemelerini koymak isteyebilirler ve bütün mesajın makul bir sürede iletilmesini zorlayabilirler.

SCTP, kayıt tabanlı olarak çalışır. Yani, UDP'de olduğu gibi bir paket içerisinde aktarılan veriler, alıcı tarafta okunmaya hazırdır [4,9]. SCTP her bir mesaja Transmission Sequence Number (TSN) atar. TSN, akışı kontrol eden sıra numaralarından farklıdır. Böylelikle akış kontrolü ile güvenilir aktarım birbirlerinden ayrılmış olur. Geri bildirim mekanizması TSN tabanlı çalışır. Böylece akış verisi değil mesajlar için geri bildirim gönderilmiş olur [9,10].

2005 yılında Argyriou, Media Stream Control Transmission Protocol (Media-SCTP) isimli bir çalışma yapmıştır. Media-SCTP, SCTP'nin bir modifikasyonudur. Eğer

alıcı taraf bir mesajın gelmediğini fark ederse bu durumda aşağıdaki karşılaştırmayı yapar [11].

$$\frac{\mathrm{RTT}}{2} \leq \mathrm{playout}_{\mathrm{time}} - \mathrm{current}_{\mathrm{time}} \tag{2.1}$$

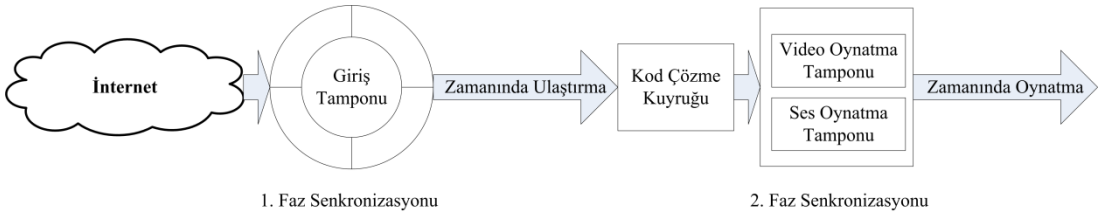
Burada RTT, bağlantının gidiş-dönüş zamanıdır. $\mathrm{playout}_{\mathrm{time}}$, tamponda tutulan frame'lerin oynatma süresi ve $\mathrm{current}_{\mathrm{time}}$ da o anki zamandır. Eğer, alınamayan mesaj için yeteri kadar süre varsa alıcı taraf bir Selective Acknowledgment (SACK) gönderir. SACK'ı alan gönderici, kayıp olan mesajı yeniden gönderir. Böylece, tampondaki frame'lerin oynatılması bitmeden kayıp olan mesaj alıcıya ulaşmış olur. Eğer yeteri kadar süre yoksa kayıp olan mesaj için hiçbir şey yapılmaz [11].

IETF tarafından 1996 yılında Real-Time Transport Protocol (RTP) isimli bir protokol tanımlanmıştır. RTP, ses ve görüntü gibi gerçek zamanlı verilerin uçtan uca aktarılmasını sağlayan bir ulaştırma katmanı protokolüdür. RTP, kaynak rezervasyonu ve hizmet kalitesi sağlamayı garanti etmez. RTP'nin amacı, gerçek zamanlı verilerin hedefe ulaştırılıp ulaştırılmadığının izlenmesi ve raporlanmasıdır. RTP, tek alıcılı sistemlerden büyük çok alıcılı sistemlere kadar ölçeklenebilir bir protokoldür [12,13].

2007 yılında Shafqaat ve ark. A Dynamic Congestion Control Mechanism for Real Time Streams over RTP isimli bir çalışma yapmışlardır. Yazarlar bu çalışmada RTP tabanlı Dynamic Congestion Control Mechanism (DCCM) isimli bir mekanizma geliştirmişlerdir. Çalışmalarının temel amacı sürekli tıkanıklık ile geçici tıkanıklığı birbirinden ayırmak olmuştur. Geliştirdikleri mekanizma, paketlerin varış süreleri arasındaki jitter'i kullanarak geçici tıkanıklığı tespit etmek ve paket kayıp oranı tahmini ile sürekli tıkanıklığı tespit etmek üzerine kurulmuştur. RTP gerçek zamanlı aktarımlar için bazı fonksiyonellikler sağlasa da gerçek zamanlı uygulamalar için QoS garantisi vermemekte ve kaynak rezervasyonu yapmamaktadır [14]. DCCM'de paket kayıp oranının ve jitter'in fonksiyonuna göre gönderici düğüm gönderme oranını ayarlamaktadır. Çalışmada, jitter'in artması, tıkanıklığın habercisi olabileceği

belirtilmiş ve tıkanıklık meydana gelmeden, göndericinin gönderme oranını ayarlayarak tıkanıklıktan kaçınılabileceği önerilmiştir. Sonuç olarak DCCM'nin, diğer birçok protokolden farklı olarak tıkanıklık meydana gelmeden tıkanıklık olacağını tahmin edebildiği ve ağı istikrarlı bir duruma çekebildiği gösterilmiştir [14].

2009 yılında Huang ve ark. A Multilayered Audiovisual Streaming System Using the Network Bandwidth Adaptation and the Two-Phase Synchronization isimli bir çalışma yayınladılar. Yazarlar, yaptıkları bu çalışmada görüntü ve ses verilerinin senkronize olarak aktarılabilmesi için MultiLayer-AudioVisual Streaming System (ML-AVSS) olarak adlandırdıkları bir sistem geliştirmişlerdir. Geliştirdikleri sistem kapsamında ağın bant genişliğini ve farklı kod çözme zaman karmaşıklığı unsurlarını dikkate alan anti jitter prosedürü, koşulsal yeniden iletim mekanizması ve oynatma senkronizasyon mekanizması tasarlamışlardır. Geliştirdikleri mekanizma, alıcı tarafta paketleri tamponlayarak uygulama katmanına senkron bir biçimde aktarılması temeline dayanmaktadır. Geliştirilmiş olan mekanizma Şekil 2. 1'de gösterilmektedir [15].

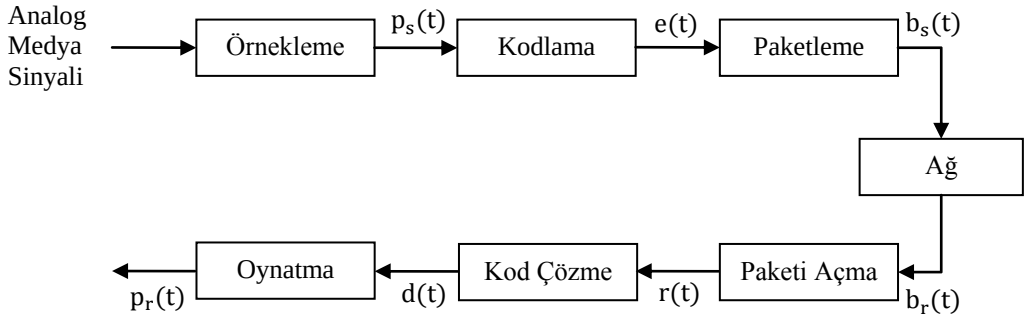


Şekil 2.1. ML-AVSS mekanizmasının gösterimi

Şekil 2.1'de de gösterildiği gibi, alınan paketler tamponlanmakta, gerektiği zamanda kod çözme kuyruğuna aktarılmakta ve daha sonra oynatma tamponlarına iletilmektedir. Böylece, video ve ses paketleri tam zamanında oynatılabilmektedir. Bu çalışmada yazarların geliştirdiği anti jitter prosedürü şu şekilde çalışmaktadır: ard arda alınan görüntü ve ses paketlerinden jitter'de bir değişiklik olduğu tespit edildiği zaman oynatma süresini gecikme süresinden daha uzun bir süreye ayarlamakta ve böylece görüntü ve sesin akıcı olması sağlanmaktadır. Sonuç olarak yazarlar geliştirdikleri sistemin benzetim sonuçlarından yola çıkarak bu sistemin gerçek

zamanlı ses ve görüntü verilerinin daha yumuşak geçişlerle oynatılabildiğini belirtmişlerdir [15].

2002 yılında Kalman ve ark. Adaptive Media Playout (AMP) isimli bir çalışma yayınladılar. Bu çalışmada yazarlar, kullanıcının paketlenmiş medya verilerinin ağ üzerinden aktarılması sonucunda yaşadığı gecikmeleri azaltan bir metot geliştirmişlerdir [16]. Bir medya verisinin göndericiden alıcıya aktarılması işlemi Şekil 2.2'de gösterildiği gibi olmaktadır.



Şekil 2.2. Sayısal medya verilerinin işlenmesi ve iletilmesi

Şekil 2.2'deki $p_s(t)$, t zamanda oluşturulan çoklu ortam verisi miktarını; e(t), t zamanda kodlayıcı tarafından oluşturulan bit akışının uzunluğunu; $b_s(t)$, t zamanda ağa enjekte edilen paket miktarını; $b_r(t)$, t zamanda alınan paketlerdeki veri miktarını; r(t), paketlerden alınan ve kod çözücü tarafından okunacak olan girişi; d(t) kod çözücü tarafından oluşturulan çıkışı ve $p_r(t)$ de t zamanda alıcı tarafından oynatılan veri miktarını tanımlamaktadır [17].

Alıcı taraflı tamponlama

Alıcı tarafın, gönderici taraftan gelen paketleri tamponlaması ve oynatma işlemini bu tampondan yürütmesi jitter'i soğurmak içindir. Tamponlama işlemi aynı zamanda, kaybolan paketlerin yeniden gönderilebilmesi için gönderici tarafa süre sağlamaktadır. Bununla birlikte, tamponlama işlemi başlangıç oynatma süresini arttırmaktadır [17].

Tampon boyutu ile oynatmaya başlama süresi arasında sıkı bir ilişki vardır. Tamponlanan paketlerin miktarı belirli bir seviyeye gelince alıcı taraf verileri oynatmaya başlar. Eğer kullanılan tampon büyük seçilirse gecikmelerden kaynaklanan etkiler yumuşatılmış olur. Ancak, uçtan uca gecikmeler küçük tutulmak isteniyorsa tampon kullanımına dikkat edilmelidir [17].

2004 yılında Huang ve Chang An Adaptive Flow Control with The Re-Transmission Policy Over The Server–Proxy–Client Networking Environment isimli bir çalışma yayınlamışlardır. Bu çalışmada yazarlar server-proxy-client üzerinden multimedia verilerinin aktarılması için bir akış kontrol mekanizması geliştirmişlerdir. Multimedia tipi olarak AVI seçilmiştir. Geliştirdikleri mekanizma, hattın bant genişliğine göre sunum kalitesini ayarlamakta, önemli video frame'leri için yeniden gönderim mekanizması işletmekte ve yumuşak geçişli video oynatma için akış tamponu kontrolü yapmaktadır. Test sonuçlarına göre, geliştirilen yöntemin bant genişliğine göre sunum kalitesini başarılı bir şekilde ayarlayabildiği görülmüştür [18].

2007 yılında Khlifi ve Grégoire ARTP: A Buffer-Aware Rate Control Protocol for Media Streaming isimli bir çalışma yayınlamışlardır. Bu çalışmada yazarlar, tıkanıklık kontrol mekanizması çalıştıran ARTP isimli bir protokol geliştirmişlerdir. Bu mekanizma ile gönderici, alıcı tarafın tampon durumuna göre akış oranını ayarlayabilmektedir. ARTP, medyanın oynatma sürekliliğini sağlamak için tıkanıklık olmadığı zamanlarda paketleri tamponlamaktadır. Yazarların gerçekleştirdiği benzetim sonuçlarına göre ARTP, kayıp oranını azaltmakta ve alıcı tarafın tampon taşmalarını engellemekte başarılı olmuştur [19].

2.2. Temel Kuyruk Yapıları

Ard arda gelen işleri sıraya koymak için kullanılan yapılar kuyruk yapılarıdır. En basit kuyruk yapısı "ilk gelen ilk hizmet alır" mantığı ile oluşturulan kuyruk yapılarıdır. Bilgisayar ağlarında bu mantıkla çalışan kuyruk yapısı drop tail olarak isimlendirilir. Bu kuyruk yapısında önce gelen paket önce yönlendirilir ve eğer

kuyruk dolduysa, en son gelen paket iptal edilir. Günümüzde kullanılan yönlendiricilerde daha karmaşık ve gelen işleri sınıflandıran kuyruk yapıları kullanılmaktadır. Belirli bir hizmet kalitesini sunabilmek için kuyruk üzerinde sınıflandırma, önceliklendirme gibi işlemlerin gerçekleştirilmesi kaçınılmazdır. Bu bölümde, genel kabul görmüş kuyruk yapılarından bahsedilecektir.

2.2.1. Random early detection gateways for congestion avoidance (RED)

1993 yılında Floyd ve Jacobson, paket anahtarlamalı ağlarda tıkanıklıktan kaçınmak için bir kuyruk modeli geliştirmişlerdir. Geliştirdikleri yöntemde ağ geçidi, ortalama kuyruk uzunluğuna bakarak tıkanıklığın başladığına karar verir. Ağ geçidi, tıkanan bağlantıları, gelen paketleri atarak veya paket başlıklarındaki bir bitlik bir alanı değiştirerek uyarır. Ortalama kuyruk uzunluğu, daha önceden belirlenmiş bir değeri geçtiğinde alınan paketler bir olasılık ile işaretlenir. Bu olasılık, ortalama kuyruk uzunluğunun bir fonksiyonudur. RED ağ geçitleri, tıkanıklık kontrolü için bir ulaştırma katmanı protokolü gibi çalışmak üzere tasarlanmışlardır [20].

RED'in drop tail kuyruk yapısına göre farkı, tıkanıklık durumunda birden fazla göndericiden gelen paketleri atarken adaletli davranmasıdır. Drop tail kuyruk yapısı ise aynı göndericiden gelen paketleri atarak hattın adaletli kullanılmamasına neden olmaktadır.

2.2.2. BLUE aktif kuyruk yönetimi algoritması

2002 yılında Feng ve ark. BLUE ismini verdikleri yeni bir kuyruk yönetimi algoritması geliştirmişlerdir. Daha önce geliştirilmiş olan kuyruk yönetimi algoritmalarından farklı olarak BLUE algoritması tıkanıklık tespiti için kuyruk uzunluğunu değil paket atılmaları ve hattın geçmiş kullanım durumu gibi parametreler kullanmaktadır. BLUE, RED'in aksine gelen paketleri farklı olasılıklarla işaretlemez. Yalnızca atılması istenen paketler işaretlenir. Yani, atılma olasılığında ara değerler yoktur. Eğer kuyruk taşmasından dolayı sürekli olarak paketler atılıyorsa, BLUE algoritması gelen paketleri işaretleme olasılığını arttırır.

Tersi durumda, yani kuyruk boş duruma yaklaşıyorsa BLUE algoritması paketleri işaretleme olasılığını azaltır. Yazarların elde ettikleri benzetim sonuçlarına göre BLUE algoritmasının RED algoritmasına göre daha düşük kuyruk boyutlarında paket atılmalarını azalttığı ve hattı daha verimli kullandırdığı görülmüştür [21].

2.2.3. GREEN proaktif kuyruk yönetimi algoritması

2002 yılında Feng ve ark., GREEN ismini verdikleri yeni bir kuyruk yönetimi algoritması geliştirmişlerdir. GREEN, TCP'nin kararlı durumdaki davranışından elde edilen bilgiler ile TCP paketlerinin akıllı ve proaktif bir biçimde atılmasını sağlamaktadır. Böylece, TCP akışları arasında hattı adaletli kullandırma yönünden başarı sağlamaktadır. Yazarların geliştirdikleri algoritma tıkanıklıktan kaçınma yerine tıkanıklığı önleme mantığına dayanmaktadır. Yani, tıkanıklığın tahmin edilmesi değil aktif olarak tespit edilmesi ile algoritma harekete geçmektedir. Bunlara ek olarak GREEN, kuyruk uzunluğunu düşük tutarak jitter'i de azaltmış olur. Bu açıdan GREEN, GZÇO uygulamalarında faydalı sonuçlar sağlamaktadır [22].

2.2.4. REM aktif kuyruk yönetimi algoritması

2001 yılında Athuraliya ve ark., Random Exponential Marking (REM) ismini verdikleri yeni bir kuyruk yönetimi algoritması geliştirmişledir. REM, bant genişliğinin yüksek derecede kullanılmasını ve göz ardı edilebilecek düzeyde paket kayıplarına ve gecikmelere erişilmesini amaçlayan bir algoritmadır. REM'in ardındaki anahtar fikir tıkanıklık ölçümünü performans ölçümünden ayırmaktır. Tıkanıklık ölçümünde, artan bant genişliği isteği ve kullanıcı sayısı; performans ölçümünde, kullanıcı sayısından bağımsız olarak kullanıcıların hedefledikleri performans etrafında stabilize olmaları birer göstergedir. REM'in RED'den farkı, tıkanıklık belirleme yönteminin ve paketleri işaretleme fonksiyonunun farklı olmasıdır. REM'in ilk amacı, giriş oranını bağlantı kapasitesine yakın tutmaktır. REM kullanan her bir çıkış kuyruğu price olarak adlandırılan ve tıkanıklığı ölçmekte kullanılan bir değer tutar. Bu değer, paketleri işaretleme olasılığına karar vermekte

kullanılır. Price değeri giriş oranı ve bağlantı kapasitesi farkına göre periyodik olarak veya asenkron olarak güncellenir. Giriş oranı ve bağlantı kapasitesi farkı pozitif ise price değeri arttırılır, değilse azaltılır [23].

TCP Reno, tıkanıklığı belirlemek için tampon taşması olup olmadığına bakar. TCP Vegas, kuyruktaki gecikmelere bakar ve RED de ortalama kuyruk uzunluğuna bakar. REM ise tıkanıklığı belirlemek için price değerini kullanır. REM paket kaybı, gecikmeler veya kuyruk uzunluğu gibi performans ile ilgili olan parametreleri kullanmadığı için tıkanıklık ile performans ölçütlerini ayırmış olur [23].

Yazarlar, yaptıkları benzetimler sonucunda REM'in bant genişliğini verimli kullandığını ve gecikmeler ile paket kayıplarının göz ardı edilebilir seviyeye çekildiğini göstermişlerdir.

2.3. Kuyruk Yapıları İle İlgili Diğer Çalışmalar

1995 yılında Floyd ve Jacobson Link-sharing and Resource Management Models for Packet Networks isimli bir çalışma yayınladılar. Yazarlar bu çalışmada Class Based Queueing (CBQ) isimli yeni bir kuyruk modeli önermişlerdir. CBQ, ağ geçidi üzerinden akan trafiğin değişik kriterlere göre sınıflandırılması ve bant genişliğinin sınıflar arasında paylaştırılması temeline dayanmaktadır. CBQ, OSI 4. katman protokolleri ve bu protokolleri kullanan servis sınıfları arasında bant genişliğini hiyerarşik olarak paylaştıran bir yapıdır. CBQ, her bir sınıfın ağ geçidi üzerinde kendi kuyruğuna sahip olduğunu varsayar. Hiyerarşik bant genişliği paylaşımı bir öneri olup, sistem yöneticileri tarafından ihtiyaca göre değiştirilebilir [24].

1999 yılında Parris ve ark. Lightweight Active Router-Queue Management for Multimedia Networking isimli bir çalışma yayınladılar. Yazarlar bu çalışmada Class-Based Thresholds (CBT) ismini verdikleri ve RED'in bir uzantısı olan kuyruk yönetimi algoritması geliştirmişlerdir. CBT'nin amacı tıkanıklığı azaltmak ve TCP'yi UDP akışlardan korurken UDP için de kabul edilebilir throughput ve gecikme sağlamaktır. CBT'nin temel mantığı TCP akışlarını UDP akışlarından ayırmak ve

UDP akışlarını da kendi içinde çoklu ortam taşınan UDP akışları ve diğerleri şeklinde ayırmaktır. Yazarlar, CBT'yi denedikleri yönlendiriciler üzerinde bant genişliğini %79 TCP, %16 çoklu ortam taşıyan UDP ve %5 diğer UDP akışları şeklinde paylaştırmışlardır. CBT'de her bir akış kendi sınıfından olan akışlardan bant genişliğini ödünç alabilmektedir. Deneysel sonuçlar sonucunda CBT'nin TCP akışları arasında daha iyi bant genişliği paylaşımı sağladığı ve düşük bant genişliklerinde çoklu ortam taşıyan UDP akışlarından daha az paket atıldığı görülmüştür [25].

2000 yılında Pan ve ark. CHOKe isimli bir kuyruk yönetimi yöntemi geliştirmişlerdir. Yazarlar yaptıkları çalışmada tıkanmış bir yönlendiricinin çıkış hattının n adet akış tarafından adaletli olarak kullanılması problemini ele almışlardır. Geliştirdikleri yöntem, adaletli bant genişliği kullanımına aykırı olarak daha fazla paketi yönlendiriciye gönderen akışlardan paket atılması temeline dayanmaktadır. Yazarlar, TCP gibi akış kontrolü olan protokollerin tıkanıklık durumunda kendilerini ayarlayabildiklerini; buna karşılık, UDP gibi protokollerin tıkanıklık durumunda kendilerini ayarlayamadıklarını ve bu durumun diğer protokolleri kullanan akışlar üzerinde olumsuz etki yarattığını; bant genişliğinin adaletli kullanılmasını sağlamak için yönlendiricilerin, kendilerini ayarlayamayan protokollerden diğer protokolleri koruması gerektiğini belirtmişlerdir. CHOKe algoritmasının çalışması için $\min_{\text{th}}$ ve $\max_{\text{th}}$ isimli iki değişken tanımlanmıştır. Bunlar sırası ile minimum kuyruk uzunluğu eşik değeri ve maksimum kuyruk uzunluğu eşik değeridir. Ayrıca, RED'in yaptığı gibi ortalama kuyruk uzunluğu (AvgQsize) hesaplama mekanizması da CHOKe'a dâhil edilmiştir. Yeni bir paket alındığında $\text{AvgQsize} \leq \min_{\text{th}}$ ise paket kuyruğa alınır; değilse kuyruktan rastgele bir paket alınır ve yeni gelen paket ile aynı akışa ait olup olmadığı kontrol edilir. Eğer aynı ise her iki paket atılır; değilse $\text{AvgQsize} \leq \max_{\text{th}}$ karşılaştırması yapılır. Eğer şart sağlanıyorsa yeni alınan paket için bir olasılık hesaplanır ve bu olasılığa göre kuyruğa alınır veya alınmaz. Şart sağlanmıyorsa yeni alınan paket atılır [26]. Yazarların gerçekleştirdiği benzetim sonuçlarına göre CHOKe algoritması RED ve DropTail kuyruk yapılarına göre akış kontrolü olan protokolleri akış kontrolü olmayan protokollerden korumakta başarı göstermiştir.

2007 yılında Yamaguchi ve Takahashi, Penalizing UNresponsive Flows with Stateless Information (PUNSI) ismini verdikleri adaletli bant genişliği dağıtımı için bir kuyruk yönetimi algoritması önerdiler. PUNSI'nin amacı CHOKe algoritmasının da olduğu gibi akış kontrolü yapan protokollerden akış kontrolü yapmayan protokolleri korumaktır. PUNSI'nin CHOKe'dan farkı, kuyruktan paket seçme işleminin tamamen rastgele değil, bir geometrik dağılım fonksiyonuna göre yapılması ve gelen paketlerin TCP ve UDP olarak ayrılmasıdır. Böylece PUNSI, CHOKe'a göre daha az TCP paketlerinin atılmasını ve tıkanıklık durumuna daha çabuk cevap verilmesini sağlamıştır [27].

2007 yılında Papadimitriou ve Tsaoussidis, gerçek zamanlı video aktarımı için bir tıkanıklık kontrol planı ve Scalable Streaming Video Protocol (SSVP) olarak adlandırdıkları ve UDP üzerinde çalışan bir video akış protokolü geliştirmişlerdir. SSVP, "birer birer arttır, birden fazla azalt" stratejisi ile göndercinin paket gönderme oranını ayarlaması temeline dayanmaktadır. Yazarların gerçekleştirdiği benzetim sonuçlarına göre SSVP, gönderim oranı ve jitter'de iyileştirmeler gerçekleştirmiştir [28].

2.4. Paket Zamanlaması İle İlgili Yapılan Çalışmalar

Hizmet kalitesini arttırmaya yönelik olarak yapılmış çalışmalardan bir başka ana başlık da zamanlayıcı tasarımıdır. Veri akışlarına paket paket hizmet verme fikri ilk olarak Demers, Keshav ve Shenker tarafından ortaya atılmıştır [29]. Yaptıkları çalışma Weighted Fair Queuing (WFQ) olarak adlandırılmaktadır. WFQ, yönlendiriciye gelen akışları sınıflandırarak her birine belirli miktarlarda kaynak ayırma temeline dayanmaktadır.

Aynı mantığı uygulayan bir başka çalışma Parekh ve Gallager tarafından yapılmıştır [30]. Generalized Processor Sharing (GPS) olarak adlandırılan çalışma, başvuru kontrolü yaparak en kötü durum için kabul edilebilir throughput ve gecikme performansı sağlamaktadır. GPS yaklaşımı, veri oranı tabanlı akış kontrolü stratejisi olarak tanımlanabilir. Bu yaklaşımda her bir akış ortalama veri oranı, maksimum veri

oranı gibi bir takım istatistiksel parametreler ile ifade edilir. Eğer ağın kaynakları bu parametreler ile belirlenen hizmet kalitesini sağlayabiliyorsa ağ kaynaklarının gerekli olan kısmı bu akış için ayrılır. GPS, her bir akışa belirli bir veri oranı sağlar.
McKenney tarafından geliştirilen Stochastic Fair Queuing (SFQ) yöntemi, alınan paketlerin hangi kuyruğa dâhil edileceğini tespit etmek için hash tablosu kullanan bir yöntemdir [31]. Her bir paketteki akış ID'sine bakılarak paketlerin hangi kuyruğa dâhil edileceğine karar verilebilir. Bu durumda her bir akış için ayrı bir kuyruk yapısı gereklidir. Bu nedenle McKenney, olası her bir akış için bir kuyruk oluşturmak yerine belirli sayıda kuyruk oluşturulmasını önermiştir. Bu durum hash hesabını kolaylaştırmakla birlikte farklı akışların aynı kuyruğa dâhil edilmesine neden olacağından akışlar arasında bant genişliğini adaletli kullandırmak mümkün olmayabilir. Bu nedenle yöntem stochastic olarak adlandırılmıştır.

Shreedhar ve Varghese tarafından geliştirilen Deficit Round-Robin (DRR) yöntemi round-robin algoritmasından farklı çalışmaktadır [32]. Round-robin algoritmasında her bir kuyruğa verilecek hizmet süresi sabit olarak alınmaktadır. Bu da farklı paket boyutlarına sahip akışlar arasında adaletsizliğe neden olmaktadır. Bu nedenle yazarlar [30]'daki gibi akışları farklı kuyruklara atayarak her bir kuyruğa belirli bir hizmet süresi ayırmışlardır. Eğer bir kuyruktaki bir paketin boyutu o kuyruk için ayrılan hizmet süresinde gönderilemeyecek kadar büyükse, kuyruğa yeniden sıra geldiğinde ilk seferde kullanılmayan hizmet süresi kadar fazla hizmet almaktadır.

Long, Feng ve Tang Rate-Guaranteed Opportunistic Scheduling (ROS) adını verdikleri çalışmalarında bir zamanlama/başvuru kontrolü çalışma çerçevesi geliştirmişlerdir [33]. ROS'un geliştirilme motivasyonu, maksimum thtoughput sağlayarak, kaynak ayırmalarını adaletli bir biçimde yapmak ve düşük algoritma karmaşıklığı sunmaktır. ROS'da yeni bir akış başvurduğu zaman belirli bir miktar tampon bu akışa ayrılır. Bu akışa ait olan paketler bir kuyruğa alınır ve ROS zamanlayıcısının seçimi ile kuyruğun başındaki paket gönderilir. Bu yöntemde kuyruk boyutunun düşük tutulması gecikmeleri düşürmesine karşın paket kayıp oranını arttırmaktadır.

Wu, Hsieh ve Lai geliştirdikleri Low Latency Efficient Packet Scheduling (LLEPS) çalışması ile kararlı duruma sahip olmayan kuyruklarda bant genişliği garantisi sağlayan bir algoritma geliştirmişlerdir [34]. LLEPS, paket kuyruğunun ve akışların davranışlarını takip ederek düşük gecikme sağlayan bir zamanlama algoritmasıdır. LLEPS, her bir akış için bant genişliği ayırarak akışların yalnızca kendilerine ayrılan bant genişliğini kullanmalarına izin verir.

Pyung, Song ve Lee yaptıkları çalışmada en erken zaman aşımına uğrayan paketi ilk gönder stratejisi kullanarak, önceden atanan değerlere göre sınıflandırdıkları akışlar arasında bir zamanlama ve bant genişliği paylaştırma mekanizması geliştirmişlerdir [35]. Geliştirdikleri yöntem ağın kaynaklarını maksimum şekilde kullanmaya olanak tanıyan bir yöntem olarak tanımlanabilir.

Lim, Lee ve Yeo yaptıkları çalışmada uç yönlendiriciler arasında aynı hizmet kalitesi gereksinimi duyan akışlar için QoS alanı geliştirmişlerdir [36]. Uç yönlendiricilere gelen akışlar zaman-kritik, jitter-hassas ve oran-tabanlı olarak sınıflandırılmış ve farklı kuyruklara alınmıştır. Her bir kuyruk için tahsis edilen belirli sürelerde kuyruklardaki paketler yönlendirilmektedir.

Yi ve Kim yaptıkları çalışmada çerçeve tabanlı bir paket zamanlama mekanizması geliştirmişlerdir [37]. Geliştirdikleri yöntem zaman çizgisini çerçeve adı verilen alt zaman aralıklarına bölerek çerçeve sayısı kadar, gerçek zamanlı paketlerin tutulduğu kuyruk yapısı oluşturma ve her bir alt çerçeve zamanı kadar bu kuyruklardan paket gönderme temeline dayanmaktadır. Yazarlar, basit olması nedeni ile ağdaki bütün paketlerin aynı boyutta olduklarını varsaymışlardır.

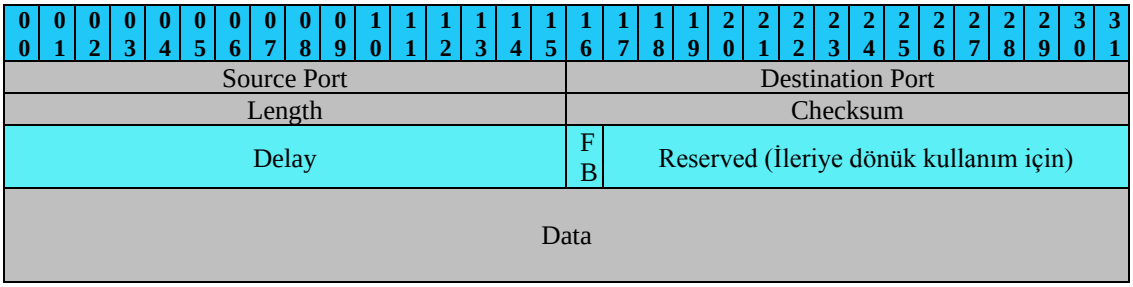
Xie ve ark., anahtarlama cihazlarında QoS garantisi sağlayan ve Insertion Based Packets Scheduling (IBPS) olarak adlandırılan bir yöntem geliştirmişlerdir [38]. Bu yöntem, gerçek zamanlı paketlerin gerçek zamanlı olmayan paketler içerisine yerleştirilmesi temeline dayanmaktadır. Böylece, nispeten küçük olan gerçek zamanlı paketler, nispeten büyük olan gerçek zamanlı olmayan paketlerin iletilmesini beklemeden onlarla birlikte iletilebilmektedir.

Tsao ve ark., kullanıcı tarafından daha önceden ağırlıklandırılarak önceliklendirilmiş sınıflardan en az hizmeti almış olan sınıfa hizmet verme mantığına dayalı Minimun-Service First Request Scheduling (MSF-RS) algoritmasını geliştirmişlerdir [39]. Böylece üst sınıf kullanıcılara daha yüksek bant genişliği sağlanmasını garanti etmişlerdir.

Yukarıda bahsedilen bütün çalışmalar, GZÇO paketlerinin geçtikleri yol üzerindeki yönlendiricileri bireysel olarak ele almış ve her bir yönlendirici üzerinde bağımsız olarak zamanlama ve önceliklendirme yapılmasını önermişlerdir. Bu tez kapsamında ise, GZÇO paketlerinin geçtikleri yol üzerindeki bütün yönlendiricilerin tek yönlendiriciymiş ve tek kuyruk kullanıyorlarmış gibi organize edilmesi önerilmiştir. Bunun gerçekleştirilebilmesi için bir GZÇO paketinin yol üzerindeki bir önceki yönlendirici üzerinde ne kadar süre bekletildiğinin bir sonraki yönlendirici tarafından bilinmesi geremektedir. Bu amaçla, UDP paket yapısının veri kısmına ek olarak iki alan daha eklenerek yeni bir paket yapısı ve ulaştırma protokolü oluşturulmuştur. Yeni eklenen bu alanlar ile iletişim yolu üzerindeki yönlendiricilerin bir GZÇO paketinin yol üzerindeki kendilerinden önceki yönlendiricilerde ne kadar süre beklediği bilgisini edinmeleri ve buna göre zamanlama yapmaları sağlanmıştır.

3. GELİŞTİRİLEN ULAŞTIRMA PROTOKOLÜ VE KUYRUK MODELİNİN (JITT) ÇALIŞMA PRENSİBİ

JITT protokolünün paket yapısı, GZÇO verilerine üç alanın daha eklenmesi ve UDP paketine veri olarak kapsüllenmesi ile elde edilir. Sonuçta ağ üzerinde UDP paketleri taşınmaktadır. JITT protokolün paket yapısı Şekil 3.1'de gösterildiği gibidir.



Şekil 3.1. JITT protokolünün paket yapısı

JITT protokolünün paket yapısı ile UDP paket yapısı FeedBack (FB), Reserved ve Delay alanları dışında aynıdır. Delay alanı ile istenen maksimum uçtan uca gecikme milisaniye cinsinden verilebilmektedir. JITT paketleri bir yönlendirici tarafından alındıkları zaman bu paketlerle birlikte Time Stamp ve Maximum Waiting Time (MWT) isimlerinde iki değişkenle birlikte bir yapı oluşturularak kuyruğa alınmaktadırlar. Time Stamp alanı, paketi alan yönlendiricinin bu paketi ne zaman aldığını saklayabileceği bir alandır. MWT alanı ise bu paketin yönlendirici kuyruğunda ne kadar süre bekleme hakkı bulunduğunu belirten bir alandır.

Paketin yaşam süresi ile ilgili olarak IPv4'te Time to Live (TTL) alanı bulunmakta fakat bu alan IPv6'da bulunmamaktadır. IPv6'nın detaylı olarak anlatıldığı RFC 2460 dokümanında şu ifadeler yer almaktadır:

"IPv4'ün aksine IPv6 bir paketin maksimum yaşamam süresini belirlemeyi zorlamamaktadır. Bunun nedeni, IPv4 başlığında bulunan Time To Live – TTL alanının IPv46'da Hop Limit olarak değiştirilmesidir. Pratikte, çok az IPv4 uygulaması paketlerin yaşam süresini kısıtlamaya ihtiyaç duyar. Böylece, pratikte değişen pek bir şey olmaz. İnternet katmanının üzerinde bulunan herhangi bir protokol, kendi oluşturacağı bir mekanizma ile kendi paketlerinin yaşam süresini belirleyebilir." [40].

İnternetin gelecekteki protokolü olan IPv6'nın standart dokümanında, paketlerin yaşam sürelerinin belirlenme işinin üst katman protokollerce yapılması gerektiği belirtilmiştir. Günümüzde en çok kullanılan ulaştırma katmanı protokolleri olan TCP ve UDP'nin böyle bir mekanizması bulunmamaktadır. JITT protokolün paket başlığındaki Delay alanı sayesinde paketler için istenen maksimum uçtan uca gecikme süresi belirlenebileceği gibi paketlerin yaşam süreleri de belirlenebilir. Bu özelliği ile JITT protokolü, kendi zamanlama mekanizması olan ulaştırma katmanı protokollerinin geliştirilmesine öncülük edebilir.

3.1. Geliştirilen Ulaştırma Protokolü ve Kuyruk Modelinin Algoritması

Genel olarak JITT'in işleyişi şu şekildedir:

Kaynak düğüm, göndereceği JITT paketlerinin Delay alanına istediği gecikme süresi bilgisini yazar. Bir JITT paketini alan yönlendirici Eş. 3.1 ile bu paketin kendi tamponunda en fazla ne kadar süre bekleyebileceğini hesaplar.

$$\mathrm{MWT} = \mathrm{Delay}/\mathrm{hopcount} \tag{3.1}$$

Burada MWT paketin yönlendirici üzerinde en fazla bekleyebileceği süre, Delay JITT paketindeki Delay alanından alınan süre ve hopcount da paketin hedefine o yönlendiriciden kaç adımda ulaşıldığı bilgisidir. MWT değerinin belirlenmesinin ardından JITT paketi ile MWT ve Time Stamp değişkenlerinden bir yapı oluşturulur ve JITT kuyruğuna alınır. Bu esnada alınan JITT paketleri haricindeki paketlerin tamamı da diğer bir kuyruğa eklenir.

Paketlerin sınıflandırılması işlemine paralel olarak, MWT değeri kadar kuyrukta beklemiş olan JITT paketlerinin ve diğer paketlerin iletilmesi için ayrı bir iş parçacığı sürekli olarak JITT kuyruğunu tarar. Bir JITT paketinin gönderilme zamanının geldiğini anlamak için o paketin kuyrukta ne kadar süre beklediğinin hesaplanması gerekmektedir. Eş. 3.2 ile bir JITT paketinin kuyrukta ne kadar süre beklediği hesaplanabilir.

$$ET = CT - TIMESTAMP \quad (3.2)$$

Burada Elapsed Time (ET) JITT paketinin kuyrukta bekleme süresi, Current Time (CT) yönlendiricinin anlık zamanı ve TIMESTAMP de JITT paketi ile yapı oluşturan Time Stamp değişkenidir.

JITT paketlerinin bulunduğu kuyruğun taranması esnasında bir paket için ET değeri hesaplandıktan sonra bu paketin iletilmesi gerekip gerekmediğine karar verilmesi şu şekilde yapılacaktır:

```
If (other queue == empty) or (ET >= MWT - ServiceTime)then
        update Delay Field on JITT packet as
        Delay = Delay - (ET+Servicetime)
        send JITT packet
else send the first packet in the other queue
```

Yukarıdaki karar verme işleminde kullanılan Servicetime değeri milisaniye cinsinden süredir. Bu değer her bir paket için ayrı ayrı hesaplanmakta olup $\frac{\lambda}{B}$ olarak ifade edilir. Burada λ paketin bit cinsinden boyutu, B yönlendiricinin çıkış bant genişliğidir. Bir JITT paketinin gönderilmesine karar verildikten sonra paket üzerinde bulunan Delay alanının güncellenmesi gerekmektedir ki sonraki yönlendiriciler de bu paketin kendi kuyruklarında ne kadar süre bekleyebileceğini hesaplayabilsinler. Bu güncelleme işlemi için Eş. 3.3 kullanılır.

$$Delay = Delay - (ET + ServiceTime) \quad (3.3)$$

Yapılan hesaplamalar sonucunda, iletilmesi gereken bir JITT paket bulunamamışsa diğer kuyruktaki ilk paket iletilir ve bu şekilde silsile yoluyla paketler hedef düğüme iletilmiş olur. Hedef düğüm tarafından alınan bir JITT paketindeki Delay alanının matematiksel gösterimi Eş. 3.4'teki gibidir.

$$LRJPD = OD - \sum_{i=1}^{n}(ET_i + ServiceTime) \quad (3.4)$$

Burada Last Received JITT Packet's Delay (LRJPD), son alınan JITT paketindeki Delay değerini; Original Delay (OD), kaynak düğümün JITT paketine yazdığı Delay değerini; n, paketin hedefe ulaşana kadar geçtiği yönlendirici sayısını ifade etmektedir.

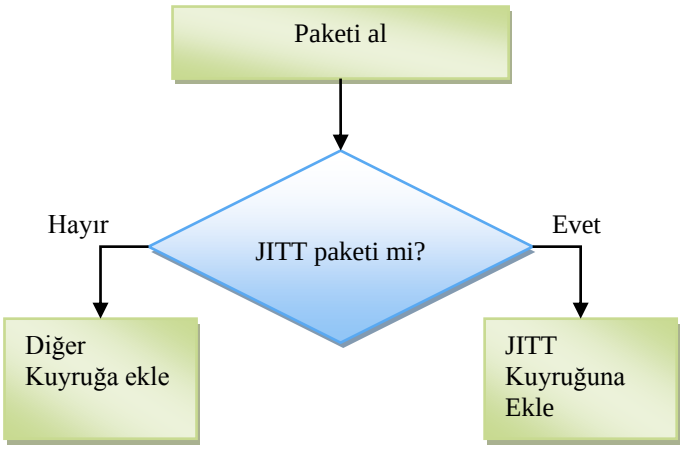
Paketin hedef düğüme iletilmesi ile birlikte karşılaşılabilecek iki durum vardır. İlk durum: eğer kaynak düğüm, Delay alanını JITT paketinin alıcısına varabileceği en küçük süreden daha düşük tutarsa veya zaman içerisinde ağın durumundan dolayı JITT paketinin alıcısına varabileceği en küçük süre Delay alanında belirlenen süreyi geçerse ne olur? Bu durumda, Delay alanı her bir yönlendirici üzerinde Eş. 3.3'e göre güncellendiği için JITT paketi alıcısına vardığı zaman Delay alanı negatif bir değere sahip olacaktır. Bu da, kaynak düğümün Delay alanında belirttiği sürede JITT paketinin iletilmesinin mümkün olmadığının ve paketlerin geçtiği yönlendiricilerden birinde veya birkaçında tıkanıklık durumunun meydana geldiğinin bir göstergesi olabilir. Diğer durum ise yukarıda anlatılanın tersi durumdur. Yani, kaynak düğüm Delay alanını, JITT paketinin alıcısına varabileceği en küçük süreden çok daha büyük tutarsa veya zaman içerisinde ağın durumundan dolayı JITT paketinin alıcısına varabileceği en küçük süre Delay alanında belirlenen süreden çok küçük olursa ne olur? Bu durumda da, JITT paketi alıcısına vardığı zaman Delay alanı büyük pozitif bir değere sahip olacaktır.

Yukarıda anlatılan her iki durumda da JITT'i ulaştırma katmanı protokolü olarak kullanan uygulamalar hattın durumu ile ilgili bilgi sahibi olmuş olurlar. Böylece, alıcının, kaynağı bilgilendirerek Delay alanını yeniden ayarlayabilmesine olanak tanıyan bir mekanizma oluşturulmuş olur. Ancak, ağın durumuna göre, hedefe varan JITT paketlerindeki Delay alanı sürekli olarak salınım gösterecebilir. Bu durumda, hedef düğümün sürekli olarak kaynak düğüme bildirimde bulunmaması için belirli bir zaman aralığında alınan JITT paketlerinin Delay alanlarından okunan değer sürekli olarak negatif ise geri bildirimde bulunulmalıdır. Geri bildirim paketinde FB alanı set edilmeli ve data kısmına LRJPD değeri yazılmalıdır. Böylece, kaynak düğüm bu geri bildirime göre veri oranını arttırabilir veya azaltabilir.

Yukarıda, JITT'in nasıl çalışacağı sözel olarak anlatılmıştır. Takip eden başlıklarda JITT'in çalışma detayları görsellerle anlatılacaktır.

3.1.1. Paketlerin sınıflandırılması

JITT paketlerinin ayrı bir kuyruk yapısına eklenmesi için JITT paketleri ile diğer paketleri ayırt etmek gerekmektedir. Bunun birden fazla yolu vardır. Birincisi, IP paket başlığındaki "Protocol" alanına, hali hazırda hiçbir protokol için kullanılmayan bir numaranın yazılmasıdır. Diğeri ise belirli UDP portlarının kullanılmasıdır. Şekil 3.2'de sınıflandırma işlemi gösterilmektedir.



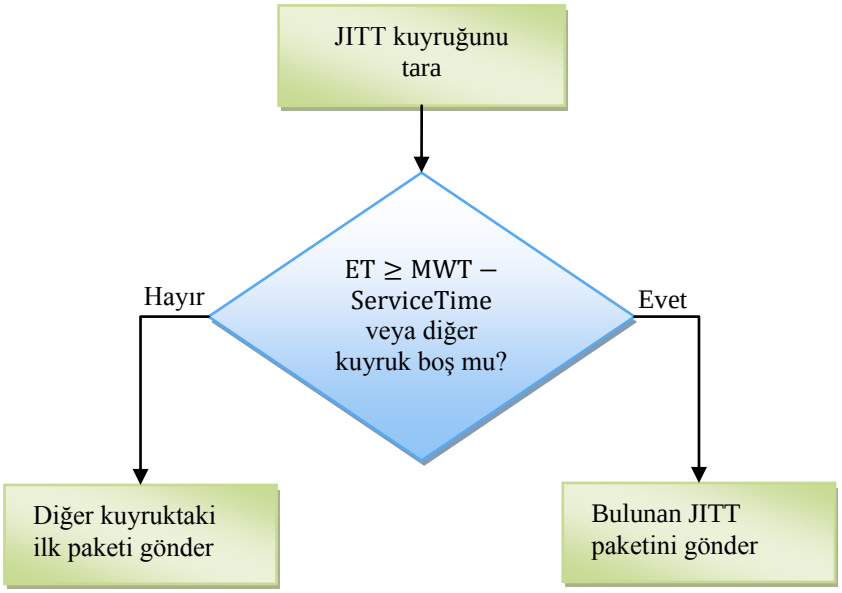
Şekil 3.2. Paketleri sınıflandıran iş parçacığı

JITT Kuyruğuna ekleme işleminin detayları

1. Paketten Delay ve hedef bilgilerini al
2. Hedefe kaç adımda gidileceğini (hopcount) bul
3. Delay/hopcount hesapla. (paketin o düğümde ne kadar süre bekleme hakkı olduğunu gösteren değer, MWT)
4. Paketi, zaman damgası ve MWT değeri ile JITT kuyruğuna al

3.1.2. Kuyruğun taranması

Gönderilme zamanı gelmiş olan bir JITT paketinin olup olmadığı JITT kuyruğu taranarak tespit edilir. Tarama işlemi yerine, JITT kuyruğundaki paketler $ET - (MWT - ServiceTime)$ değerlerine göre sıralanabilirler. Bu durumda kuyruktaki ilk paket ilk gönderilmesi gereken paket olacağından tarama işlemine gerek kalmayacaktır. Ancak bu durum da, sıralama için işlem yükü getirmektedir.

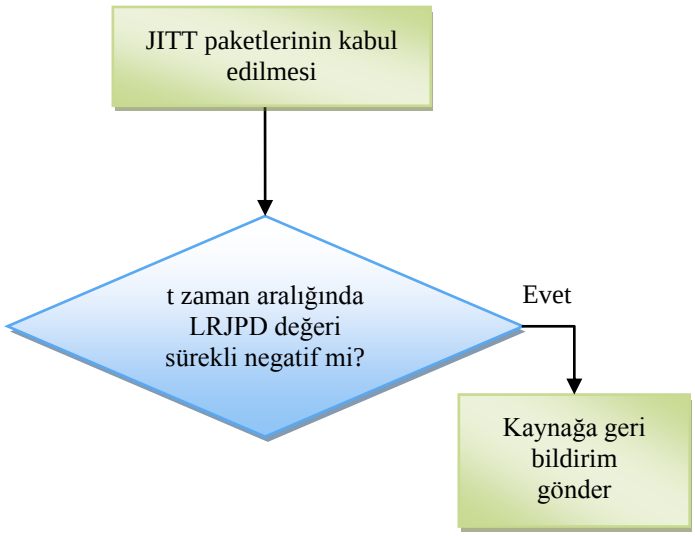


Şekil 3.3. JITT kuyruğunu tarayan iş parçacığı

JITT Kuyruğunu tarama işleminin detayları

1. Kuyruk bir iş parçacığı tarafından sürekli olarak taranır
2. Herhangi bir JITT paketi için $ET \geq MWT - ServiceTime$ koşulu sağlanmıyorsa diğer kuyruktaki bir paketi gönder
3. Herhangi bir JITT paketi için $ET \geq MWT - ServiceTime$ koşulu sağlanıyorsa veya diğer kuyruk boşsa
   a. Paketin Delay alanını Eş. 3.3'e göre güncelle
   b. Paketi gönder

3.1.3. Hedef düğümün geri bildirim göndermesi



Şekil 3.4. JITT alıcısının geri bildirim göndermesi

Geri bildirim gönderme işleminin detayları

1. Hedef düğüm, t zaman aralığında aldığı JITT paketlerinden Delay değerini alır.
2. Alınan her bir paket için LRJPD $<$ 0 koşulu sağlanıyorsa kaynak düğüme geri bildirim gönderir

3.1.4. Geri bildirim alan kaynak düğümün davranışı

Şekil 3.5. JITT kaynağının Delay alanını güncellemesi

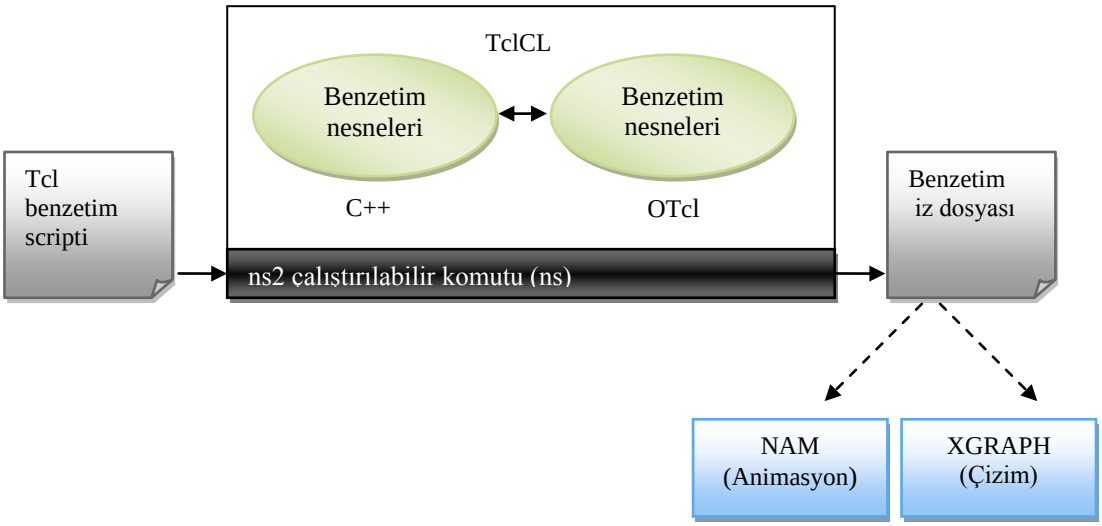
"İlgili akış için ayarlama yap" işleminin detayları

1. Eğer alınan paket geri bildirim paketi ise veri oranını düşür ve/veya Delay değerini arttır.

3.2. Geliştirilen Ulaştırma Protokolü ve Kuyruk Modelinin Kodlandığı Benzetim Ortamının Özellikleri

Gerçekleştirilen benzetimlerden bahsetmeden önce ns2 benzetim aracı ile ilgili bilgi vermek faydalı olacaktır. ns2 açık kaynak kodlu olup, kablosuz ağlar, kablolu ağlar ve algılayıcı ağlar için destek vermektedir. Betik dil olarak Object Tool Command Language (OTCL) kullanan, C++ dilinde yazılmış nesne tabanlı bir araçtır. ns2 her olayı benzetim süresince iz formatında rapor eder. Bu izler daha sonra istenen istatistikleri elde etmek için analiz edilebilir. ns2, içerisinde 458 C++ dosyası, 539 C++ başlık dosyası, 755 TCL dosyası barındırmaktadır (2.27 sürümü için). Institute of Electrical and Electronics Engineers (IEEE) ve Association for Computing Machinery (ACM) yayınlarında yayınlanan ve ağ benzetim aracı kullanılan

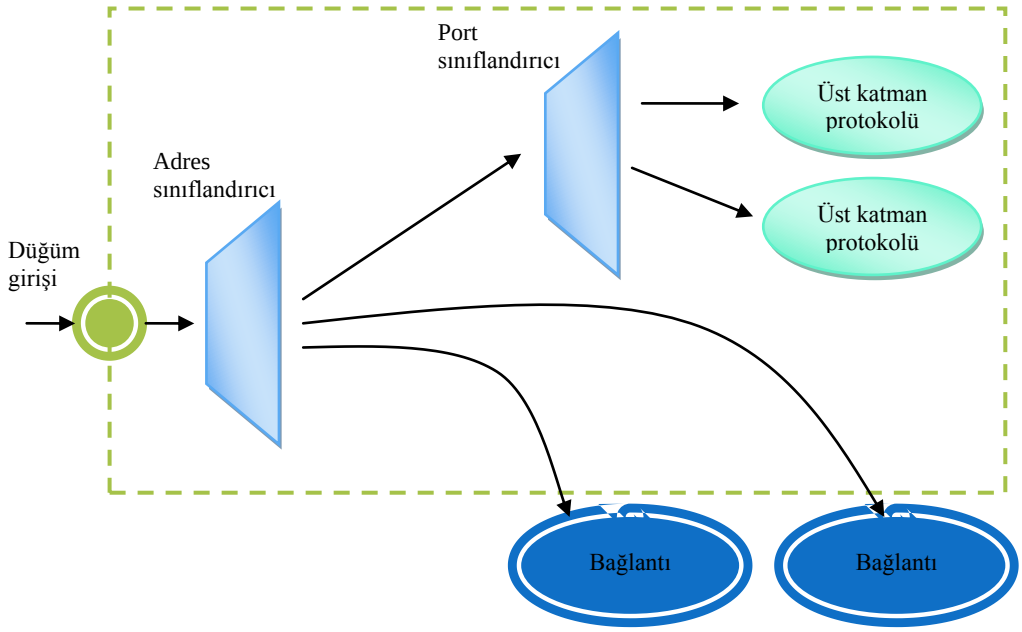
yayınların yarısından fazlasında ns2 kullanılmıştır [41]. ns2 ile bir benzetim gerçekleştirilmesi Şekil 3.6'da gösterildiği gibidir [42].



Şekil 3.6. ns2 ile benzetim

Şekil 3.6'da da gösterildiği gibi Tcl dili ile yazılan benzetim scripti "ns" komutu ile çalıştırılmaktadır. Bunun ardından, ilgili nesneler yaratılmakta ve Otcl tarafında bu nesnelerin gölgeleri oluşturulmaktadır. Sonrasında ise benzetim çalıştırılmakta ve çıktı dosyaları (animasyon ve iz) oluşturulmaktadır.

Bir ağ benzetiminin en önemli bileşeni düğümlerdir. Düğümler, veri gönderme-alma ve yönlendirme yapabilen nesnelerdir. ns2 içerisindeki bir unicast düğüm Şekil 3.7'de gösterilmektedir [43]. Şekil 3.7'de görüldüğü gibi bir düğüm adres sınıflandırıcısı, port sınıflandırıcısı ve protokol ajanlarından oluşmaktadır. Düğüm tarafından alınan paketler ilk olarak adres sınıflandırıcısına girmektedir. Paketin hedef adresi bu düğümün adresi ile aynı ise paket port sınıflandırıcısına gönderilmekte, aynı değil ise yönlendirilmek üzere ilgili bağlantıya gönderilmektedir. Port sınıflandırıcısı gelen paketin hedef port adresini kontrol ederek bu port numarası ile ilişkilendirilmiş bir protokol ajanı olup olmadığını kontrol etmekte ve ilgili bir protokol ajanı varsa paketi bu protokol ajanına aktarmaktadır.



Şekil 3.7. ns2'de unicast düğüm yapısı

3.3. Geliştirilen Ulaştırma Protokolü ve Kuyruk Modeli İçin Gerçekleştirilen Benzetim

Önceki başlıklarda çalışma prensibi anlatılan JITT protokolü ve kuyruk yapısının ns2 ortamına aktarılması için ns2 içerisine yeni sınıflar eklenmiş ve var olan bazı sınıflar değiştirilmiştir. Öncelikle JITT protokolünün nasıl kodlandığı anlatılacaktır.

JITT protokolünün paket yapısı, paket başlığındaki 3 alan dışında UDP ile benzer yapıya sahiptir. Bu nedenle ns2 içerisinde var olan UDP sınıfı ve bu sınıfın kullandığı Packet sınıflarında değişiklikler yapılmıştır. Öncelikle Packet.h dosyası içerisinde ilgili yere takip eden satırlar eklenerek başlıkta yeni alanlar açılmıştır.

```
double  JITT_Delay_;
char    JITT_FB_;
```

Daha sonra aynı değişkenler farklı isimlerde UDP.h dosyası içerisinde aşağıdaki şekilde tanımlanmıştır.

```
double  JITTdelay_;
char    JITTFB_;
int     JITT_TTL;
```

Yukarıda tanımlanan JITT_TTL değişkeni paket başlığında bulunmamakta; yalnızca kullanıcının IP başlığındaki TTL alanını istediği gibi belirlemesine aracı olmaktadır. Daha sonra UDP.cc dosyası içerisinde ilgili yerlere aşağıdaki satırlar eklenerek C++ değişkenlerinin Tcl değişkenleri ile bağlanması sağlanmıştır. Bu şekilde kullanıcı benzetim scripti içerisinden bu değişkenlere değer atayabilmektedir.

```
bind("JITT_delay_", &JITTdelay_);
bind("JITT_TTL_", &JITT_TTL);
```

Son olarak UDP. cc dosyası içerisinde paket oluşturma bloğunun içerisinde aşağıdaki satırlar ile ilgili başlık alanları belirlenmektedir.

```
hdr_cmn::access(p)->JITT_delay() = JITTdelay_;
hdr_ip::access(p)->ttl_ = JITT_TTL;
```

UDP.cc dosyası UDP paketlerinin oluşturulduğu yer olduğu için Time_stamp ve MWT değişkenlerine JITTdelay_ değeri atanmıştır. Paket başlığındaki bu alanlar JITT çalışma prensibi gereği paketi yönlendiren her bir düğüm tarafından gerektiği gibi güncellenecektir. Bu şekilde ilgili dosyalar derlenerek UDP'den JITT türetilmiş olmaktadır.

İkinci olarak JITT kuyruk yapısı kodlanmıştır. ns2 içerisinde yer alan ve ilk giren ilk çıkar mantığı ile çalışan kuyruk yapısı olan Drop Tail kuyruk yapısı kullanılarak JITT kuyruk yapısı türetilmiştir. Drop Tail'i modifiye etme yerine bağımsız ".cc" ve ".h" dosyaları oluşturularak kuyruk yapısı tanımlanmıştır. JITT kuyruk yapısında biri JITT paketleri için diğeri ise diğer paketler için kullanılan iki adet kuyruk bulunmaktadır. Oluşturulan JITT-queue.h dosyası içerisinde bu iki kuyruk yapısı şu şekilde tanımlanmıştır:

```
PacketQueue *q_;
PacketQueue *q_jitt_;
```

Kuyruk yapısının esas iş yaptığı kısım JITT-queue.cc dosyasında yer almaktadır. Gelen paketlerin JITT ve diğer şeklinde ayrılması işlemi aşağıdaki kod ile yapılmaktadır:

```
if (hdr_cmn::access(p)->ptype()!= 2)
```

Buradaki 2 değeri bu paketin bir CBR (Constant Bit Rate) paketi olduğunu göstermektedir. Benzetimde UDP ve JITT protokolünü kullanan uygulama katmanı protokolü CBR olduğu için paket başlığında CBR'ye karşılık gelen değer bulunmaktadır. Eğer alınan paket CBR ise (yani bu durumda JITT paketi ise) Time Stamp, MWT, Delay gibi alanların belirlenerek paketin ayrı bir kuyruğa alınması işlemi aşağıdaki kodlarla yapılmaktadır.

```
double now = Scheduler::instance().clock();
// bu kuyrukta ne kadar bekleme hakkı var?
double MWT = (hdr_cmn::access(p)->JITT_Delay_ / (ih->ttl_-1));
    hdr_cmn::access(p)->MWT_   = MWT;
    hdr_cmn::access(p)->Time_Stamp_ =  now;
    q_jitt_->enque(p);
```

Yukarıdaki kodlarda bir paketin kuyrukta bekleme süresi hesaplanırken TTL değerinin bir eksiği kullanılmıştır. Bunun nedeni, TTL değerinin ilk olarak belirlenmesi esnasında alıcı düğümün de hesaba katılmasıdır. Ancak bekleme süresi aradaki yönlendirici düğümler üzerinde söz konusu olduğu için hedef düğüm hesaba katılmamış ve TTL değerinin bir eksiği ile hesaplama yapılmıştır.

Yukarıda anlatılan hesaplamalar paketler kuyruğa alınırken yapılmaktadır. Benzetim aracının kuyruktan bir paket almak istemesi durumunda, yani bir paketin iletilmesi istendiğinde hangi paketin gönderileceği aşağıdaki kodlarla belirlenmektedir.

```
double now = Scheduler::instance().clock();
int found = 0;
Packet *pkt=Packet::alloc();
if (q_jitt_->length()>0)
        {
        for (int i=0;i<=q_jitt_->length()-1;i++)// Kuyruğu tara
        {
      *pkt = *q_jitt_->lookup(i);
      // Gönderilme zamanı gelmiş paket var mı?
```

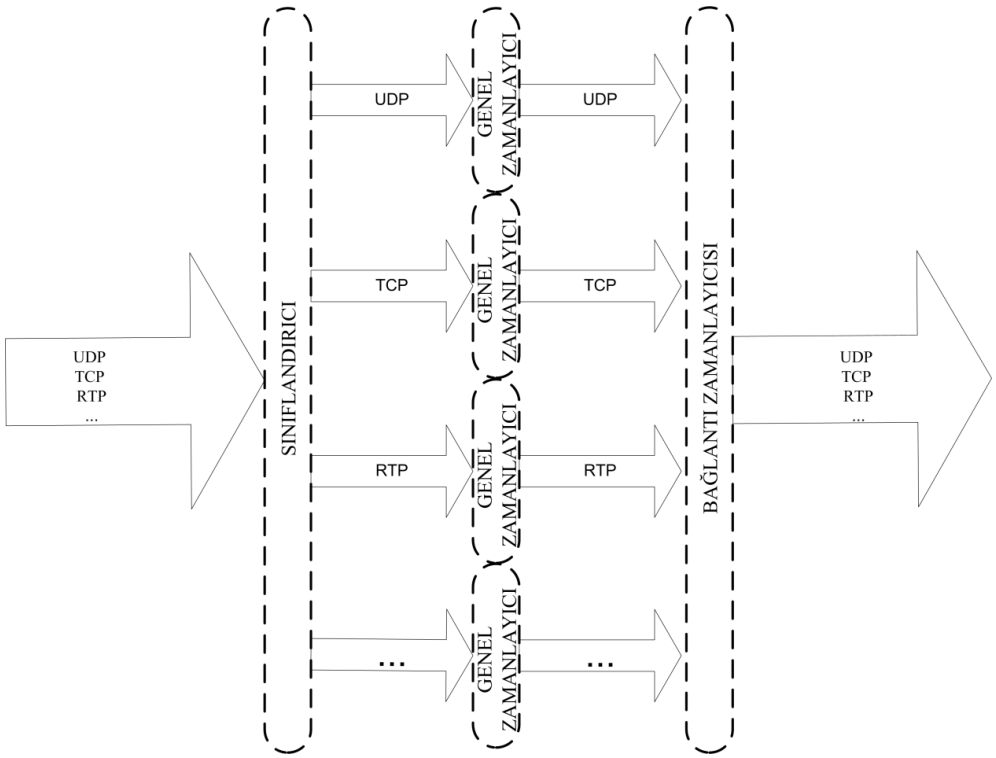
```
if ((now-hdr_cmn::access(pkt)->Time_Stamp_-
pkt->length()/BW)>hdr_cmn::access(pkt)->MWT_)
  {
  found=1;
  q_jitt_->remove(q_jitt_->lookup(i),q_jitt_->lookup(i-1));
  hdr_cmn::access(pkt)->JITT_Delay_ =
(hdr_cmn::access(pkt)->JITT_Delay_-
(now-hdr_cmn::access(pkt)->Time_Stamp_)+( pkt->length()/BW));
break;

}
  }
  }
  if (found==1) return(pkt); // zamanı gelen JITT paketi var
  else return q_->deque(); } // zamanı gelen JITT paketi yok
```

Yukarıdaki kodlar kuyruk yapısının deque (kuyruktan çıkarma) fonksiyonunda bulunmaktadır. Bir paket gönderileceği zaman öncelikle JITT kuyruğu taranmakta ve gönderilme zamanı gelen bir paket olup olmadığı kontrol edilmektedir. Eğer gönderilme zamanı gelen paket varsa bu paket gönderilmekte, yoksa diğer kuyruktaki ilk paket gönderilmektedir. Kontrol esnasında `pkt->length()/BW`, paketin zaman damgası değerinden çıkarılarak bir hesaplama yapılmaktadır. Böylece, gönderilecek pakete karar verildikten sonra paketin gönderilmesi için gereken süre göz önünde bulundurulmaktadır.

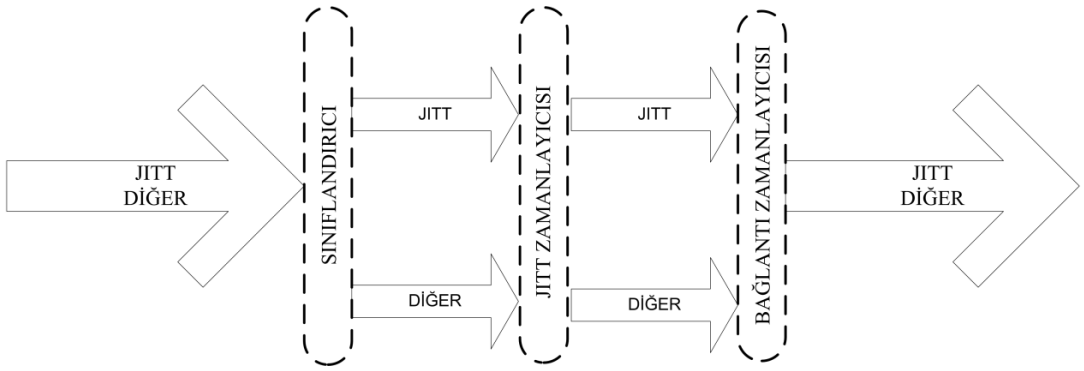
3.4. Bağlantı Zamanlayıcısı ve Geliştirilen Kuyruk Modeli

Yönlendiriciler üzerinde paket sınıflandırması ve zamanlama yapılması ile birlikte iki farklı zamanlayıcı kavramı ortaya çıkmıştır. Bunlar bağlantı zamanlayıcısı ve genel zamanlayıcıdır [24]. Genel zamanlayıcı, sınıflara ayrılmış olan trafikler üzerinde çalışan ve her bir trafiğe özel olan zamanlayıcı türüdür. Bağlantı zamanlayıcısı ise, tıkanıklık durumunda bant genişliğinin sınıflar arasında nasıl dağıtılacağını belirleyen zamanlayıcı türüdür. JITT hem kendi trafik türü ile hem de diğer trafikler ile ilgilendiğinden JITT kuyruk modelinde yapılan zamanlama hem genel zamanlayıcı hem de bağlantı zamanlayıcısı sınıfına girmektedir. Genel zamanlayıcı ve bağlantı zamanlayıcılarının nasıl çalıştığı Şekil 3.8'de gösterilmektedir.



Şekil 3.8. Genel zamanlayıcı ve bağlantı zamanlayıcısı

Bir yönlendirici üzerinde bir bağlantı zamanlayıcısının çalıştırılması mecburiyeti bulunmamaktadır. Ancak bant genişliğinin, sınıflandırılmış trafikler arasında adaletli kullandırılması isteniyorsa bir bağlantı zamanlayıcısının kullanılması yerinde olacaktır. JITT hem genel zamanlayıcı hem de bağlantı zamanlayıcısı gibi çalıştığından herhangi bir bağlantı zamanlayıcısı ile çalışmasında problem yaşanmayacaktır. JITT zamanlayıcısının çalışma mantığı Şekil 3.9'da gösterilmektedir.



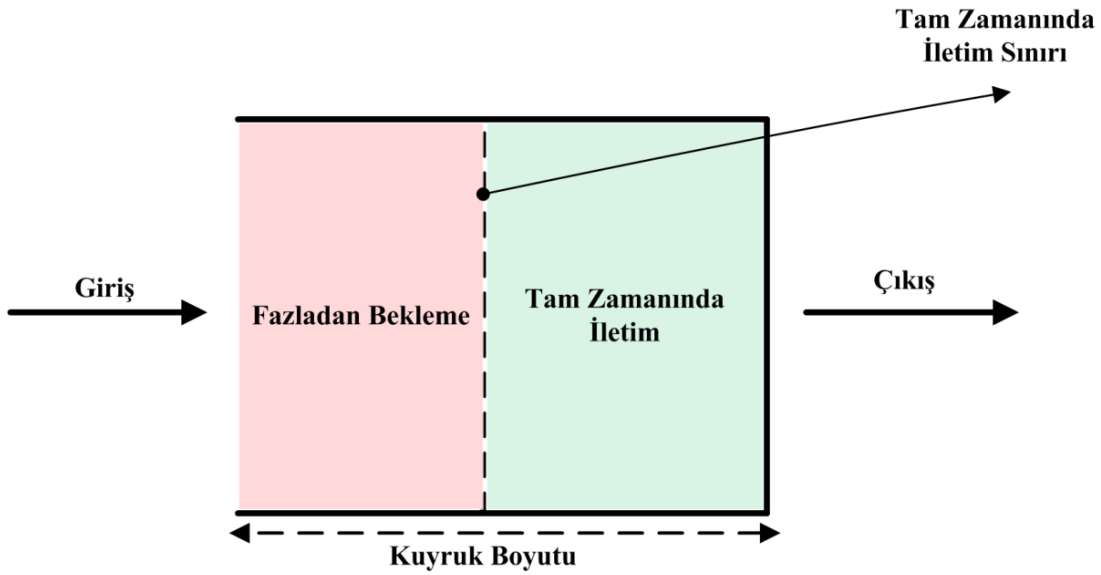
Şekil 3.9. JITT zamanlayıcısı

JITT zamanlayıcısı, bağlantı zamanlayıcısından paket gönderme için sırayı aldığında eğer JITT kuyruğunda gönderilme zamanı gelmiş bir paket varsa bunu gönderir. Gönderilme zamanı gelmiş bir paket yoksa diğer kuyruktan bir paket gönderir ve gönderme sırasını diğer kuyruğa devretmiş olur.

Literatürde gerçekleştirilmiş olan birçok bağlantı zamanlayıcısı bulunmaktadır [24]. Bu çalışmanın 3.8 başlığında JITT'in TCP trafiği üzerindeki etkisi incelenirken bant genişliğini tıkanıklık durumunda eşit olarak paylaştıran bağlantı zamanlayıcısı kullanılmıştır.

3.5. Geliştirilen Kuyruk Modeli İçin Kuyruk Boyutunun Belirlenmesi

JITT kuyruk modelinin uygulanacağı bir yönlendiricide JITT kuyruk boyutunun seçilmesi, modelin düzgün çalışması açısından önemlidir. JITT kuyruk boyutu belirli bir seviyeden daha düşük seçilirse, zamanında hedefine ulaştırılabilecek JITT paketleri kuyruğa alınmayabilir. Durumu daha iyi açıklamak için Şekil 3.10 irdelenmelidir.



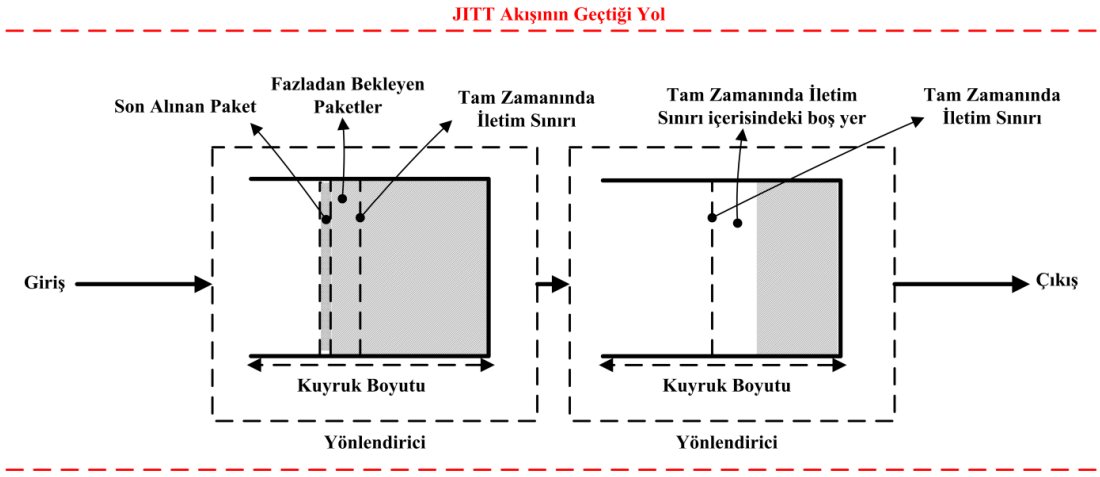
Şekil 3.10. Bir yönlendirici üzerindeki JITT kuyruğu

Bir JITT kuyruğu $Q = \{P_1, P_2, \cdots, P_n\}$ şeklinde gösterilsin. Burada P_i ,$1 \leq i \leq n$ kuyrukta bulunan paketleri göstermektedir. τ_i, P_i paketinin MWT değeri olmak üzere $\tau_{max} = \max_{1 \leq i \leq n}\{\tau_i\}$ olsun. P_{max} ise MWT değeri τ_{max} olan paketi göstersin. O zaman $\{\Phi = Q - P_{max}\}$ alt kümesi oluşturulabilir. $\{\forall P_i \in \Phi\}$ için φ_i i. paketin bit cinsinden boyutunu göstersin. Φ alt kümesinin her bir elemanının bit cinsinden boyut toplamı $\Theta = \sum_{i=1}^{n-1} \varphi_i$ olarak hesaplanır. B_{out} yönlendiricinin çıkış bant genişliği olmak üzere $\frac{\Theta}{B_{out}} < \tau_{max}$ durumunda "Tam Zamanında İletim" yapılmakla birlikte tam zamanında iletilmesi mümkün olan $\left(\tau_{max} - \frac{\Theta}{B_{out}}\right) \times B_{out}$ bit kadar veri kuyruğa alınmadan atılacaktır. $\frac{\Theta}{B_{out}} = \tau_{max}$ durumunda "Tam Zamanında İletim" yapılacak ve çıkış bant genişliği tam verimle kullanılacaktır. Yani, kendi τ süresinde gönderilebilecek bir paketin kuyruğa kabul edilmemesi durumu oluşmayacaktır. $\frac{\Theta}{B_{out}} > \tau_{max}$ durumunda ise "Fazladan Bekleme" söz konusu olacaktır. Yani paketler kendi τ sürelerinden (kuyruktaki bekleme süresi haklarından, yani MWT değeri) daha fazla bir süre kuyrukta bekleyeceklerdir. JITT, bir bağlantının üzerinde bulunan yönlendiricilerdeki JITT kuyruklarının tamamını tek kuyruk olarak kabul etmektedir. Bir JITT paketi bir yönlendirici üzerinde fazladan bir süre beklemişse ve paketin geçeceği diğer yönlendiricilerde **anlık olarak** $\frac{\Theta}{B_{out}} < \tau_{max}$ durumu söz konusu ise bu süre ilgili yönlendirici tarafından telafi edilebilir. Bu nedenle bir yönlendirici üzerindeki JITT kuyruk boyutu en az $\{\tau_{max} \times B_{out}\}$ bit seçilmelidir ki çıkış bant genişliği verimli kullanılabilsin ve bir yönlendirici üzerinde yaşanabilecek fazladan gecikmeler diğer yönlendiriciler tarafından telafi edilebilsin.

3.6. Geliştirilen Kuyruk Modelinin Bir Şebeke Olarak Ele Alınması

JITT, başlığındaki Delay alanı sayesinde bir yönlendirici üzerinde yaşanabilecek fazladan veya az bekleme sürelerini JITT akışının geçtiği yol üzerinde yer alan sıradaki yönlendiricilere yansıtır. Örneğin, Delay alanı 100 ms olan bir JITT paketi 2 yönlendiricinin bulunduğu bir yol üzerinden aktarılmak istendiğinde normal koşullarda her bir yönlendiricide 50 ms kuyrukta bekleme hakkı vardır. Bu paketi

alan ilk yönlendirici diğer kuyruk trafiğinin yoğun olmaması nedeni ile, ilgili paketi 30 ms sonra diğer yönlendiriciye gönderirse sıradaki yönlendiricinin bu paketi bekletme hakkı (50-30)+50 = 70 ms olur. Yani ilk yönlendirici, kullanmadığı 20 ms'lik bekletme hakkını sıradaki yönlendiriciye devreder. Tersi durumda, önceki yönlendiricide yaşanabilecek fazladan bir gecikme de sonraki yönlendiricilere yansıtılır. Bu bakış açısı ile JITT, akışın geçtiği yol üzerindeki bütün yönlendiricilerde bulunan kuyrukları tek bir kuyrukmuş gibi değerlendirir. Şekil 3.11'de yukarıda anlatılan durum için bir örnek gösterilmektedir.



Şekil 3.11. Bir şebeke olarak JITT kuyruk yapısının incelenmesi

Şekil 3.11'e göre 2 yönlendiricinin kullanıldığı bir şebekede ilk yönlendiricideki JITT kuyruğuna son alınan paketin durumu incelenecek olursa, bu paketin ilk yönlendiriciden tam zamanında gönderilemeyeceği açıktır. Son alınan paketin MWT süresinin, kuyruktaki diğer tüm paketlerin MWT sürelerinden büyük olduğu kabul edilerek; ilgili paketin bit cinsinden boyutu φ, ilk yönlendiricide yer alan paketlerin (son alınan paket dâhil) bit cinsinden boyut toplamları Θ_1, ikinci yönlendiricide kuyrukta bekleyen paketlerin bit cinsinden boyut toplamları Θ_2, ikinci yönlendiricinin girişindeki bps cinsinden veri oranı λ_2, ikinci yönlendiricinin bps cinsinden çıkış bant genişliği μ_2 ve ikinci yönlendiricideki tam zamanında iletim sınırı içerisindeki boş yer σ_2 olsun. İlgili paketin hedefine τ süre içerisinde varabilmesi için Eş. 3.5'in sağlanması yeterlidir.

$$\Theta_2 + \varphi + \left(\frac{\Theta_1}{\mu_1}\right)(\lambda_2 - \mu_2) \leq \sigma_2 \qquad (3.5)$$

Eş. 3.5, bir JITT paketinin, iletişim yolu üzerindeki 2 yönlendiricinin kuyruklarına varış ve kuyruklarından ayrılış zamanlarına göre ilgili kuyruk boyutlarının ne şekilde değiştiklerini göstermektedir. Eş. 3.5, n adet yönlendirici bulunan bir iletişim yolu için genelleştirilirse Eş. 3.6 elde edilir.

$$\sum_{j=2}^{n}\left[\left(\Theta_j + \varphi + \frac{\Theta_{j-1}}{\mu_{j-1}}(\lambda_j - \mu_j)\right) - \sigma_j\right] \leq 0 \qquad (3.6)$$

3.7. Geliştirilen Ulaştırma Protokolü ve Kuyruk Modelinin Performansının Benzetim Ortamında Değerlendirilmesi

JITT'in performansının değerlendirilmesi ve diğer ulaştırma katmanı protokol ve kuyruk yapılarıyla karşılaştırılabilmesi amacıyla 4 adet senaryo üretilmiş ve denemeler gerçekleştirilmiştir.

GZÇO verilerinin aktarıldığı bir ağ ortamını benzetebilmek için ilk şart doğru biçimde GZÇO verilerini üreten kaynakları oluşturabilmektir. Domoxoudis ve ark., yaptıkları çalışmada 7 farklı VBR (Variable Bit Rate) kodlayıcı ile gerçek ortamda 1 saat deneme yaparak video konferans iletişimi için trafik desenleri çıkartmışlardır [44]. Çizelge 3.1'de bu desenler gösterilmektedir.

Çizelge 3.1. VBR kodlayıcıları trafik desenleri [44]

Kodlayıcı	NV	NVDCT	H261	H263	H263+	CELLB	BVC
Deneme Süresi (sn)				3600			
Ortalama Video Bit Oranı (Kbps)	182	121	63	54	13	93	92
Ortalama Çerçeve Oranı (fps)	14	15	15	15	5	15	15
Ortalama Çerçeve Boyutu (Byte)	1638	1023	527	457	331	779	766

Tez kapsamında gerçekleştirilen benzetimlerde en yüksek ortalama çerçeve boyutu olan 1638 Byte ve saniyede 25 çerçeve oranı kullanılmıştır. Böylece 327,600 Kbps bit oranı elde edilmiştir. Gerçek zamanlı ses kaynağını benzetmek için de benzer

şekilde bir VBR trafik deseni kullanılmıştır. Ses için 47,2 Kbps bit oranı elde edilmiştir.

JITT'in performansını değerlendirmek ve karşılaştırmak için her bir senaryo JITT haricinde UDP-DropTail, UDP-RED, RTP-DropTail ve RTP-RED ulaştırma protokolü-kuyruk yapısı çiftleri ile denenmiştir. Karşılaştırma ölçütleri olarak gecikme, jitter, ortalama gecikmeden maksimum sapma ve paket atılma oranları kullanılmıştır. Gecikme ve gecikmenin kararlı olması, gerçek zamanlı haberleşmelerde son kullanıcılara verilen hizmet kalitesi için önemli parametrelerdir.

Jitter, RFC 3393'te şu şekilde tarif edilmektedir:

"Gecikme gibi bir ölçütün bir referans değere göre (ortalama gecikme veya minimum gecikme gibi) gösterdiği değişimdir." [4].

Tez kapsamında gerçekleştirilen benzetimlerde JITT protokolü için referans değer olarak, JITT parametrelerinden olan Delay değişkeninde (kullanıcı tarafından ilgili haberleşme için istenilen gecikme değerini belirleyen parametre) belirtilen değer alınmıştır. Diğer protokol-kuyruk yapısı çiftleri için referans değer olarak kendi ortalama gecikme değerleri alınmıştır. Böylece jitter değerleri bu referans değerlere göre hesaplanmıştır. Ortalama gecikmeden maksimum sapma parametresi, son kullanıcı tarafında bulunan oynatma tamponlarının boyutlarının belirlenmesi için önemli bir parametredir [4,45]. Oynatma tamponunun çok büyük olması haberleşmeye yapay bir gecikme eklemektedir.

Son olarak paket atılma oranının yüksek olması hem gerçek zamanlı haberleşmenin kalitesini hem de diğer trafiklerin kalitesini olumsuz yönde etkilemektedir. Bu nedenle, gerçekleştirilen senaryolarda hem GZÇO trafiklerden atılan veri oranları hem de TCP trafiklerinden atılan veri oranları ölçülmüştür. İnternet trafiğinin %90-95'i TCP üzerinden akmaktadır [46]. Bu nedenle, geliştirilen yöntemin TCP trafiğine olan etkisi de analiz edilmiştir.

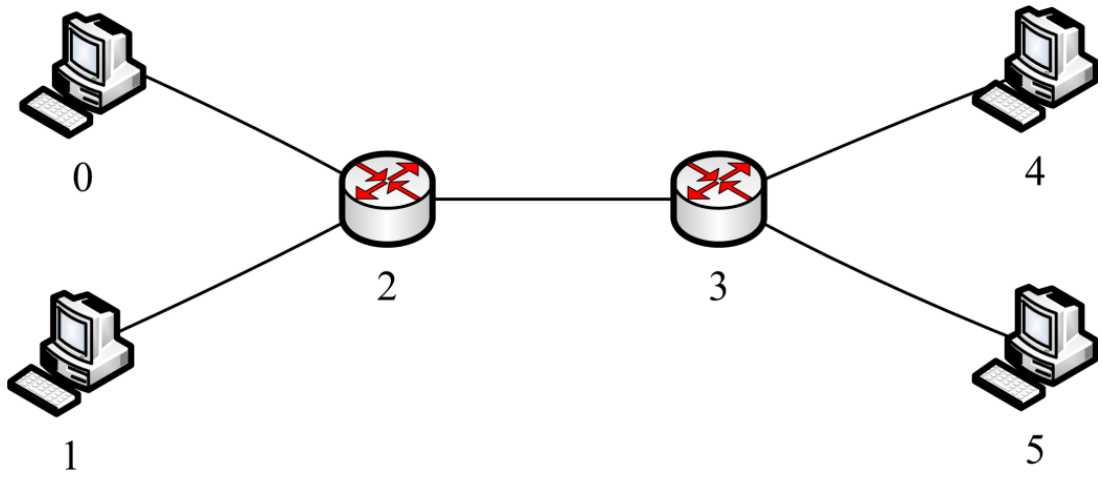
3.7.1. Birinci senaryo

Gerçekleştirilen benzetimde kullanılan parametreler aşağıdaki gibidir.

Çizelge 3.2. Birinci benzetimde kullanılan parametreler

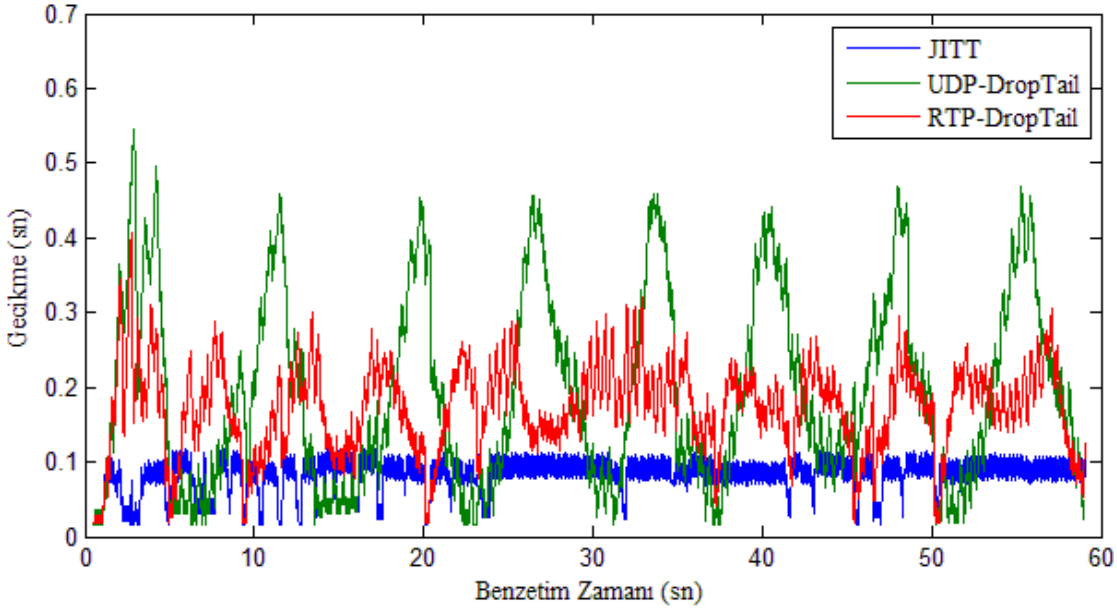
Benzetim Süresi			
65 sn			
Yönlendirici Olarak Görev Yapan Düğümler ve Aralarındaki Bağlantı Kapasitesi			
2-3, 0.5Mbps 10ms			
Trafik Tipi			
Video Konferans		TCP	
Haberleşen Düğümler		*Haberleşen Düğümler*	
0-4	4-0	1-5	5-1
Haberleşme Süresi	*Haberleşme Süresi*	*Haberleşme Süresi*	*Haberleşme Süresi*
0.5-59 sn	0.5-59 sn	1-59 sn	1-59 sn

Birinci senaryonun topolojisi Şekil 3.12'deki gibidir.

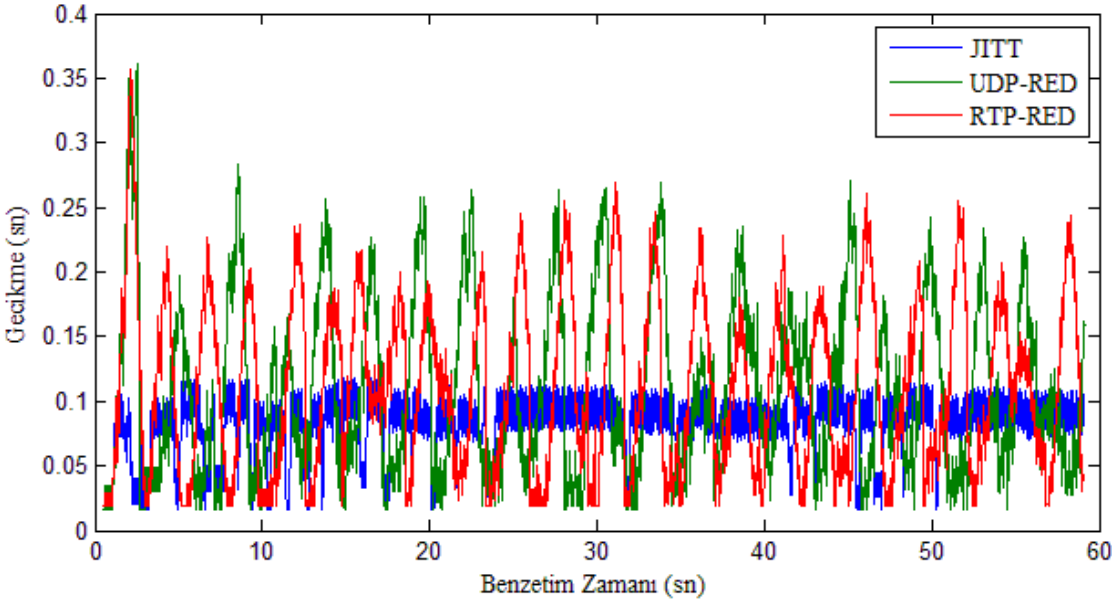


Şekil 3.12. Birinci senaryonun topolojisi

Birinci senaryo ile elde edilen gecikme grafikleri Şekil 3.13 ve Şekil 3.14'te gösterilmektedir.



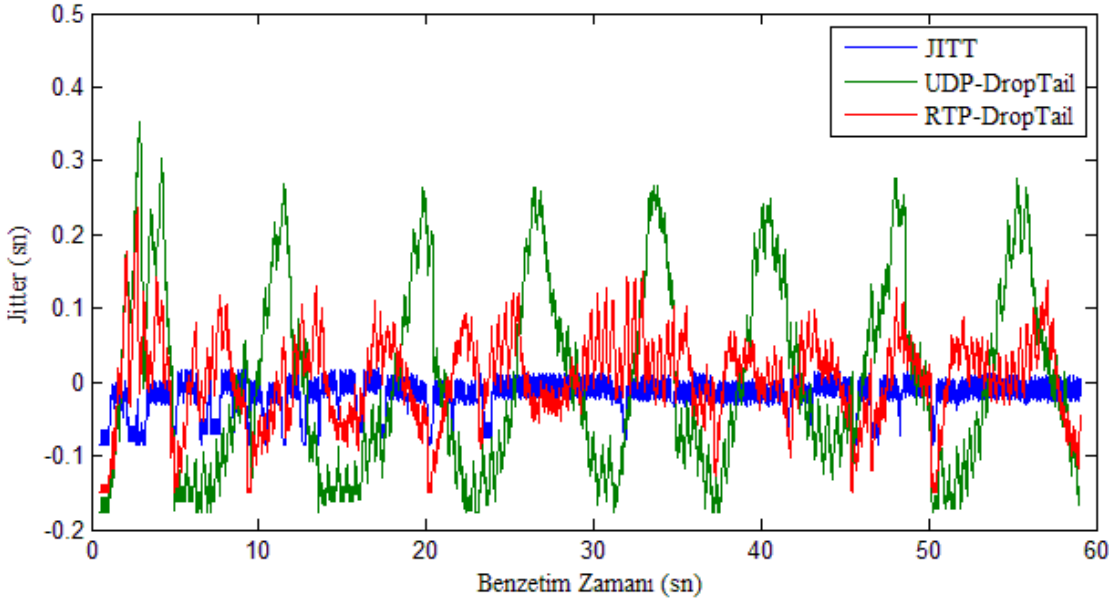
Şekil 3.13. Birinci senaryo gecikme grafiği (JITT, UDP-DropTail, RTP-DropTail)



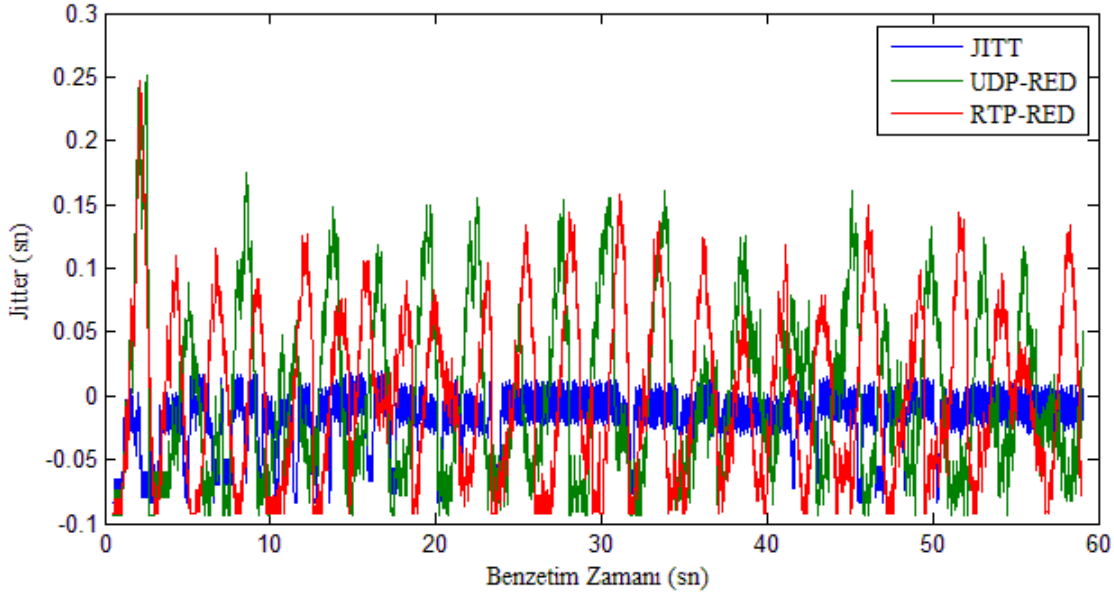
Şekil 3.14. Birinci senaryo gecikme grafiği (JITT, UDP-RED, RTP-RED)

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük ve daha kararlı gecikme sağladığı görülmektedir.

Birinci senaryo ile elde edilen jitter grafikleri Şekil 3.15 ve Şekil 3.16'da gösterilmektedir.

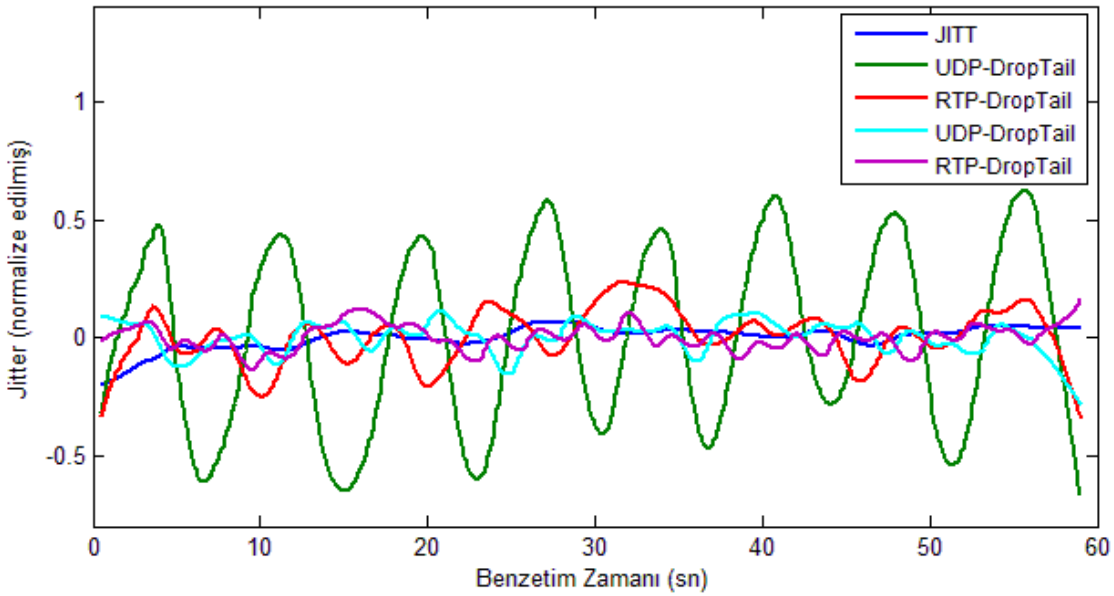


Şekil 3.15. Birinci senaryo jitter grafiği (JITT, UDP-DropTail, RTP-DropTail)



Şekil 3.16. Birinci senaryo jitter grafiği (JITT, UDP-RED, RTP-RED)

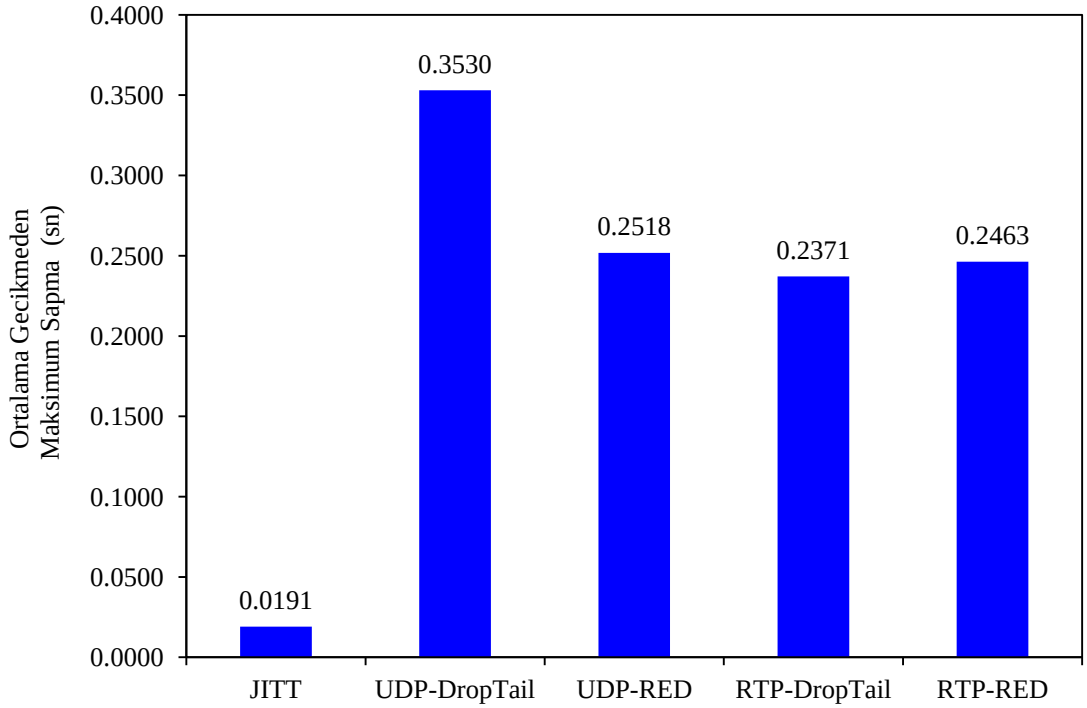
Ortalama gecikmesi en yüksek olan denemenin ortalama gecikmesi 1 olarak kabul edilerek bütün denemelerden elde edilen gecikmelerin buna göre normalize edilmesi sonucunda elde edilen jitter grafiği aşağıdaki şekilde gösterilmektedir.



Şekil 3.17. Birinci senaryo jitter grafiği (normalize edilmiş)

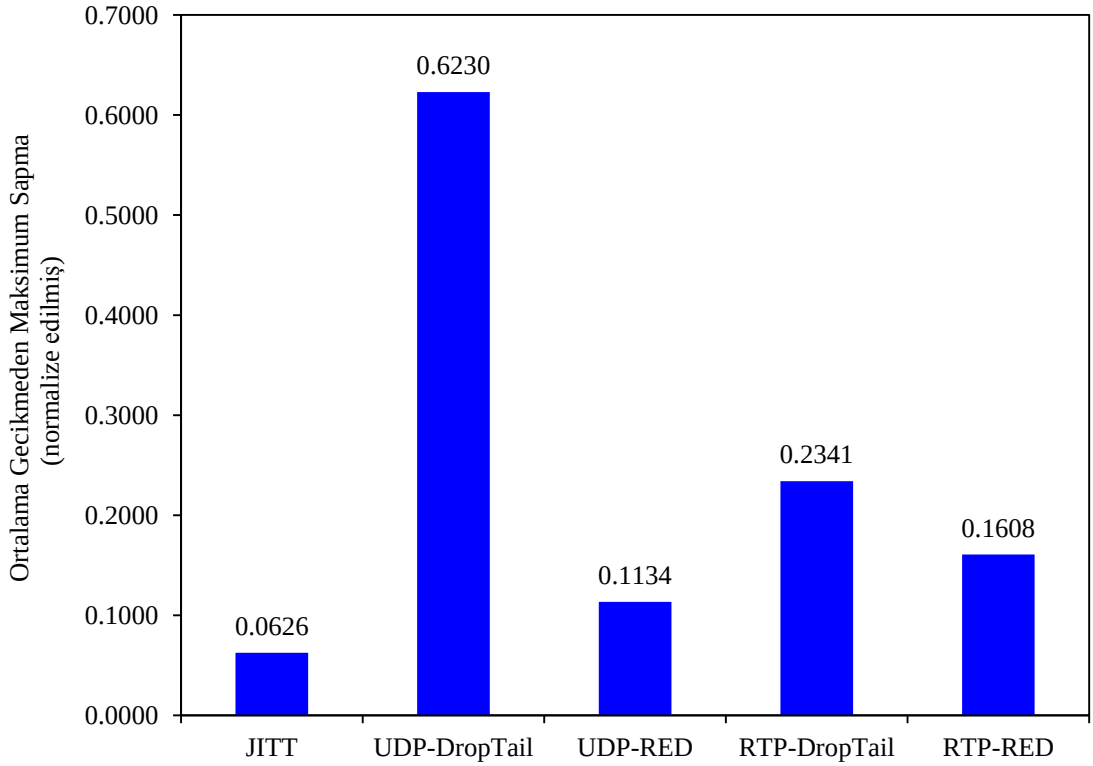
Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük jitter sağladığı görülmektedir.

Oynatma tamponunun belirlenmesinde önemli bir parametre olan ortalama gecikmeden maksimum sapma için elde edilmiş olan değerler Şekil 3.18'de gösterilmektedir.



Şekil 3.18. Birinci senaryo ortalama gecikmeden maksimum sapma grafiği

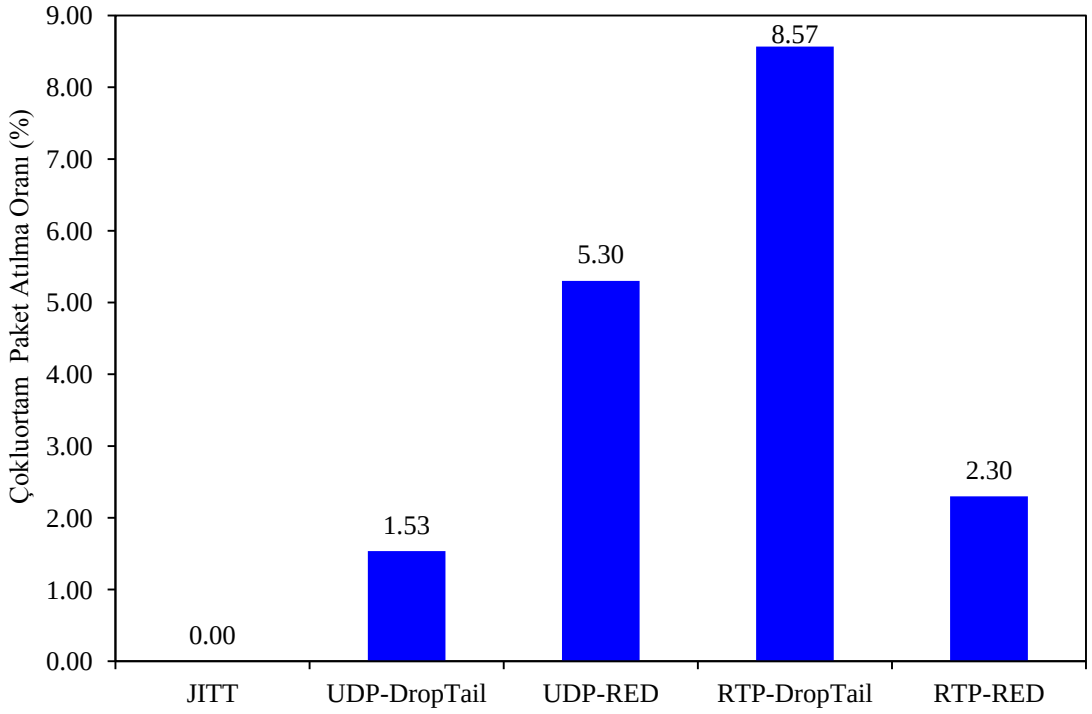
Ortalama gecikmesi en yüksek olan denemenin ortalama gecikmesinin 1 olarak kabul edilerek bütün denemelerden elde edilen ortalama gecikmeden maksimum sapma değerlerinin buna göre normalize edilmesi sonucunda elde edilen ortalama gecikmeden maksimum sapma grafiği Şekil 3.19'da gösterilmektedir.



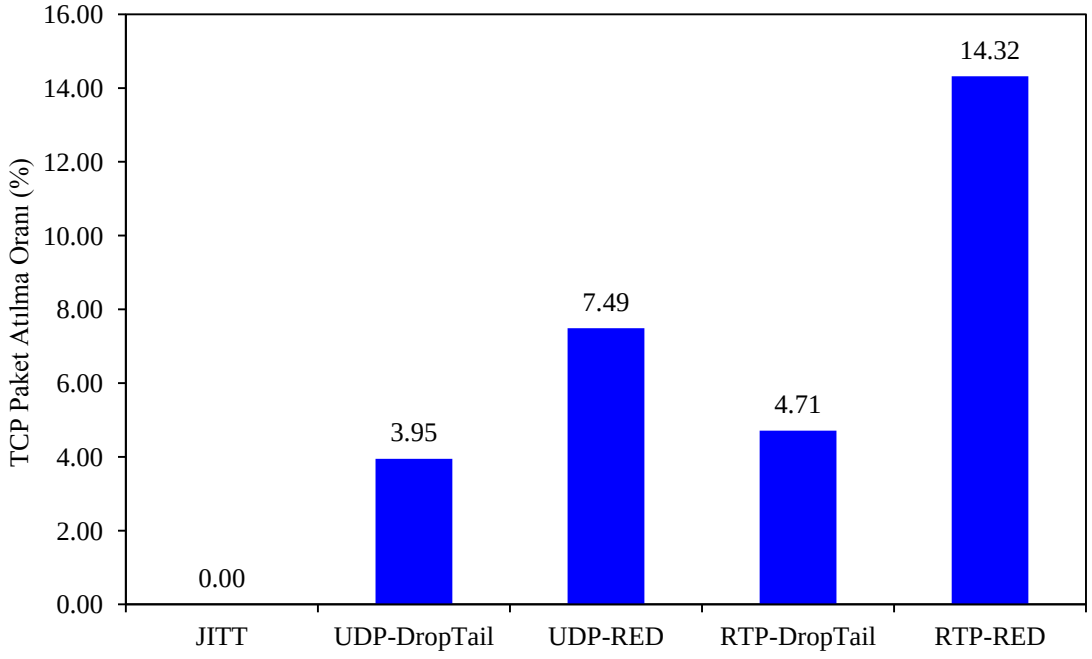
Şekil 3.19. Birinci senaryo ortalama gecikmeden maksimum sapma grafiği (normalize edilmiş)

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük ortalama gecikmeden maksimum sapma sağladığı görülmektedir.

Son olarak incelenen ölçüt, atılan veri miktarının üretilen veri miktarına oranıdır. Şekil 3.20'de GZÇO trafiğinden atılan veri oranları; Şekil 3.21'de TCP trafiğinden atılan veri oranları gösterilmektedir. İlgili şekillerden de görüldüğü gibi birinci senaryoda JITT'in paket atılmalarını önlediği görülmektedir. Gecikme ve jitter gibi iki önemli parametrede büyük iyileştirmeler yaparken, aynı zamanda diğer protokol-kuyruk yapısı çiftlerine göre TCP trafiğinin de performansını etkilememek/az etkilemek kayda değer bir başarı olarak gözükmektedir.



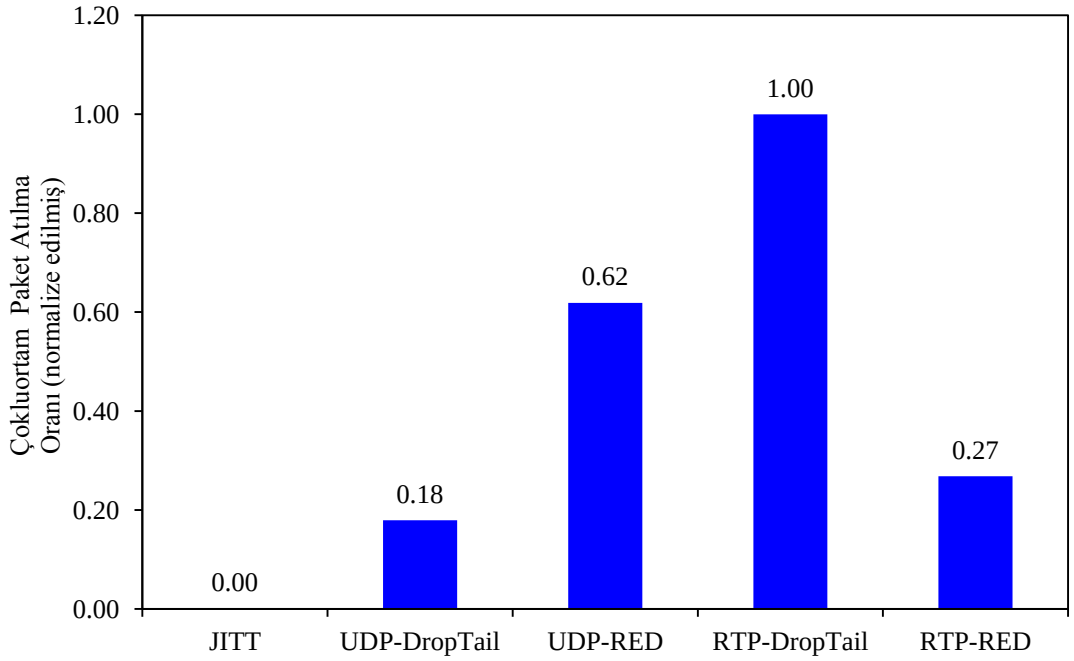
Şekil 3.20. Birinci senaryo GZÇO veri atılma oranları grafiği



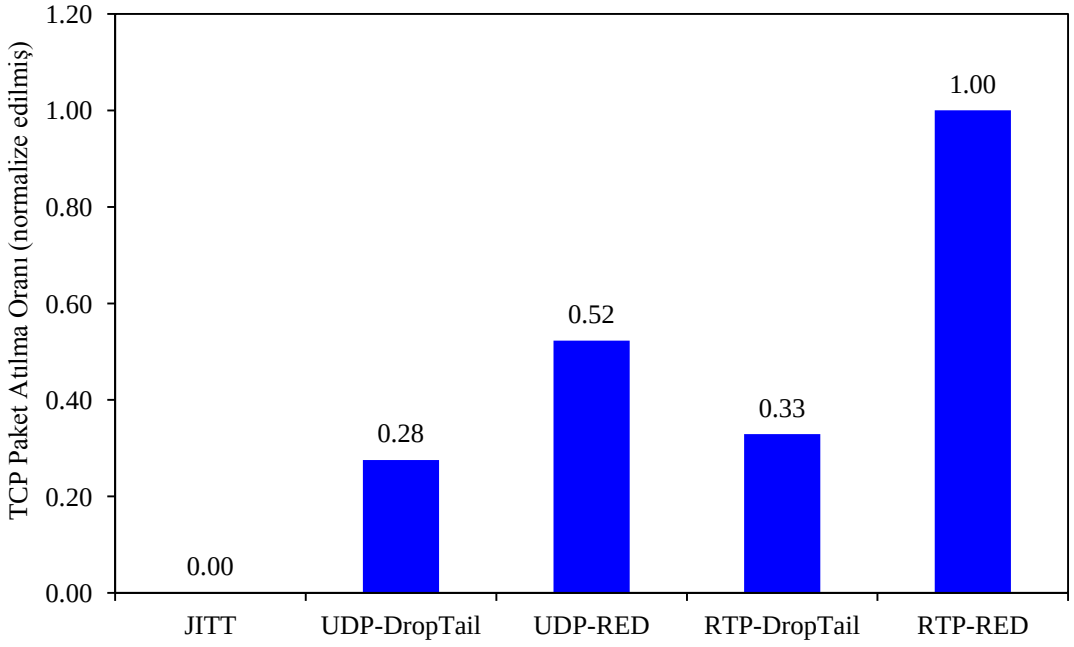
Şekil 3.21. Birinci senaryo TCP veri atılma oranları grafiği

Paket atılma oranı en yüksek olan denemenin paket atılma oranı 1 olarak kabul edilerek bütün denemelerden elde edilen paket atılma oranlarının buna göre

normalize edilmesi sonucunda elde edilen paket atılma oranları grafikleri aşağıdaki şekillerde gösterilmektedir.



Şekil 3.22. Birinci senaryo GZÇO veri atılma oranları grafiği (normalize edilmiş)



Şekil 3.23. Birinci senaryo TCP veri atılma oranları grafiği (normalize edilmiş)

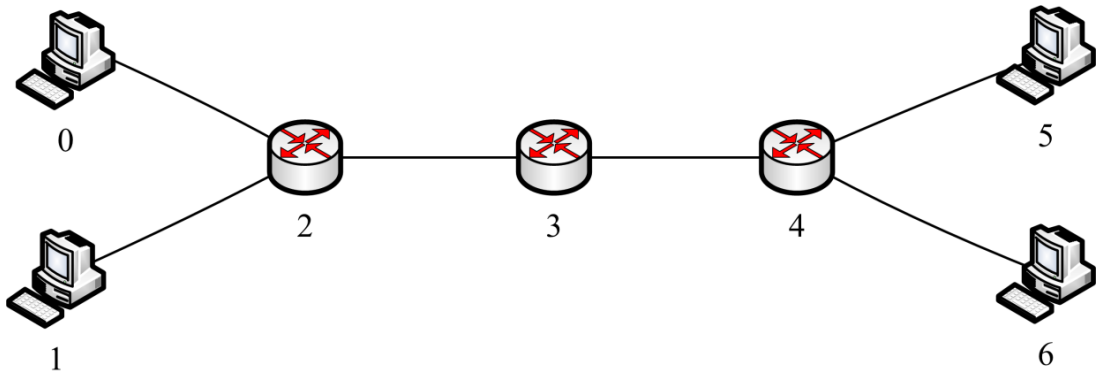
3.7.2. İkinci senaryo

Gerçekleştirilen benzetimde kullanılan parametreler aşağıdaki gibidir.

Çizelge 3.3. İkinci benzetimde kullanılan parametreler

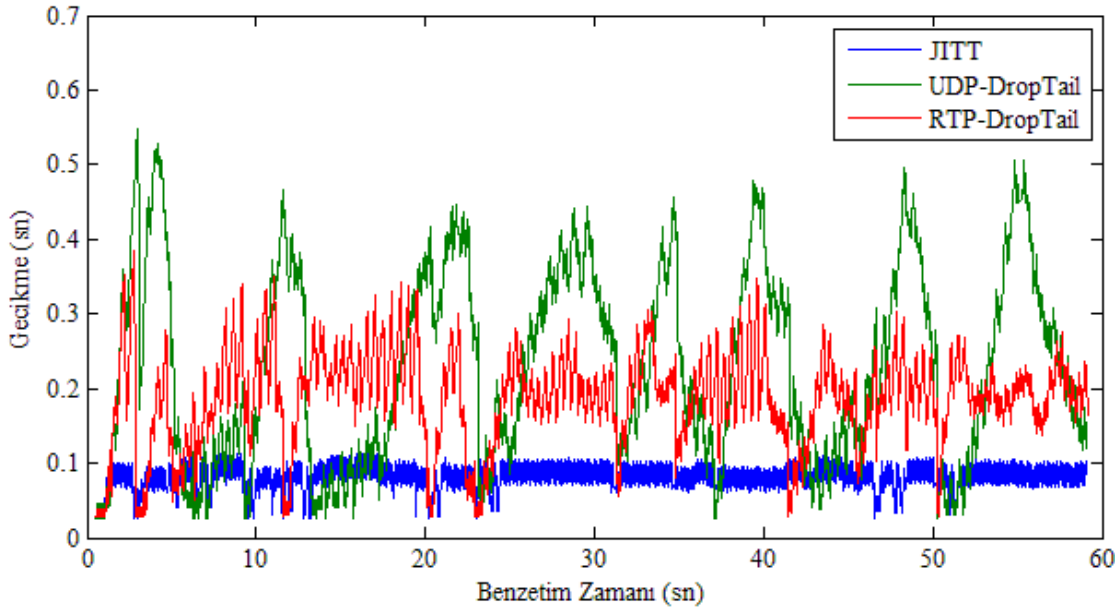
Benzetim Süresi			
65 sn			
Yönlendirici Olarak Görev Yapan Düğümler ve Aralarındaki Bağlantı Kapasitesi			
2-3-6, 0.5Mbps 10ms			
Trafik Tipi			
Video Konferans		TCP	
Haberleşen Düğümler		*Haberleşen Düğümler*	
0-5	5-0	1-6	6-1
Haberleşme Süresi	*Haberleşme Süresi*	*Haberleşme Süresi*	*Haberleşme Süresi*
0.5-59 sn	0.5-59 sn	1-59 sn	1-59 sn

İkinci senaryonun topolojisi Şekil 3.24'deki gibidir.

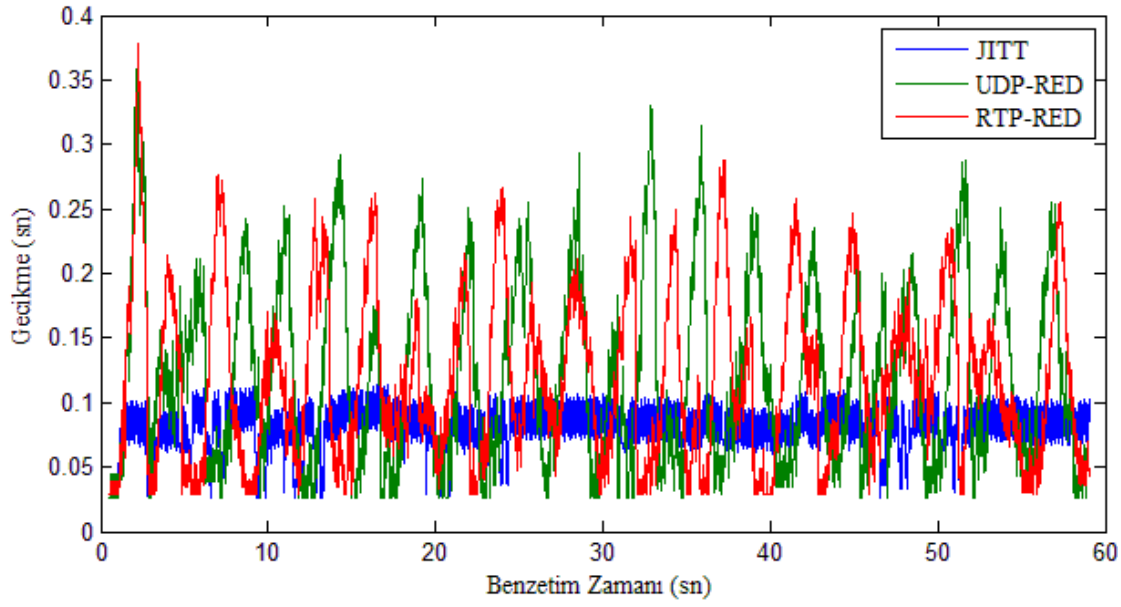


Şekil 3.24. İkinci senaryonun topolojisi

İkinci senaryo ile elde edilen gecikme grafikleri Şekil 3.25 ve Şekil 3.26'da gösterilmektedir.



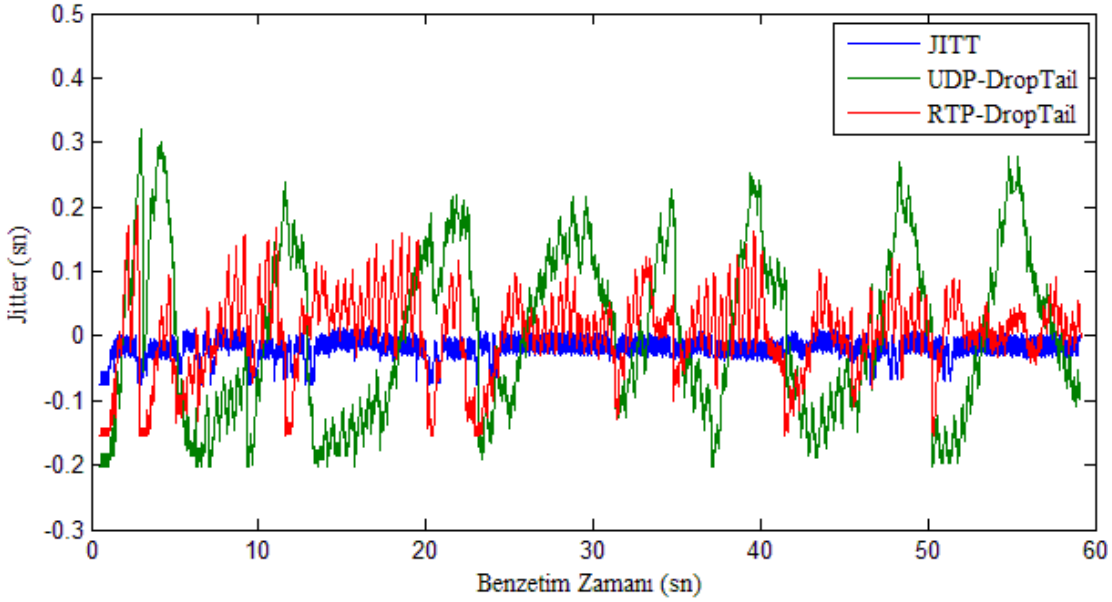
Şekil 3.25. İkinci senaryo gecikme grafiği (JITT, UDP-DropTail, RTP-DropTail)



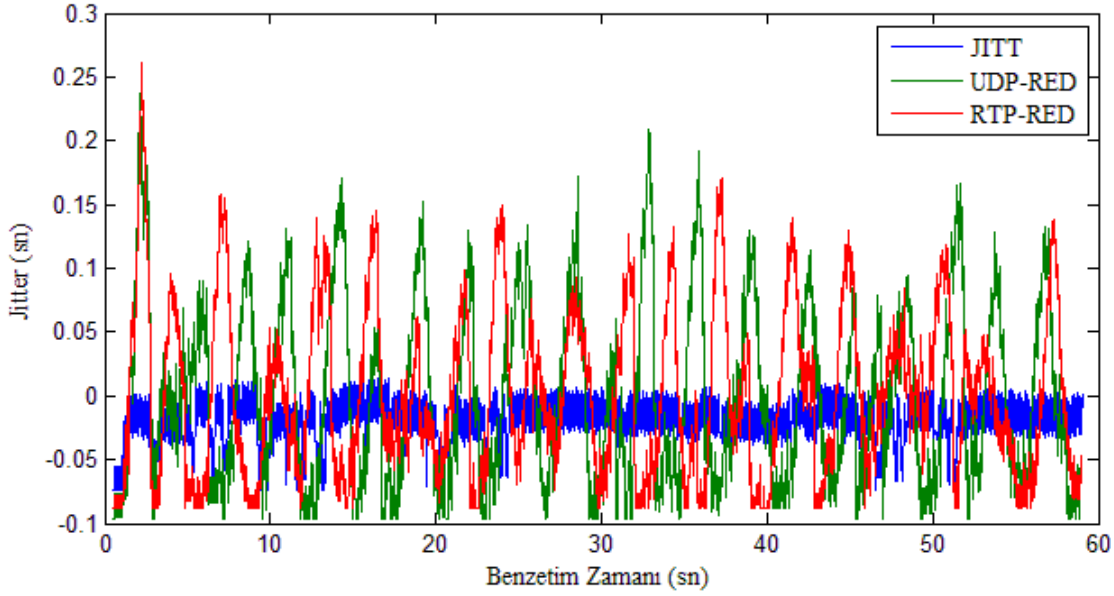
Şekil 3.26. İkinci senaryo gecikme grafiği (JITT, UDP-RED, RTP-RED)

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük ve daha kararlı gecikme sağladığı görülmektedir.

İkinci senaryo ile elde edilen jitter grafikleri Şekil 3.27 ve Şekil 3.28'de gösterilmektedir.

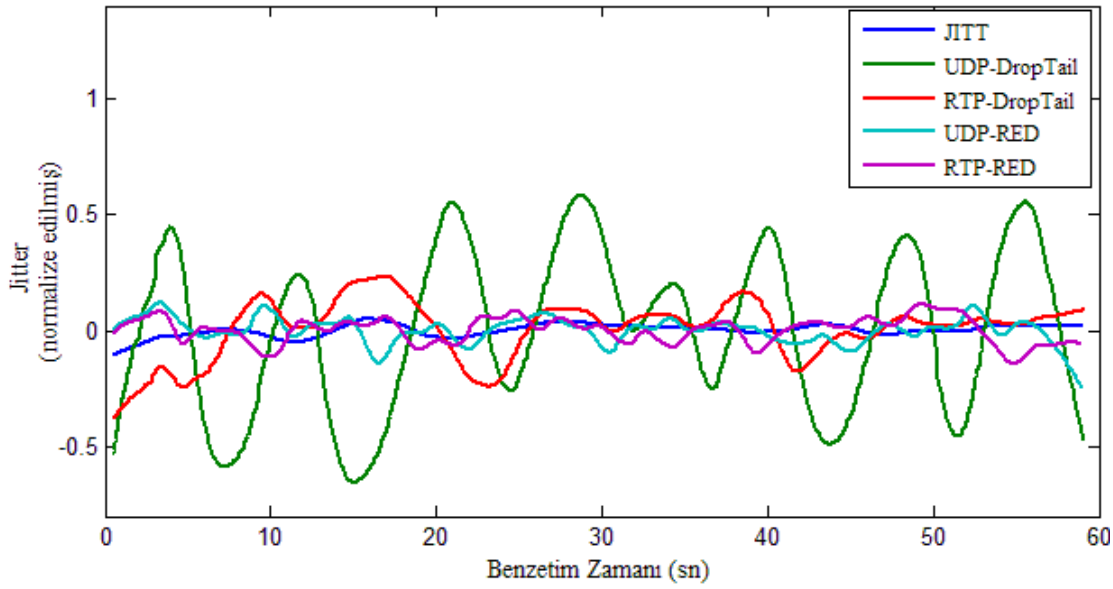


Şekil 3.27. İkinci senaryo jitter grafiği (JITT, UDP-DropTail, RTP-DropTail)



Şekil 3.28. İkinci senaryo jitter grafiği (JITT, UDP-RED, RTP-RED)

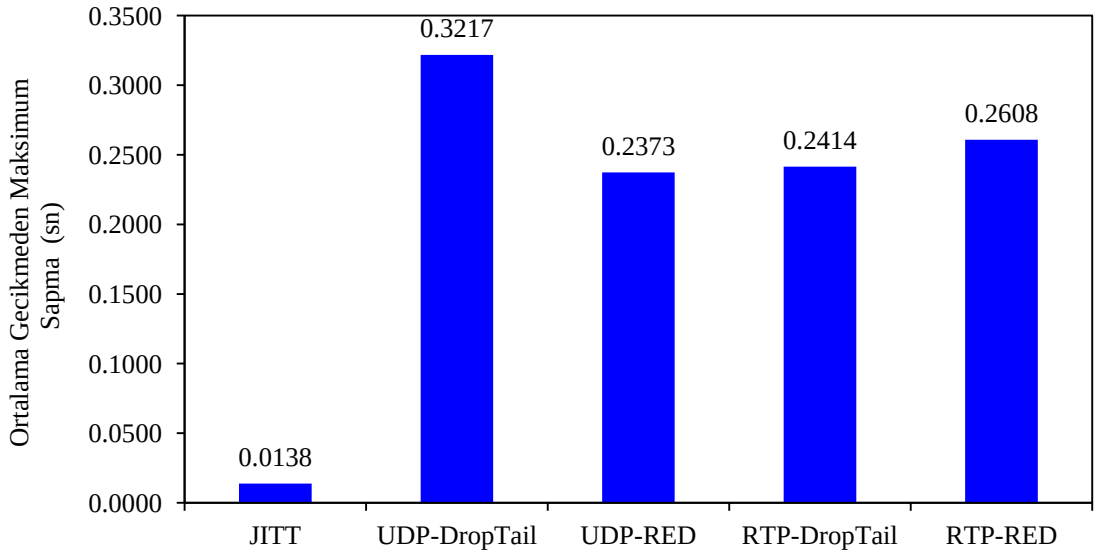
Ortalama gecikmesi en yüksek olan denemenin ortalama gecikmesi 1 olarak kabul edilerek bütün denemelerden elde edilen gecikmelerin buna göre normalize edilmesi sonucunda elde edilen jitter grafiği aşağıdaki şekilde gösterilmektedir.



Şekil 3.29. İkinci senaryo jitter grafiği (normalize edilmiş)

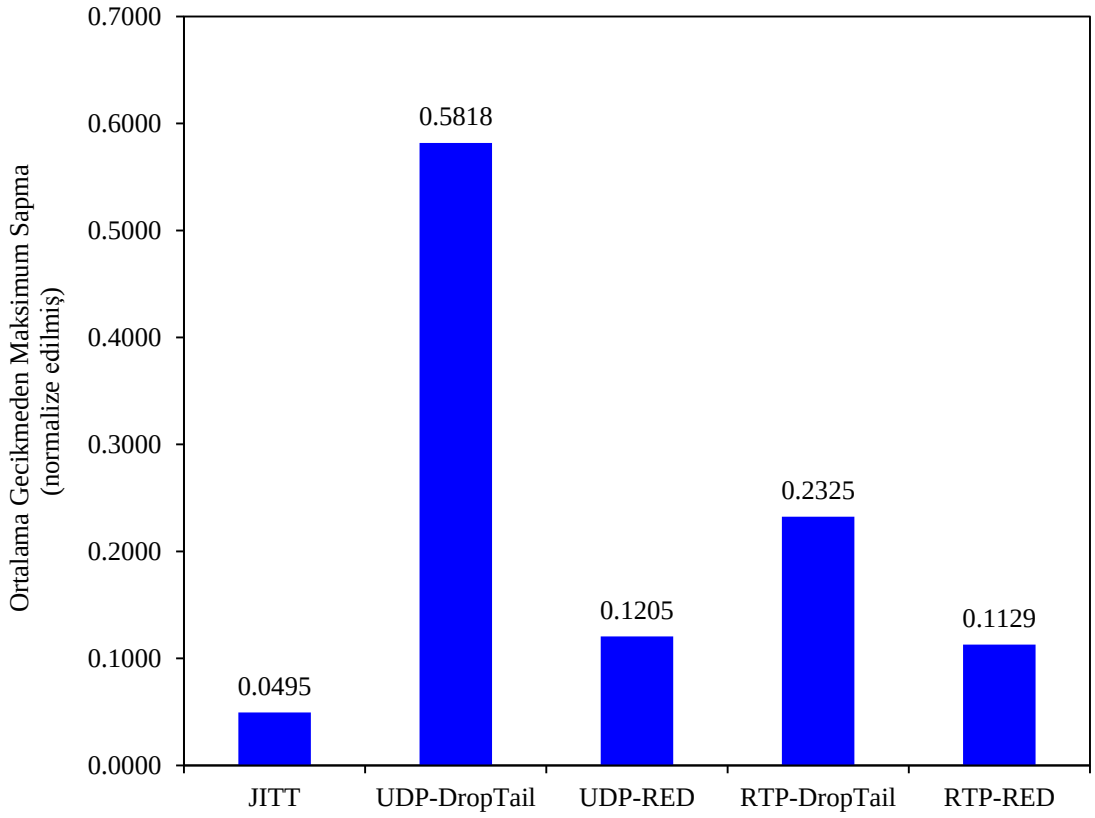
Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük jitter sağladığı görülmektedir.

Oynatma tamponunun belirlenmesinde önemli bir parametre olan ortalama gecikmeden maksimum sapma için elde edilmiş olan değerler Şekil 3.30'da gösterilmektedir.



Şekil 3.30. İkinci senaryo ortalama gecikmeden maksimum sapma grafiği

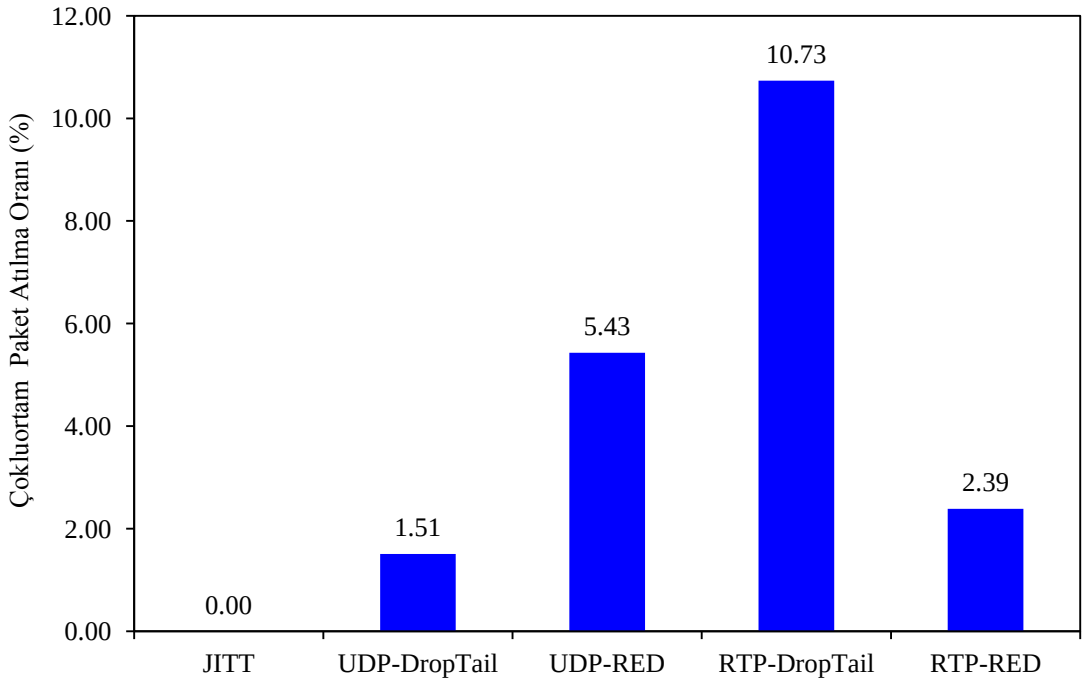
Ortalama gecikmesi en yüksek olan denemenin ortalama gecikmesinin 1 olarak kabul edilerek bütün denemelerden elde edilen ortalama gecikmeden maksimum sapma değerlerinin buna göre normalize edilmesi sonucunda elde edilen ortalama gecikmeden maksimum sapma grafiği aşağıdaki şekilde gösterilmektedir.



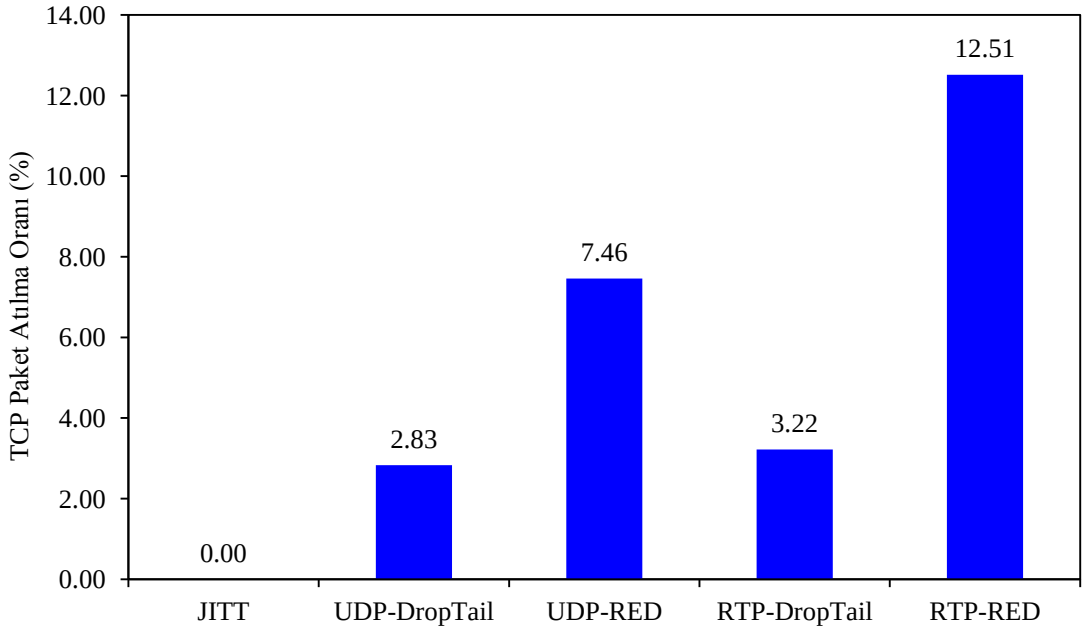
Şekil 3.31. İkinci senaryo ortalama gecikmeden maksimum sapma grafiği (normalize edilmiş)

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük ortalama gecikmeden maksimum sapma sağladığı görülmektedir.

Son olarak incelenen ölçüt, atılan veri miktarının üretilen veri miktarına oranıdır. Şekil 3.32'de GZÇO trafiğinden atılan veri oranları; Şekil 3.33'de TCP trafiğinden atılan veri oranları gösterilmektedir. İlgili şekillerden de görüldüğü gibi ikinci senaryoda JITT'in paket atılmalarını önlediği görülmektedir.



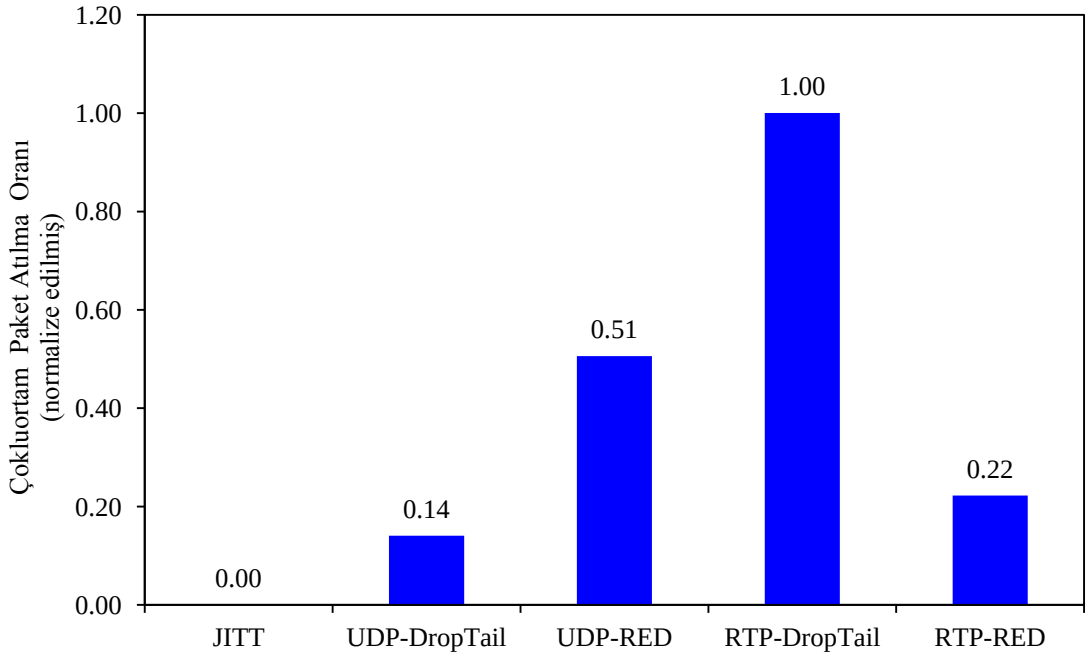
Şekil 3.32. İkinci senaryo GZÇO veri atılma oranları grafiği



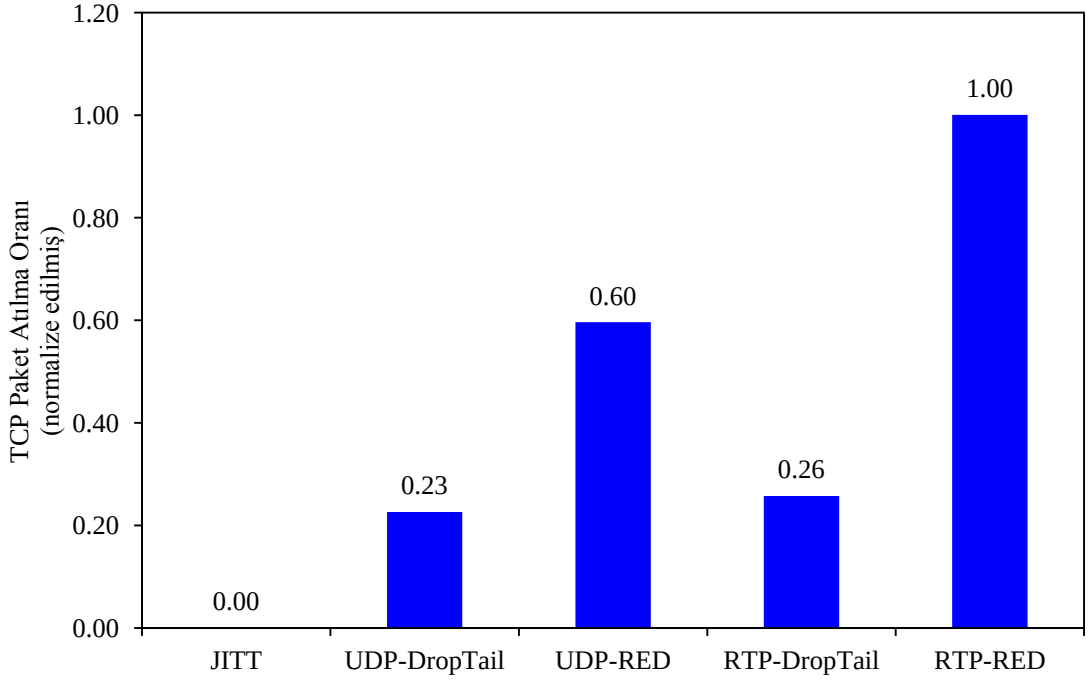
Şekil 3.33. İkinci senaryo TCP veri atılma oranları grafiği

Paket atılma oranı en yüksek olan denemenin paket atılma oranı 1 olarak kabul edilerek bütün denemelerden elde edilen paket atılma oranlarının buna göre

normalize edilmesi sonucunda elde edilen paket atılma oranları grafikleri aşağıdaki şekillerde gösterilmektedir.



Şekil 3.34. İkinci senaryo GZÇO veri atılma oranları grafiği (normalize edilmiş)



Şekil 3.35. İkinci senaryo TCP veri atılma oranları grafiği (normalize edilmiş)

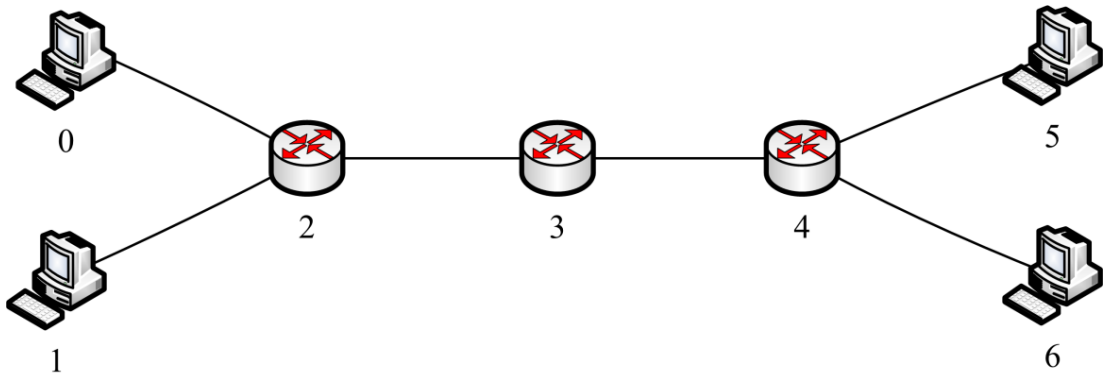
3.7.3. Üçüncü senaryo

Gerçekleştirilen benzetimde kullanılan parametreler aşağıdaki gibidir.

Çizelge 3.4. Üçüncü benzetimde kullanılan parametreler

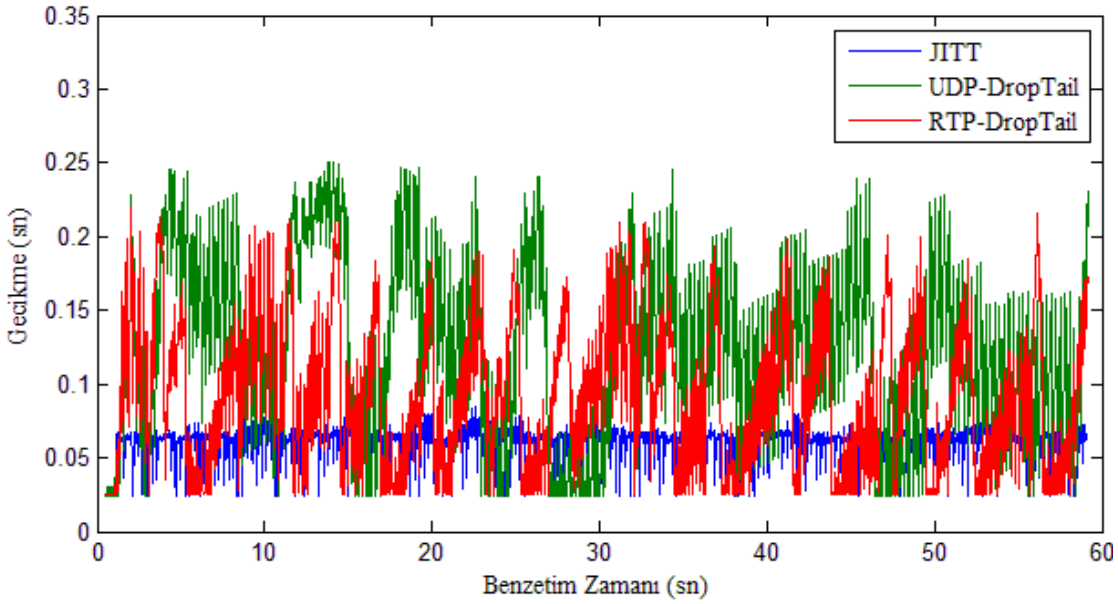
Benzetim Süresi					
65 sn					
Yönlendirici Olarak Görev Yapan Düğümler ve Aralarındaki Bağlantı Kapasitesi					
2-3-6, 0.5Mbps 10ms					
Trafik Tipi					
Video Konferans		TCP			
Haberleşen Düğümler		*Haberleşen Düğümler*			
0-5	5-0	1-6	6-1	0-5	5-0
Haberleşme Süresi	*Haberleşme Süresi*	*Haberleşme Süresi*		*Haberleşme Süresi*	
0.5-59 sn	0.5-59 sn	1-59 sn	1-59 sn	1-59 sn	1-59 sn

Üçüncü senaryonun topolojisi Şekil 3.36'daki gibidir.

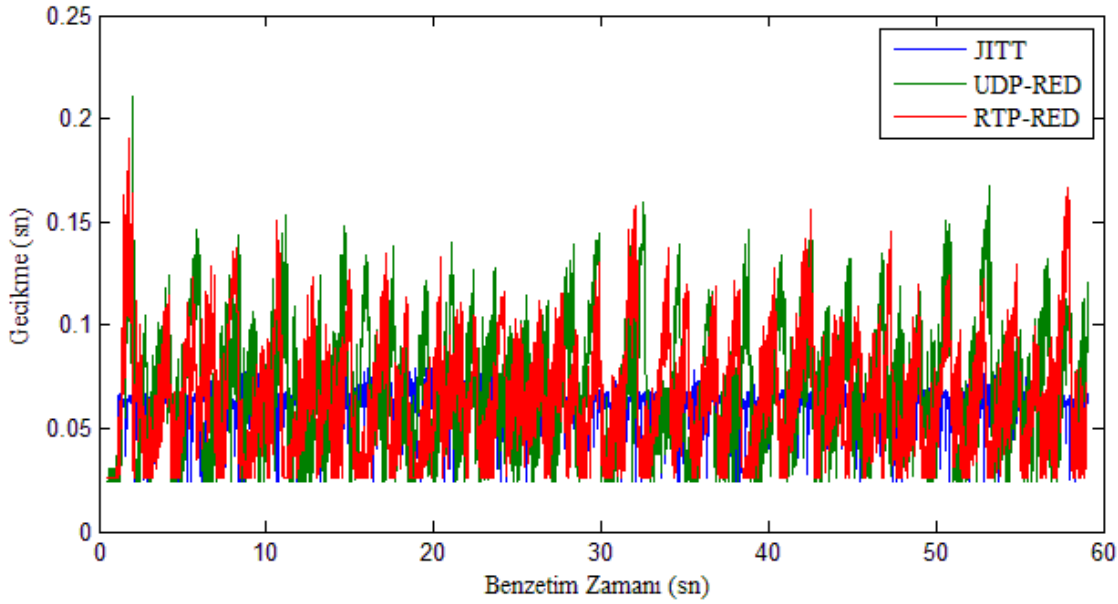


Şekil 3.36. Üçüncü senaryonun topolojisi

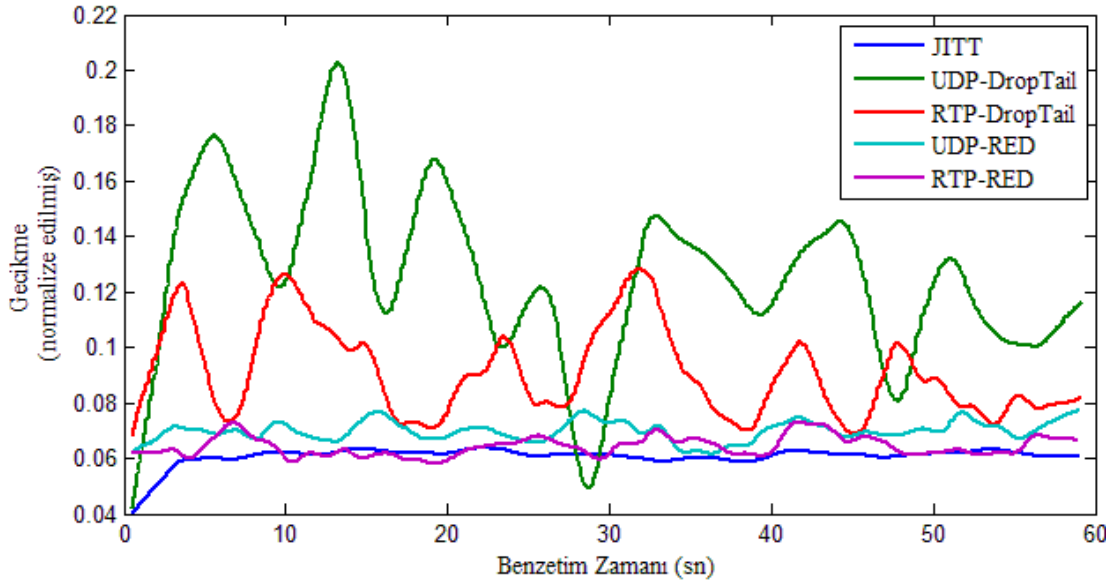
Üçüncü senaryo ile elde edilen gecikme grafikleri Şekil 3.37 ve Şekil 3.38'de; normalize edilmiş gecikme grafikleri ise Şekil 3.39'da gösterilmektedir.



Şekil 3.37. Üçüncü senaryo gecikme grafiği (JITT, UDP-DropTail, RTP-DropTail)



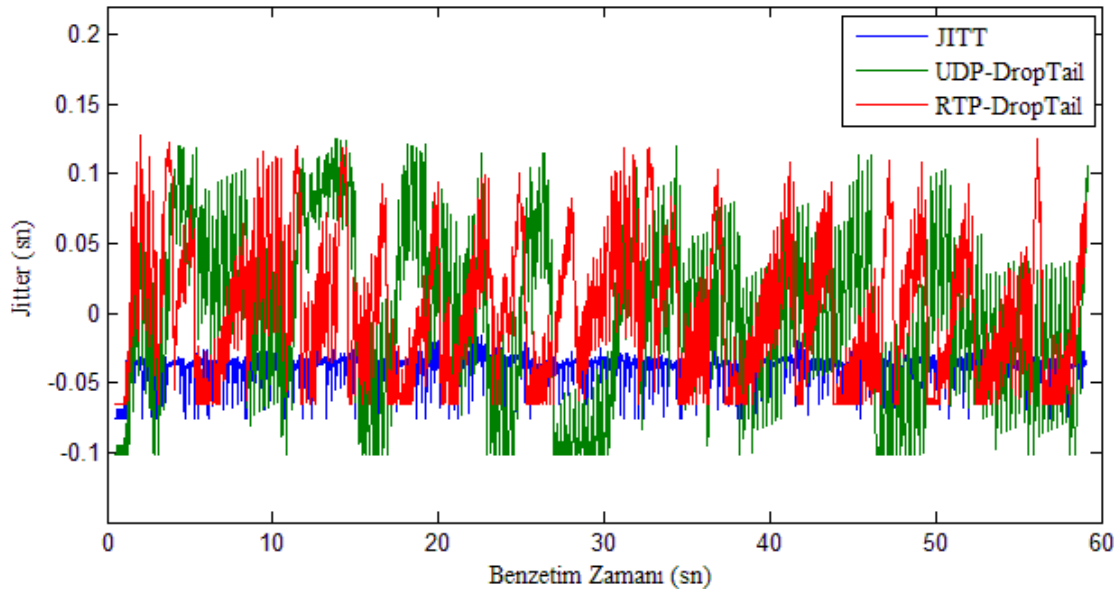
Şekil 3.38. Üçüncü senaryo gecikme grafiği (JITT, UDP-RED, RTP-RED)



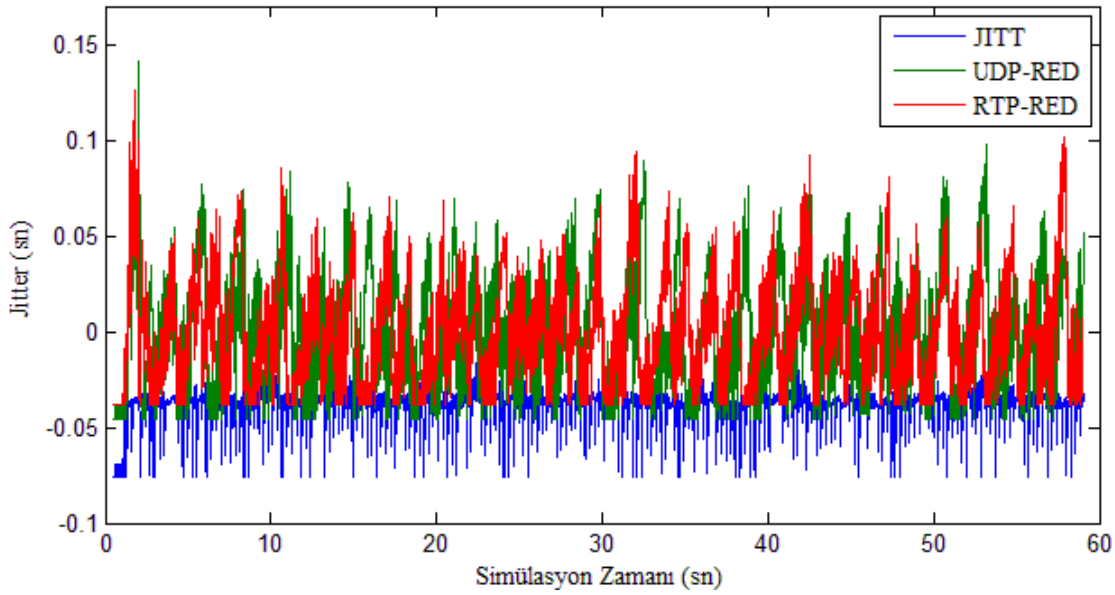
Şekil 3.39. Üçüncü senaryo normalize edilmiş gecikme grafiği

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük ve daha kararlı gecikme sağladığı görülmektedir.

Üçüncü senaryo ile elde edilen jitter grafikleri Şekil 3.40 ve Şekil 3.41'de gösterilmektedir.

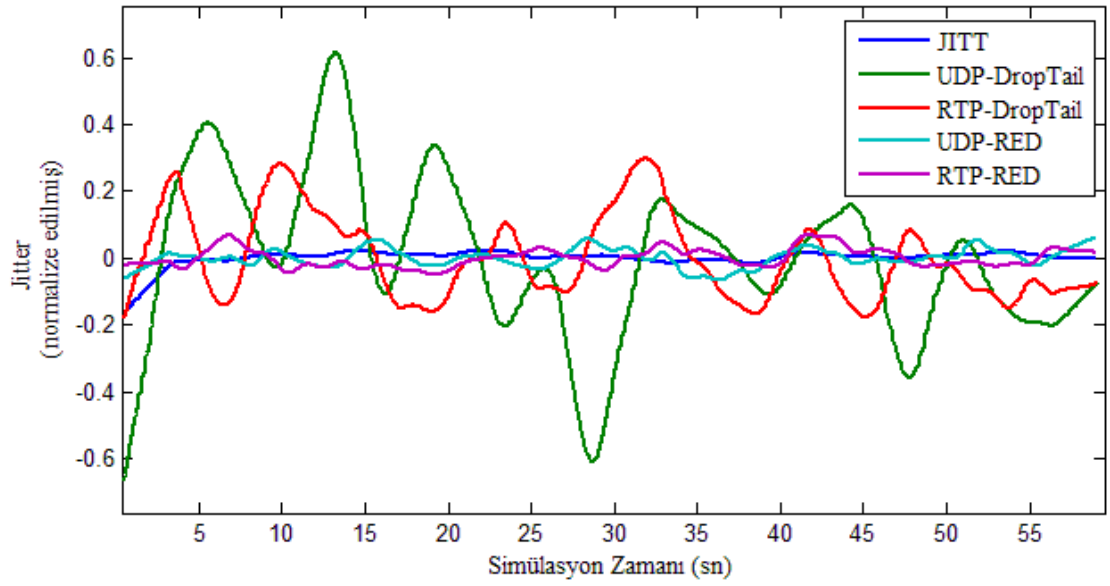


Şekil 3.40. Üçüncü senaryo jitter grafiği (JITT, UDP-DropTail, RTP-DropTail)



Şekil 3.41. Üçüncü senaryo jitter grafiği (JITT, UDP-RED, RTP-RED)

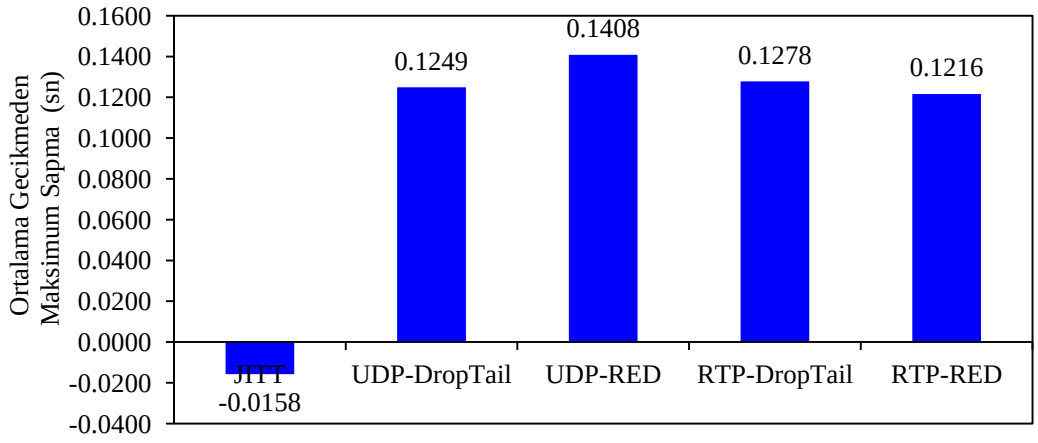
Ortalama gecikmesi en yüksek olan denemenin ortalama gecikmesi 1 olarak kabul edilerek bütün denemelerden elde edilen gecikmelerin buna göre normalize edilmesi sonucunda elde edilen jitter grafiği aşağıdaki şekilde gösterilmektedir.



Şekil 3.42. Üçüncü senaryo jitter grafiği (normalize edilmiş)

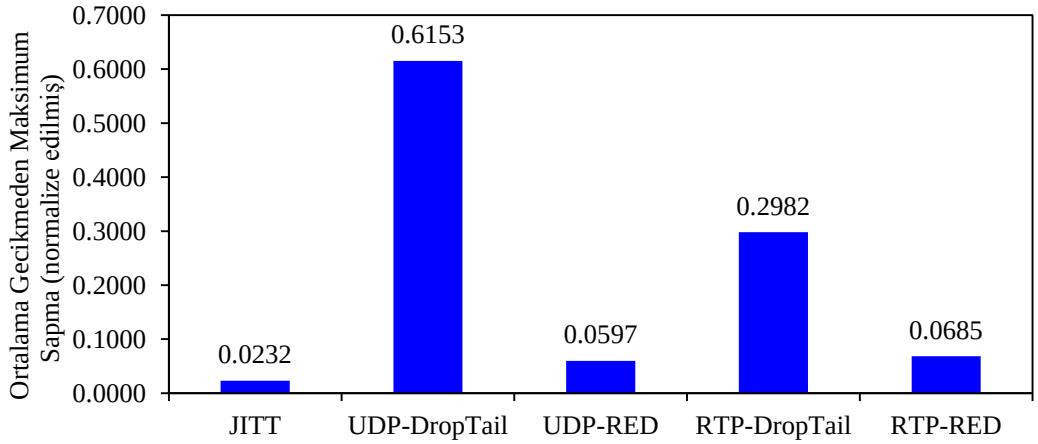
Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük jitter sağladığı görülmektedir.

Oynatma tamponunun belirlenmesinde önemli bir parametre olan ortalama gecikmeden maksimum sapma için elde edilmiş olan değerler Şekil 3.43'de gösterilmektedir.



Şekil 3.43. Üçüncü senaryo ortalama gecikmeden maksimum sapma grafiği

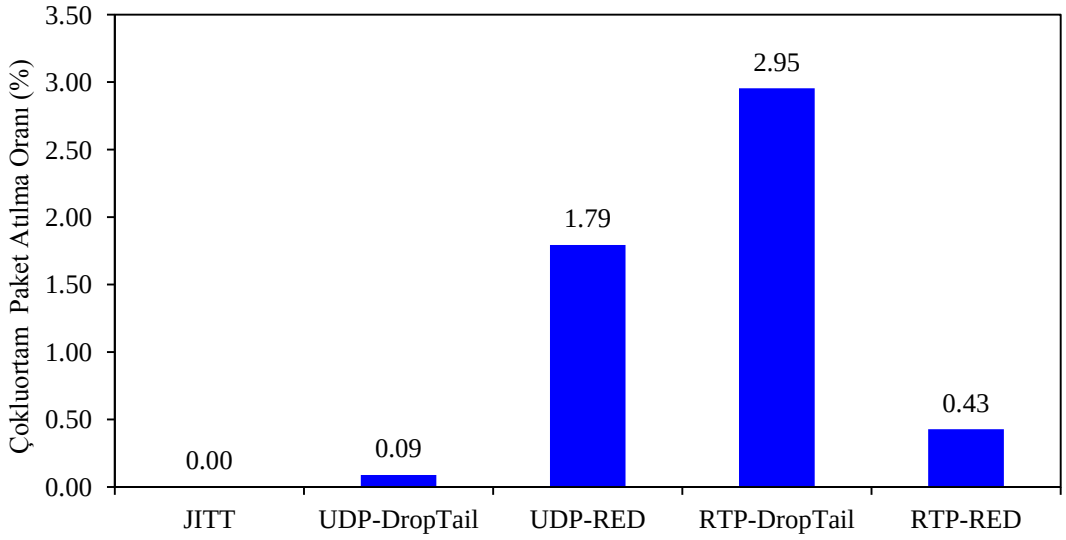
Ortalama gecikmesi en yüksek olan denemenin ortalama gecikmesinin 1 olarak kabul edilerek bütün denemelerden elde edilen ortalama gecikmeden maksimum sapma değerlerinin buna göre normalize edilmesi sonucunda elde edilen ortalama gecikmeden maksimum sapma grafiği aşağıdaki şekilde gösterilmektedir.



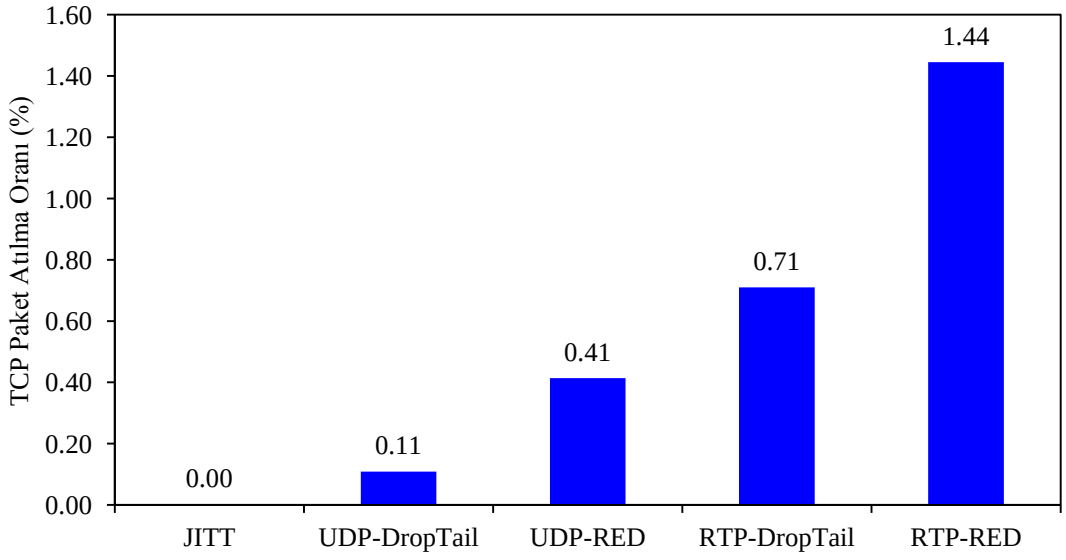
Şekil 3.44. Üçüncü senaryo ortalama gecikmeden maksimum sapma grafiği (normalize edilmiş)

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük ortalama gecikmeden maksimum sapma sağladığı görülmektedir.

Son olarak incelenen ölçüt, atılan veri miktarının üretilen veri miktarına oranıdır. Şekil 3.45'de GZÇO trafiğinden atılan veri oranları; Şekil 3.46'da TCP trafiğinden atılan veri oranları gösterilmektedir. İlgili şekillerden de görüldüğü gibi üçüncü senaryoda JITT'in paket atılmalarını önlediği görülmektedir.

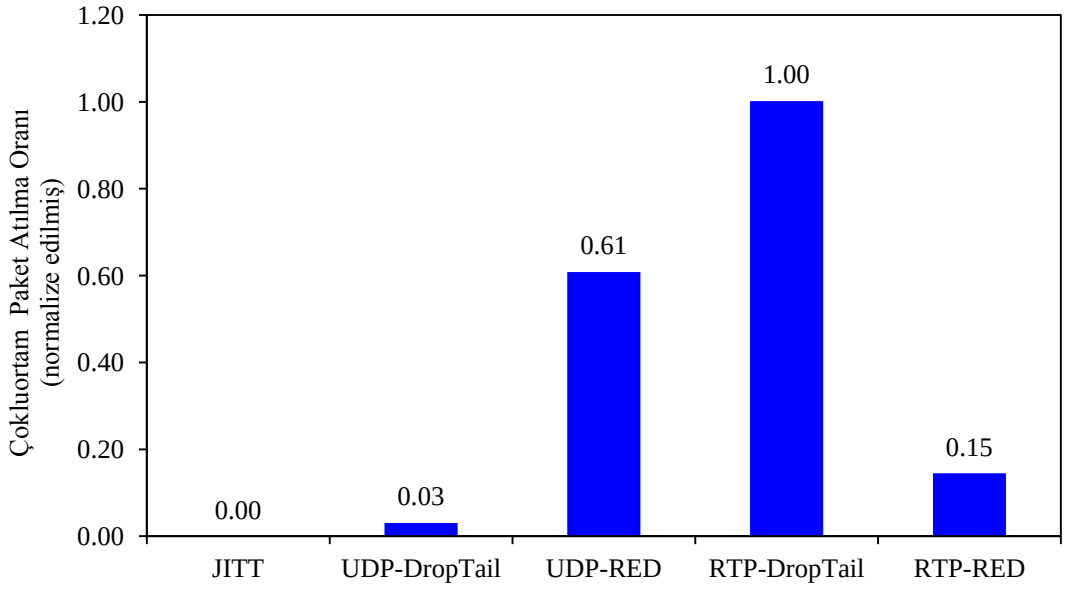


Şekil 3.45. Üçüncü senaryo GZÇO veri atılma oranları grafiği

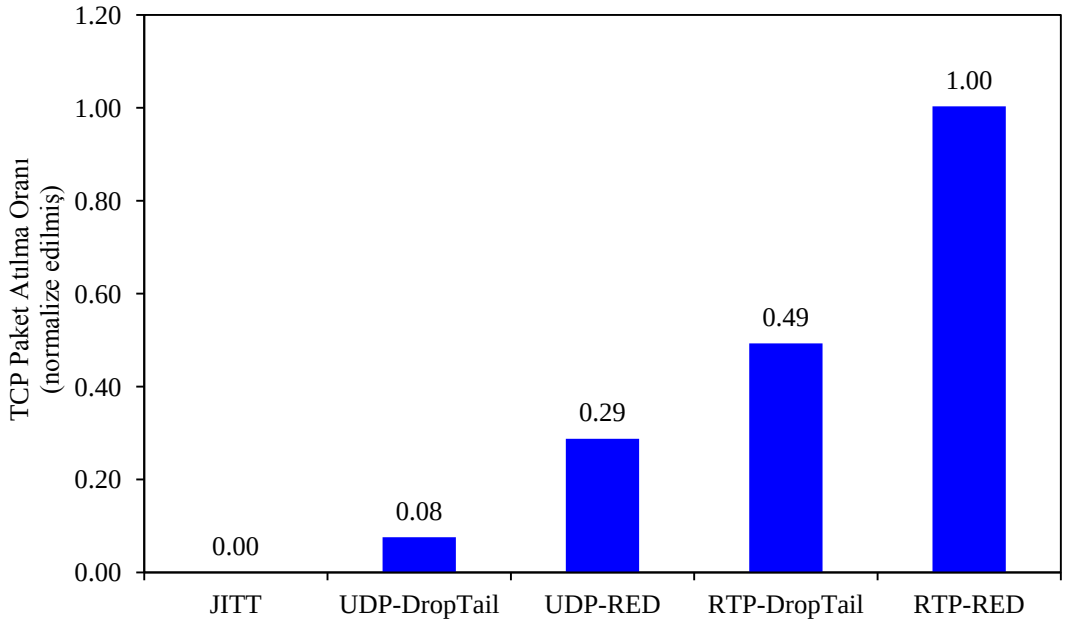


Şekil 3.46. Üçüncü senaryo TCP veri atılma oranları grafiği

Paket atılma oranı en yüksek olan denemenin paket atılma oranı 1 olarak kabul edilerek bütün denemelerden elde edilen paket atılma oranlarının buna göre normalize edilmesi sonucunda elde edilen paket atılma oranları grafikleri aşağıdaki şekillerde gösterilmektedir.



Şekil 3.47. Üçüncü senaryo GZÇO veri atılma oranları grafiği (normalize edilmiş)



Şekil 3.48. Üçüncü senaryo TCP veri atılma oranları grafiği (normalize edilmiş)

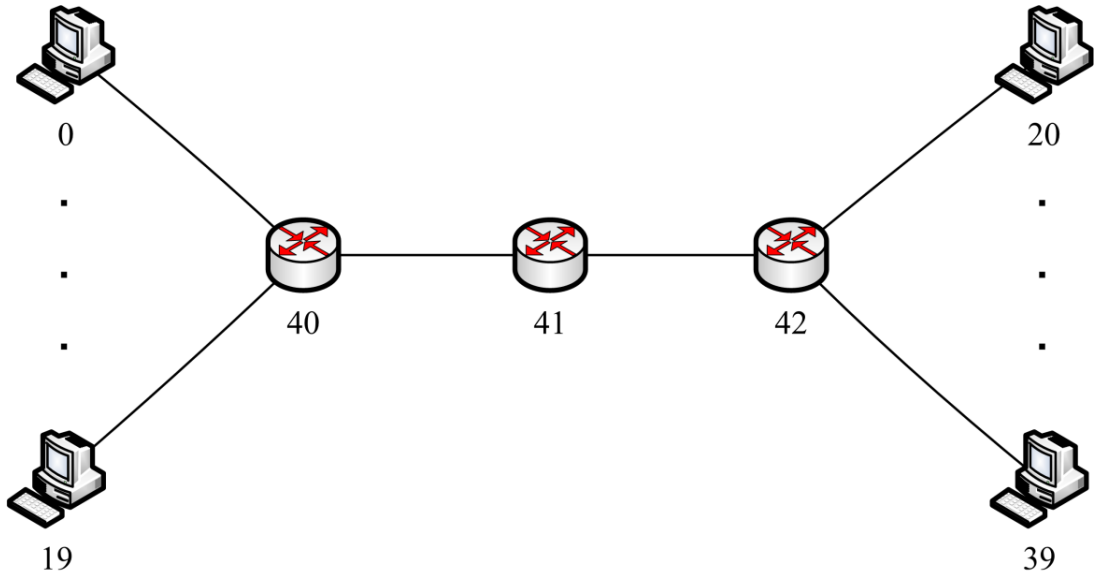
3.7.4. Dördüncü senaryo

Gerçekleştirilen benzetimde kullanılan parametreler aşağıdaki gibidir.

Çizelge 3.5. Dördüncü benzetimde kullanılan parametreler

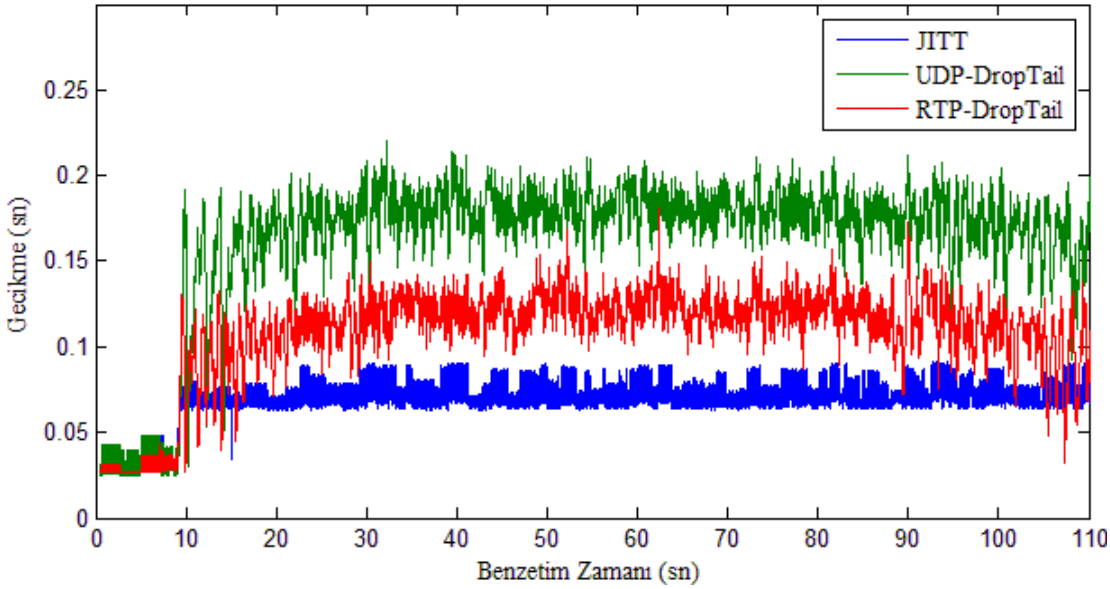
Benzetim Süresi				
120 sn				
Yönlendirici Olarak Görev Yapan Düğümler ve Aralarındaki Bağlantı Kapasitesi				
40-41-42, 2Mbps 10ms				
Trafik Tipi				
Video Konferans				TCP
Haberleşen Düğümler				*Haberleşen Düğümler*
0-20	20-0	21-1	1-21	2-19 arası düğümler ile 22-39 arası düğümler arasında toplam 18 adet bağlantı
Haberleşme Süresi		*Haberleşme Süresi*		*Haberleşme Süresi*
0.5-110 sn		0.5-110 sn		Bütün bağlantılar 5-105 sn'ler arasında 76'şar sn sürmüştür

Dördüncü senaryonun topolojisi Şekil 3.49'daki gibidir.

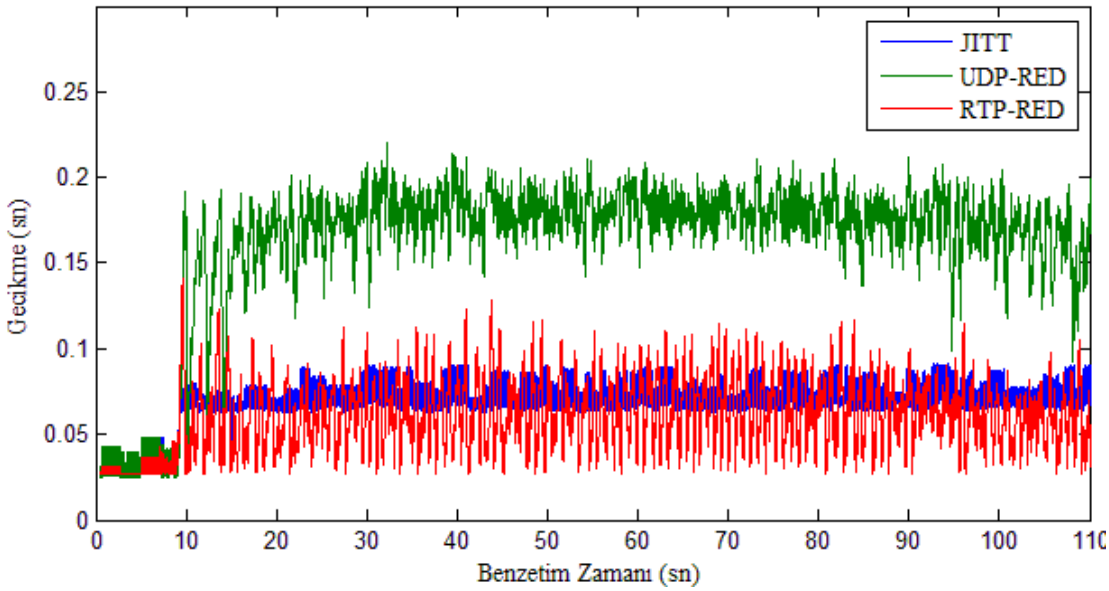


Şekil 3.49. Dördüncü senaryonun topolojisi

Dördüncü senaryo ile elde edilen gecikme grafikleri Şekil 3.50 ve Şekil 3.51'de gösterilmektedir.



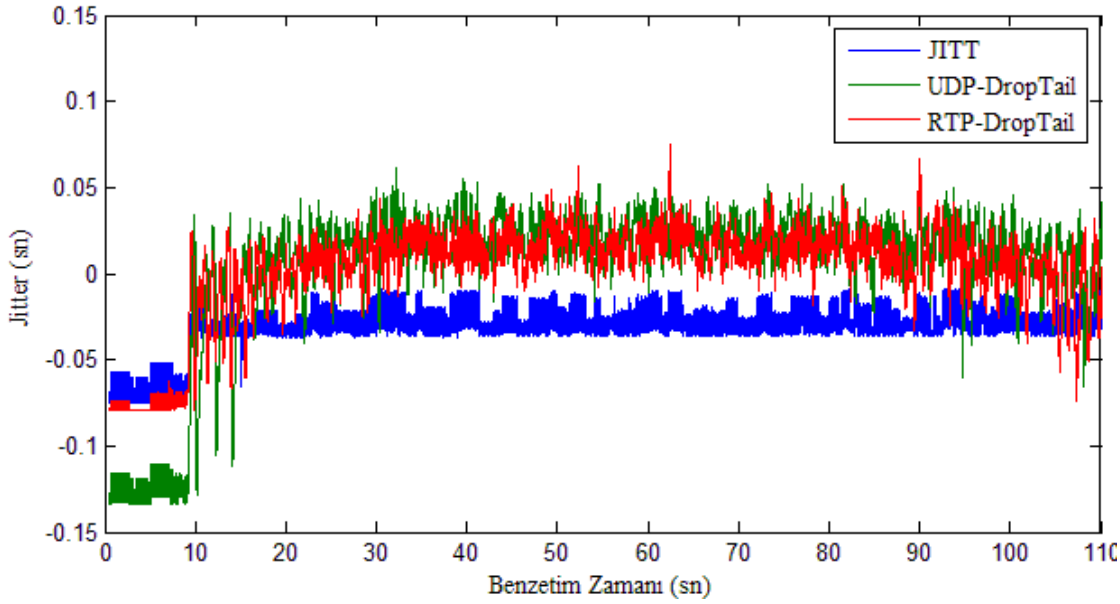
Şekil 3.50. Dördüncü senaryo gecikme grafiği (JITT, UDP-DropTail, RTP-DropTail)



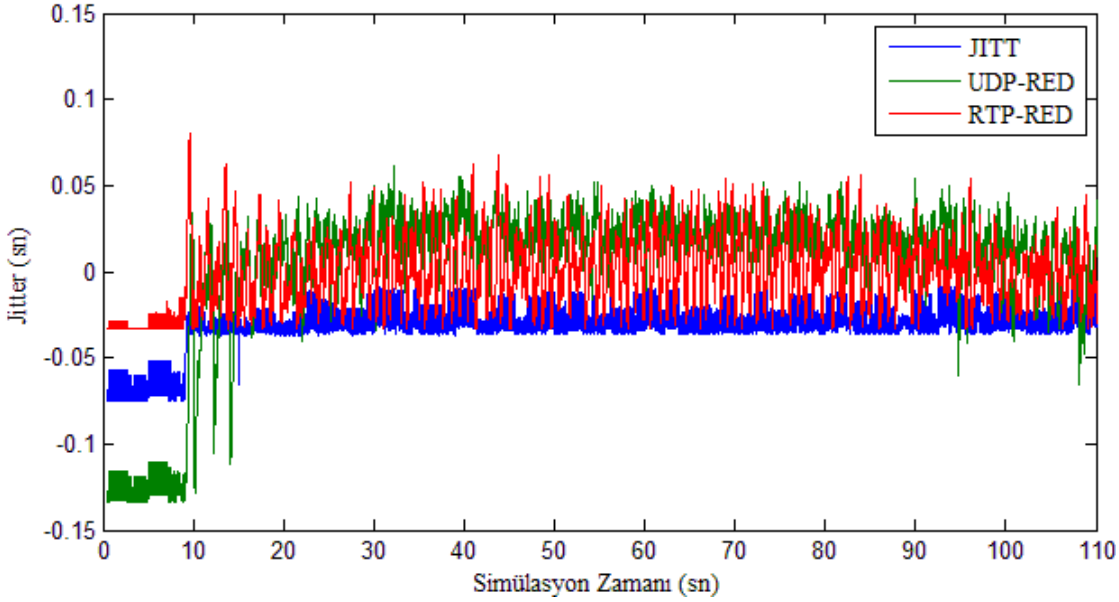
Şekil 3.51. Dördüncü senaryo gecikme grafiği (JITT, UDP-RED, RTP-RED)

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük ve daha kararlı gecikme sağladığı görülmektedir.

Dördüncü senaryo ile elde edilen jitter grafikleri Şekil 3.52 ve Şekil 3.53'de gösterilmektedir.

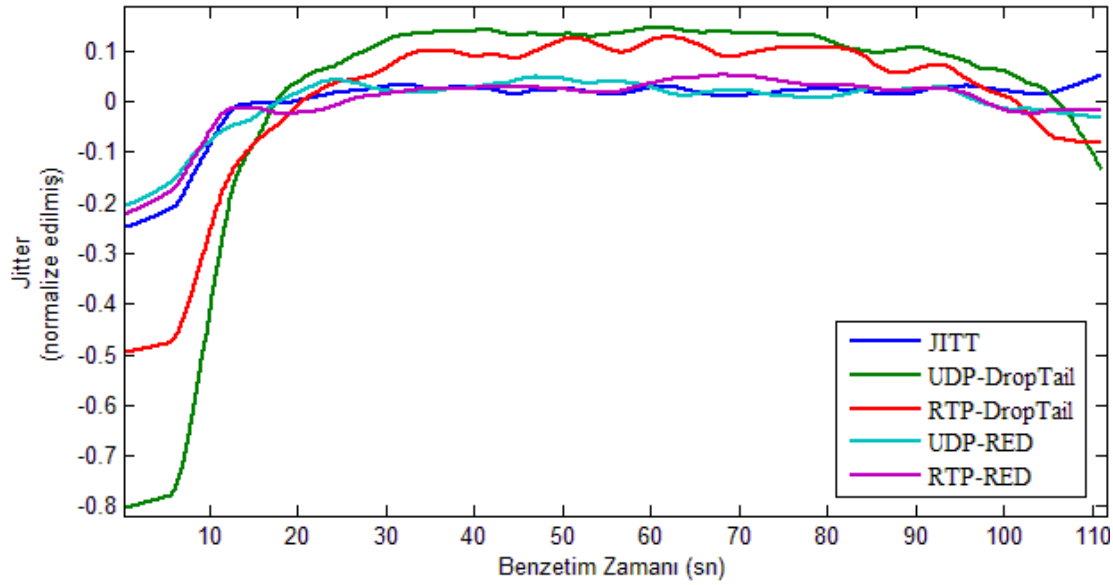


Şekil 3.52. Dördüncü senaryo jitter grafiği (JITT, UDP-DropTail, RTP-DropTail)



Şekil 3.53. Dördüncü senaryo jitter grafiği (JITT, UDP-RED, RTP-RED)

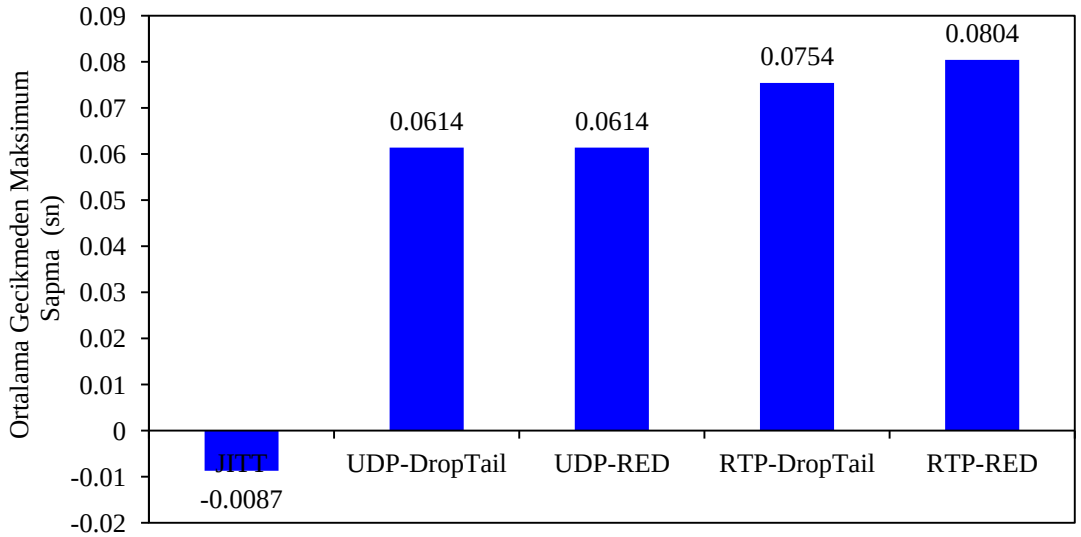
Ortalama gecikmesi en yüksek olan denemenin ortalama gecikmesi 1 olarak kabul edilerek bütün denemelerden elde edilen gecikmelerin buna göre normalize edilmesi sonucunda elde edilen jitter grafiği aşağıdaki şekilde gösterilmektedir.



Şekil 3.54. Dördüncü senaryo jitter grafiği (normalize edilmiş)

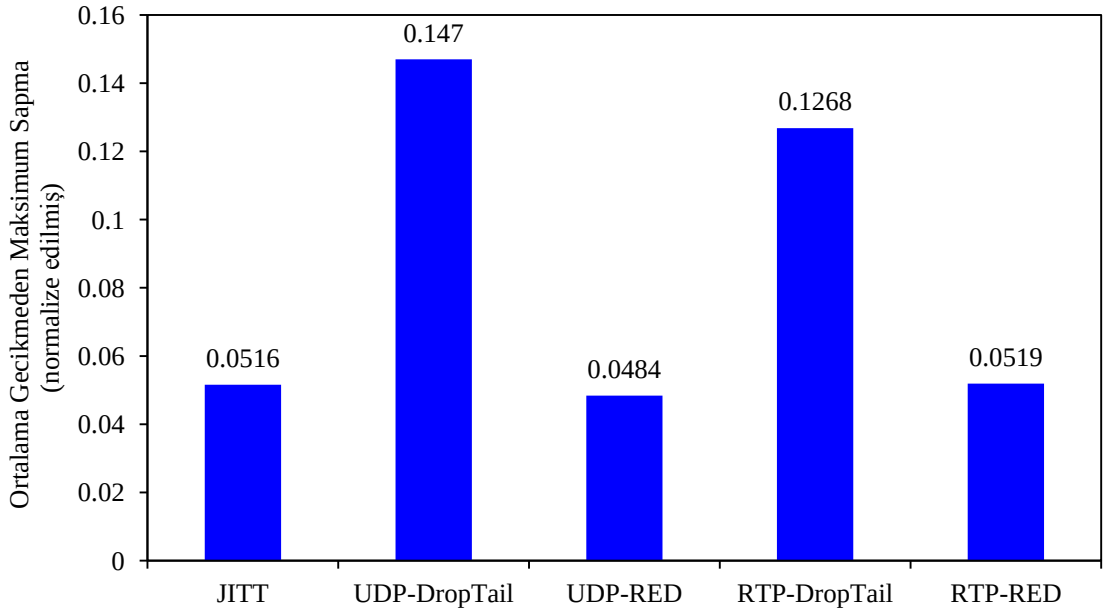
Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük jitter sağladığı görülmektedir.

Oynatma tamponunun belirlenmesinde önemli bir parametre olan ortalama gecikmeden maksimum sapma için elde edilmiş olan değerler Şekil 3.55'te gösterilmektedir.



Şekil 3.55. Dördüncü senaryo ortalama gecikmeden maksimum sapma grafiği

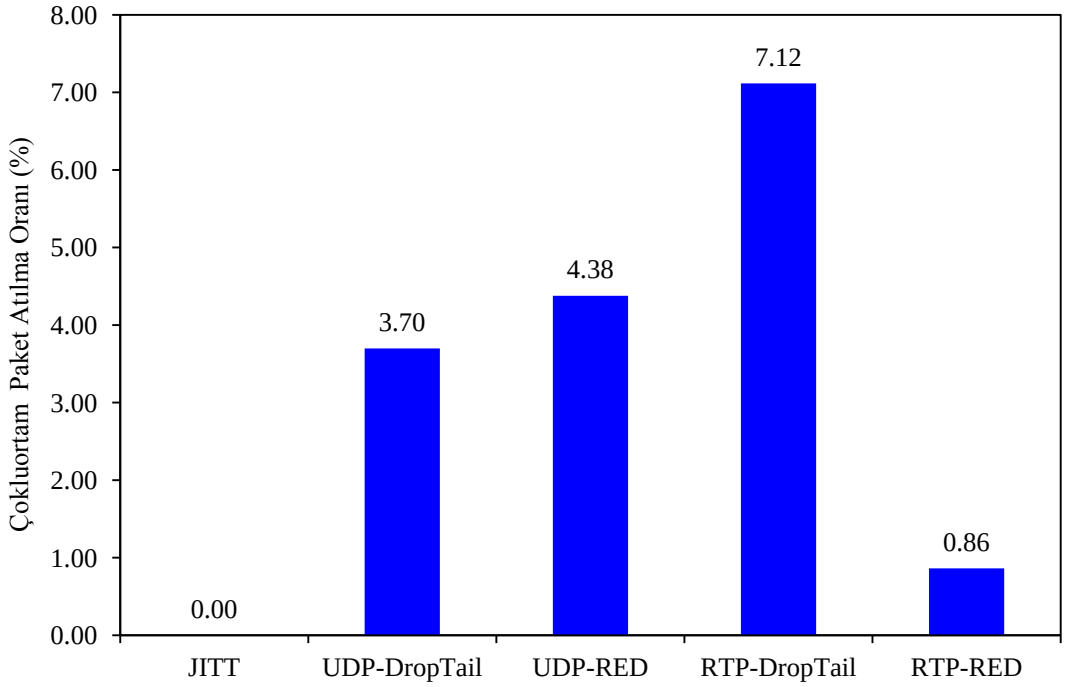
Ortalama gecikmesi en yüksek olan denemenin ortalama gecikmesinin 1 olarak kabul edilerek bütün denemelerden elde edilen ortalama gecikmeden maksimum sapma değerlerinin buna göre normalize edilmesi sonucunda elde edilen jitter grafiği aşağıdaki şekilde gösterilmektedir.



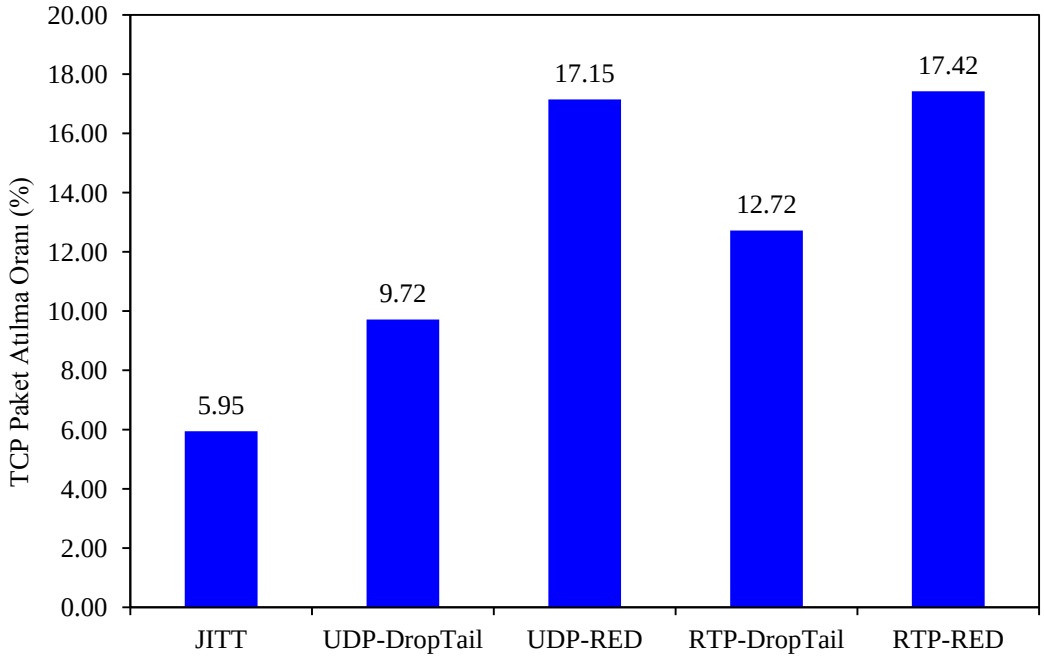
Şekil 3.56. Dördüncü senaryo ortalama gecikmeden maksimum sapma grafiği (normalize edilmiş)

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük ortalama gecikmeden maksimum sapma sağladığı görülmektedir.

Son olarak incelenen ölçüt, atılan veri miktarının üretilen veri miktarına oranıdır. Şekil 3.57'de GZÇO trafiğinden atılan veri oranları; Şekil 3.58'de TCP trafiğinden atılan veri oranları gösterilmektedir. JITT'in GZÇO paket atılmalarını önlediği ve bununla birlikte TCP paketlerinden ise diğer protokol-kuyruk yapısı çiftlerine göre çok daha az atılma oranı sağladığı söylenebilir.



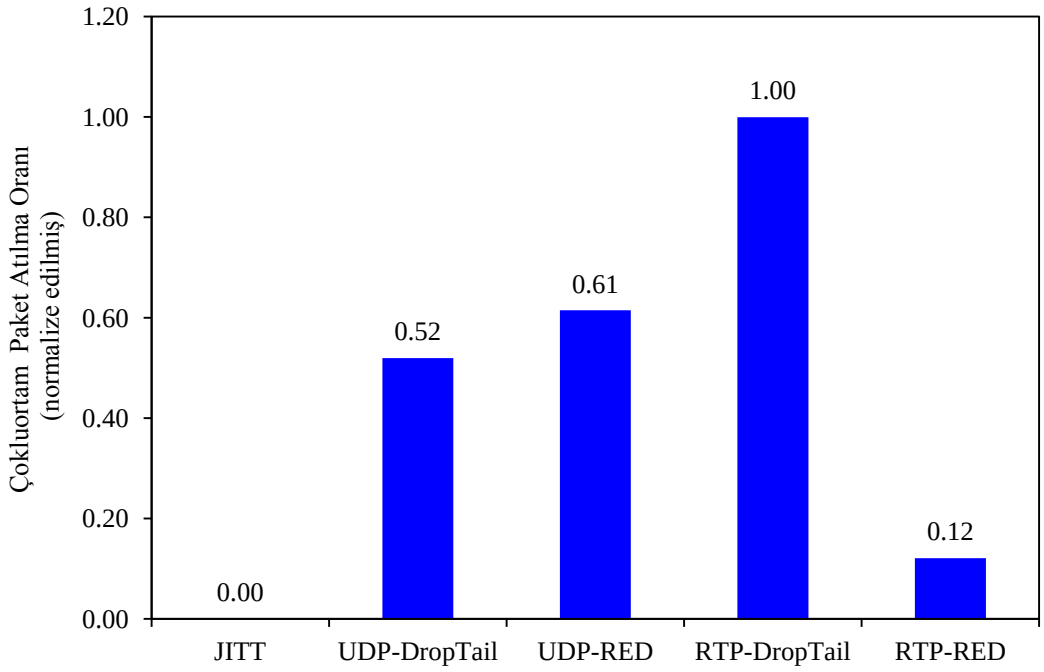
Şekil 3.57. Dördüncü senaryo GZÇO veri atılma oranları grafiği



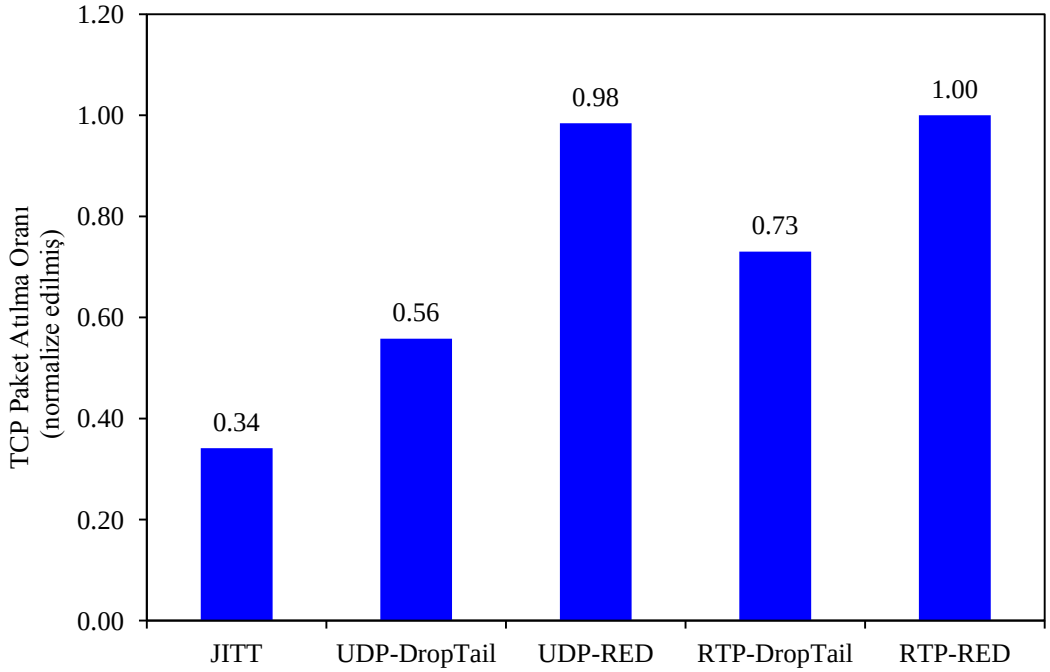
Şekil 3.58. Dördüncü senaryo TCP veri atılma oranları grafiği

Paket atılma oranı en yüksek olan denemenin paket atılma oranı 1 olarak kabul edilerek bütün denemelerden elde edilen paket atılma oranlarının buna göre

normalize edilmesi sonucunda elde edilen paket atılma oranları grafikleri aşağıdaki şekillerde gösterilmektedir.



Şekil 3.59. Dördüncü senaryo GZÇO veri atılma oranları grafiği (normalize edilmiş)



Şekil 3.60. Dördüncü senaryo TCP veri atılma oranları grafiği (normalize edilmiş)

JITT'in hangi senaryoda hangi performans kriterinde en iyi sonucu verdiğini görebilmek amacıyla Çizelge 3.6 oluşturulmuştur. Gecikme parametresi için en iyi sonucun 4. Senaryoda, jitter ve ortalama gecikmeden maksimum sapma (OGMS) parametreleri için en iyi sonucun 3. senaryoda, GZÇO veri atılma oranı parametresi için en iyi sonucun bütün senaryolarda ve arka plan trafiği veri atılma oranı parametresi için en iyi sonucun 1-3 senaryolarında elde edildiği görülmektedir.

Çizelge 3.6. Benzetim senaryolarına göre performans kriterlerinin değerleri

<table>
<tr><th></th><th>1. Senaryo</th><th>2. senaryo</th><th>3. Senaryo</th><th>4. Senaryo</th></tr>
<tr><td>Gecikme</td><td>50 ms – 75 ms</td><td>60 ms – 75 ms</td><td>45ms – 70 ms</td><td>20 ms – 75 ms</td></tr>
<tr><td>Jitter</td><td>-20 ms – +5 ms</td><td>-10 ms – + 5 ms</td><td>-15ms – + 2 ms</td><td>-25 ms – 5 ms</td></tr>
<tr><td>O.G.M.S.</td><td>0,0626</td><td>0,0495</td><td>0,0232</td><td>0,0516</td></tr>
<tr><td>GZÇO Veri Atılma Oranı</td><td colspan="4">0</td></tr>
<tr><td>Arka Plan Trafiği Veri Atılma Oranı</td><td colspan="3">0</td><td>5,95</td></tr>
</table>

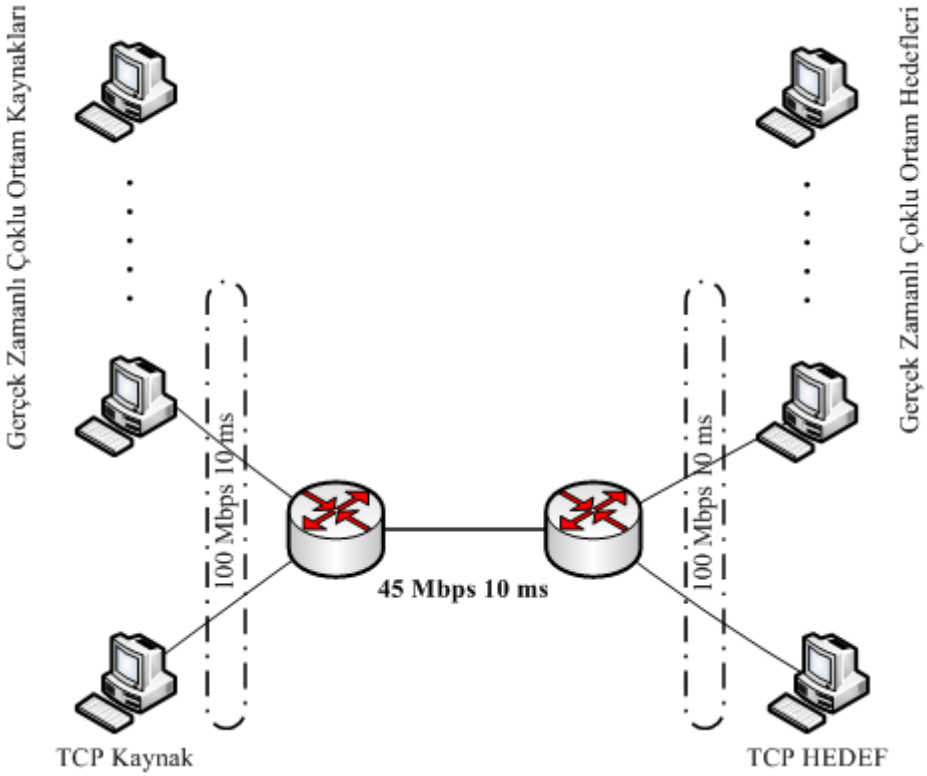
3.8. Geliştirilen Ulaştırma Protokolü ve Kuyruk Modeli'nin Ölçeklenebilirliği

JITT protokol ve kuyruk yapısının analitik, benzetim ve uygulama değerlendirmeleri önceki bölümlerde verilmiştir. Bu bölümde, GZÇO akış sayısı ile TCP performansının değerlendirilmesi için bir çalışma yapılmıştır. Gerçekleştirilen çalışmada farklı sayıdaki GZÇO bağlantı sayıları için TCP trafiğinin ve GZÇO bağlantılarının paket ulaştırılma oranları incelenmiştir. ns2 simülatörü ile yapılan çalışmanın parametreleri Çizelge 3.7'de gösterilmektedir. Benzetimlerde tıkanıklık durumunda bant genişliğini TCP ve JITT akışları arasında eşit olarak paylaştıran bir bağlantı zamanlayıcısı kullanılmıştır.

Çizelge 3.7. Ölçeklendirme senaryosunda kullanılan parametreler

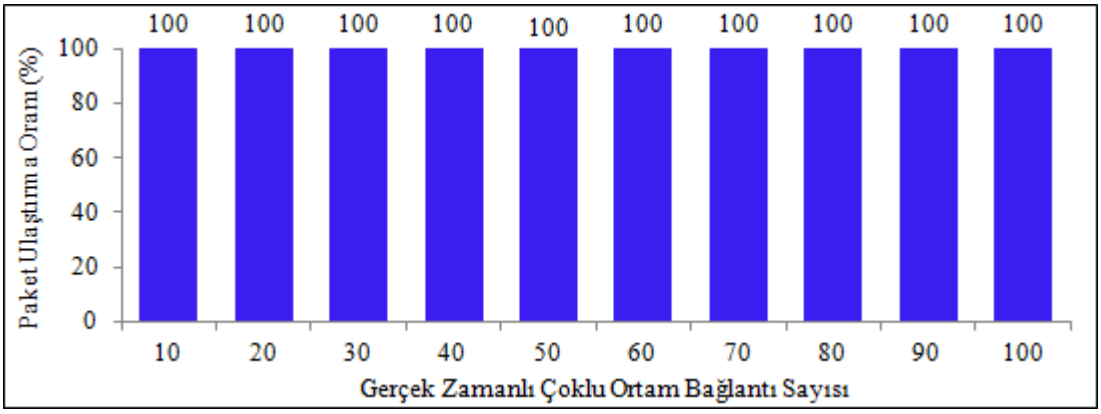
Benzetim Süresi	
120 sn	
Yönlendiriciler Aralarındaki Bağlantı Kapasitesi	
45Mbps, 10 ms	
Yönlendiriciler İle Bilgisayarlar Arasındaki Bağlantı Kapasiteleri	
100 Mbps, 10 ms	
Trafik Tipi	
GZÇO Trafiği	*TCP*
528 Kbps Exp., burst time 0.9, idle time 0.1	TCP NewReno
10,20,30,40,50,60,70,80,90,100 bağlantı	1 bağlantı

Gerçekleştirilen çalışmanın topolojisi Şekil 3.61'de gösterilmektedir.



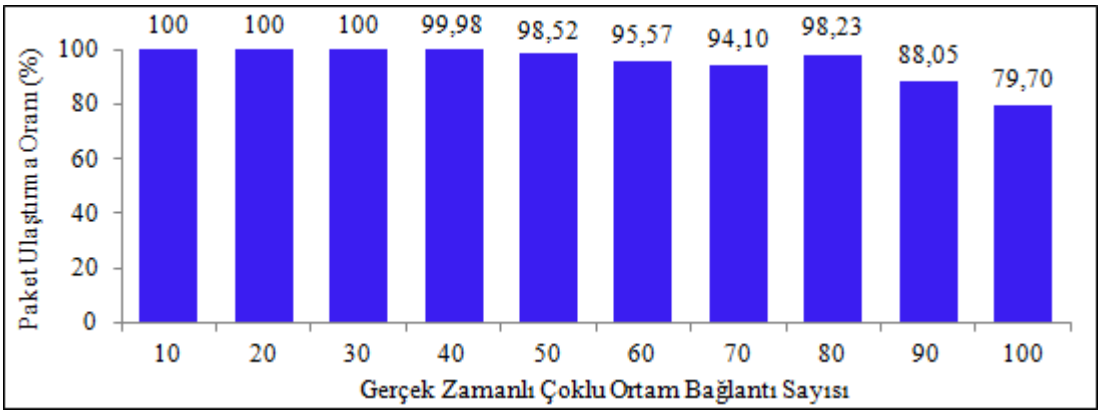
Şekil 3.61. Topoloji

Her seferinde GZÇO bağlantı sayısı 10'ar 10'ar arttırılarak benzetim sonuçları elde edilmiştir. Gerçekleştirilen denemeler sonucunda elde edilen TCP trafiğine ait paket ulaştırma oranları Şekil 3.62'de gösterilmektedir.



Şekil 3.62. TCP trafiği paket ulaştırma oranları

Şekil 3.62'den de görüleceği gibi gerçek zamanlı bağlantı sayısının artması ile birlikte TCP paketlerinin ulaştırma oranında bir değişiklik olmamış ve %100 olarak ölçülmüştür. Bunun nedeni, tıkanıklık durumunda bant genişliği paylaşımı yapan bağlantı zamanlayıcısının kullanılmasıdır. Gerçekleştirilen denemeler sonucunda elde edilen GZÇO trafiklerine ait paket ulaştırma değerleri Şekil 3.63'de gösterilmektedir.



Şekil 3.63. GZÇO trafikleri paket ulaştırma oranları

GZÇO trafiklerinin ortalama throughput değerleri 80 bağlantı sayısına kadar anlamlı bir değişikliğe uğramamış, 90 ve 100 bağlantı sayılarında ise düşüş yaşamıştır. Bunun nedeni, 90 ve 100 GZÇO bağlantı sayıları için yönlendiricilerin girişindeki trafik oranının yönlendiriciler arasındaki bağlantı kapasitesi olan 45 Mbps'den büyük olması ve bu durumda bağlantı zamanlayıcısının bant genişliğini paylaştırmasıdır. 60

ve 70 GZÇO bağlantı sayıları için GZÇO paket ulaştırma oranlarında bir azalış görülmektedir. Bu bağlantı sayıları ile gerçekleştirilen denemelerde TCP bağlantısının throughput değerinin diğer denemelere göre daha yüksek olduğu gözlemlenmiştir. Bu nedenle TCP paketlerinin ulaştırma oranları bütün denemeler için %100 olsa da, 60 ve 70 GZÇO bağlantı sayıları için GZÇO paket ulaştırma oranları 10-50 GZÇO bağlantı sayıları ile gerçekleştirilen denemelere göre düşük çıkmıştır.

3.9. Geliştirilen Ulaştırma Protokolü ve Kuyruk Modelinin Performansının Uygulama Ortamında Değerlendirilmesi

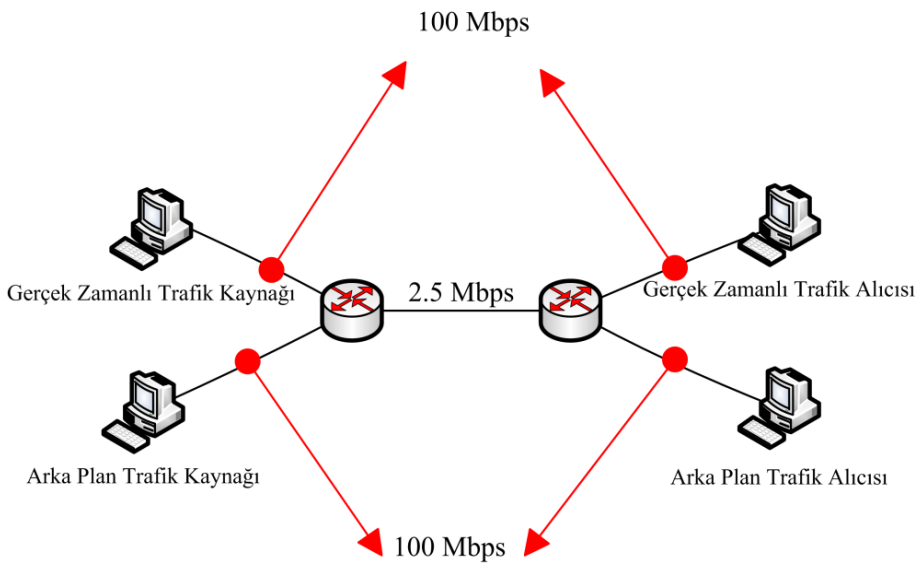
JITT'in performansının uygulama ortamında test edilmesi için Şekil 3.64'deki gibi bir topoloji oluşturulmuştur. Topoloji, literatürde kuyruk yönetimi algoritmalarının ve gerçek zamanlı trafik performanslarının test edilmesi için oluşturulan topolojiler dikkate alınarak oluşturulmuştur [21,25,47]. Gerçek zamanlı trafik kaynağının bütün denemelerde aynı olması için Planet Earth-Çöller belgeselinden 9:58'lik bir bölümün H.264 standardı ile sıkıştırılması ile elde edilen ses+video bileşenleri kullanılmıştır. H.264, ITU ve ISO'nun yeni video kodlama standardıdır ve iyileştirilmiş sıkıştırma özelliği nedeni ile video telefon, yayın ve video akışları için iyi bir performans sağlamaktadır [48,49,50]. Kullanılan videonun özellikleri Çizelge 3.8'de gösterilmektedir.

Çizelge 3.8. Uygulamada kullanılan videonun özellikleri

Süre	
598 sn	
Görüntü Bileşeni	*Ses Bileşeni*
640x480, 533 Kbps, 30 fps	2 kanal, 132 Kbps

Oluşturulan topolojide 2 farklı arka plan trafik yoğunluğu için denemeler gerçekleştirilmiştir. Arka plan trafiği ağ üzerinde sürekli bulunan bir trafiktir. Bu tip bir trafiğin kullanılmasının nedeni ağ üzerinde bulunabilecek diğer olası trafiklerin temsil edilmesidir. Her bir deneme DropTail-UDP, DropTail-RTP, RED-UDP, RED-RTP ve JITT protokol-kuyruk yapısı çiftleri için denenmiştir. JITT protokolü için *Delay* değeri 40ms olarak belirlenmiştir. GZÇO akışlarının performansını

karşılaştırmak için kullanılan temel parametre tek yönlü gecikmedir. Tek yönlü gecikmenin gerçek uygulamada ölçülmesi benzetimlerdeki gibi basit değildir. Literatürde tek yönlü gecikmenin ölçülmesi ile ilgili birçok çalışma bulunmaktadır [51-56]. Bu çalışmalarda temel olarak tek yönlü gecikme iki kısma ayrılmış ve ayrı ayrı ölçülmeye çalışılmıştır. Bunlar: sabit gecikme ve kuyruk gecikmesidir [57]. Kuyruk gecikmesi değişken olduğundan ölçülmesi zordur. Bu nedenle uçtan uca toplam gecikmenin ölçülmesi yerine hattaki sabit gecikmelerin bir defaya mahsus olmak üzere ölçülerek, kuyruk gecikmelerinin de yönlendiriciler üzerinde ölçülmesi ile elde edilen gecikmelerin toplamı alınmaktadır. Bu çalışmada izlenen yöntem de budur. [57] çalışmasında önerilen yöntem temel alınarak gerçekleştirilen sabit gecikme ölçümleri sonucunda gerçek zamanlı trafik kaynağı ile alıcısı arasındaki hattın 0,848 ms tek yönlü sabit gecikmeye sahip olduğu tespit edilmiştir. Deneme aşamasındaki kuyruk gecikmeleri ise her iki yönlendirici üzerinde ayrı ayrı yapılmış olup toplam gecikme elde edilmiştir. Cisco'nun yönlendiricileri üzerindeki gecikme, jitter ve paket kayıp oranı ölçümü için kullanmakta olduğu Service Assurance Agent (SSA) ve Round Trip Time Monitor (RTTMON) servisleri de yukarıda anlatılan yöntemi uygulamaktadır [58]. Denemelerde, tıkanıklık durumunda bant genişliğini arka plan trafiği ve JITT akışları arasında eşit olarak paylaştıran bir bağlantı zamanlayıcısı kullanılmıştır.



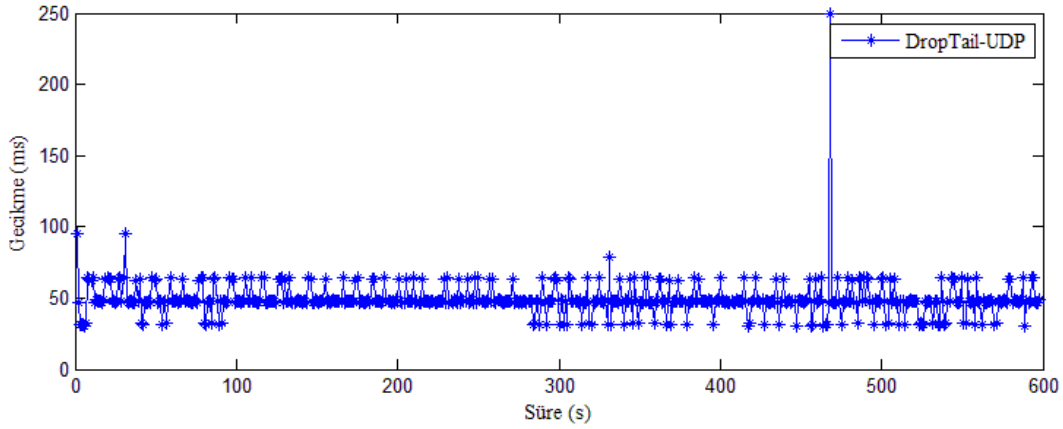
Şekil 3.64. Uygulama topolojisi

Uygulamada kullanılan parametreler aşağıdaki gibidir.

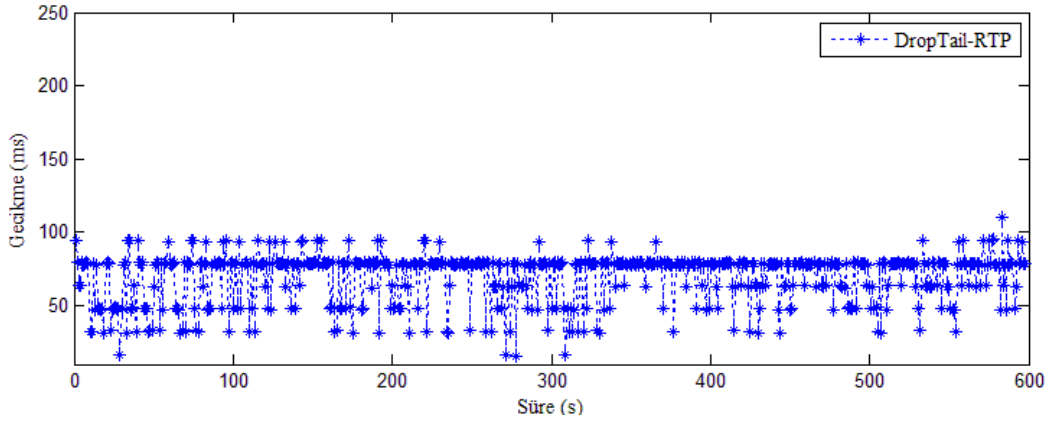
Çizelge 3.9. Uygulamada kullanılan parametreler

Uygulama Süresi	
598 sn	
Yönlendiriciler Aralarındaki Bağlantı Kapasitesi	
2.5Mbps	
Trafik Tipi	
GZÇO Trafiği	*Arka Plan Trafiği*
635 Kbps H.264 Görüntü ve Ses	1 Mbps ve 2 Mbps

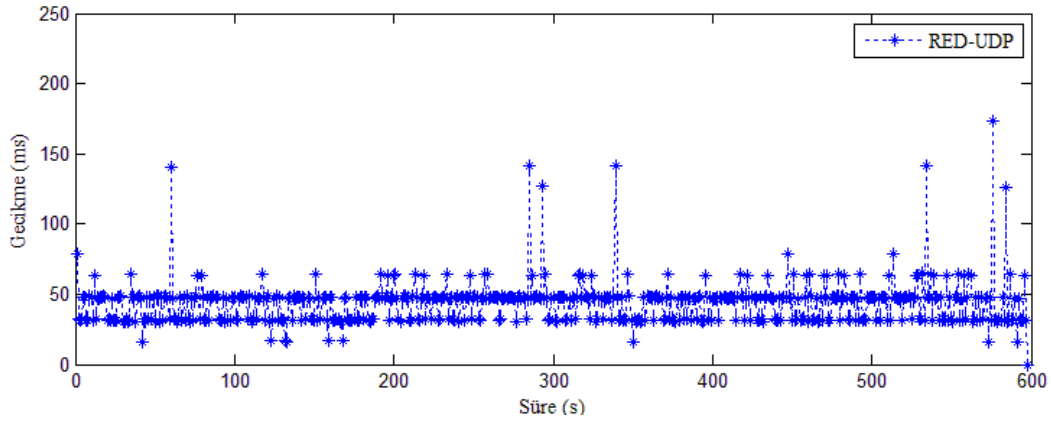
1 Mbps arka plan trafiği kullanılarak gerçekleştirilen deneme ile elde edilen gecikme grafikleri Şekil 3.65 – Şekil 3.69'larda gösterilmektedir.



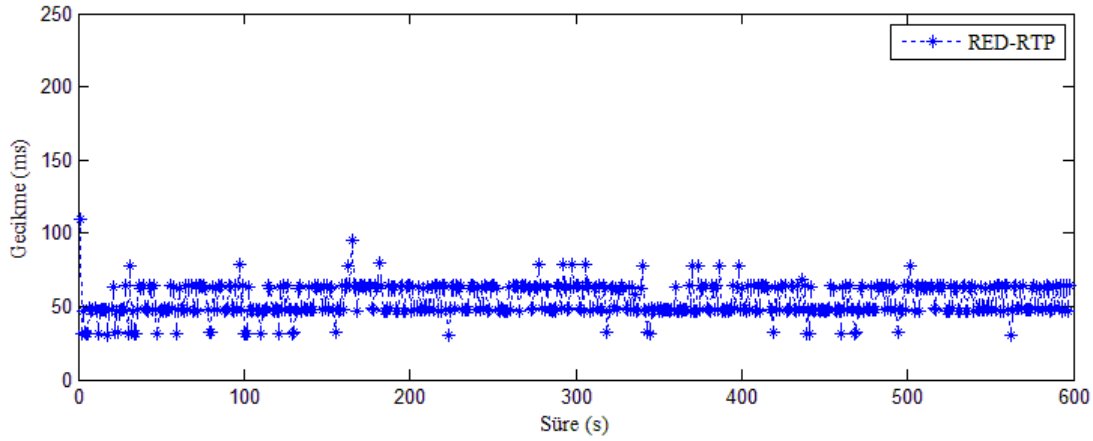
Şekil 3.65. DropTail-UDP ile elde edilen gecikme



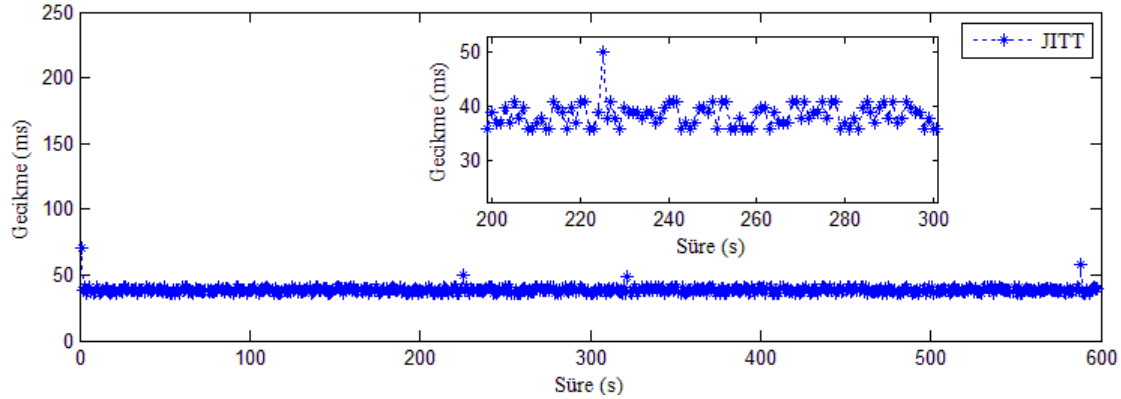
Şekil 3.66. DropTail-RTP ile elde edilen gecikme



Şekil 3.67. RED-UDP ile elde edilen gecikme



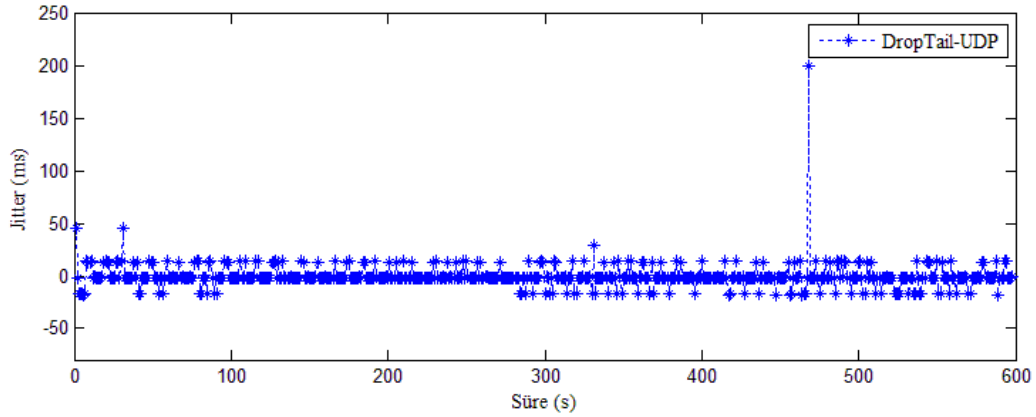
Şekil 3.68. RED-RTP ile elde edilen gecikme



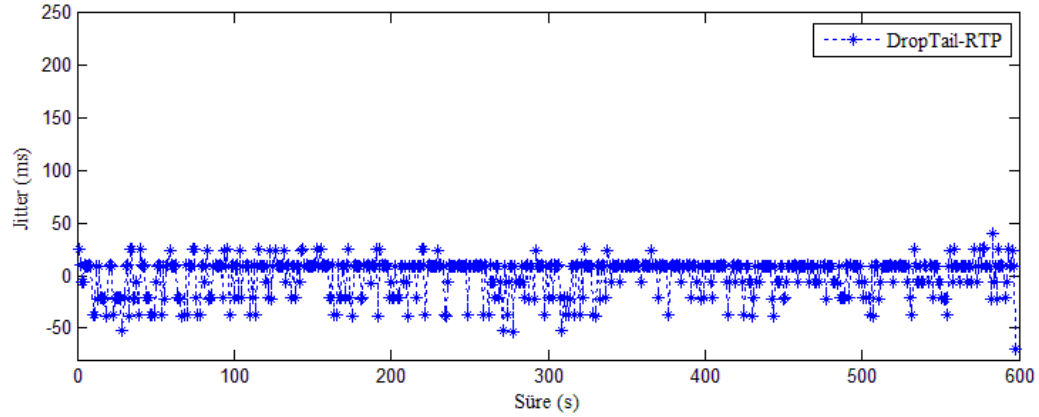
Şekil 3.69. JITT ile elde edilen gecikme

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük ve daha kararlı gecikme sağladığı görülmektedir.

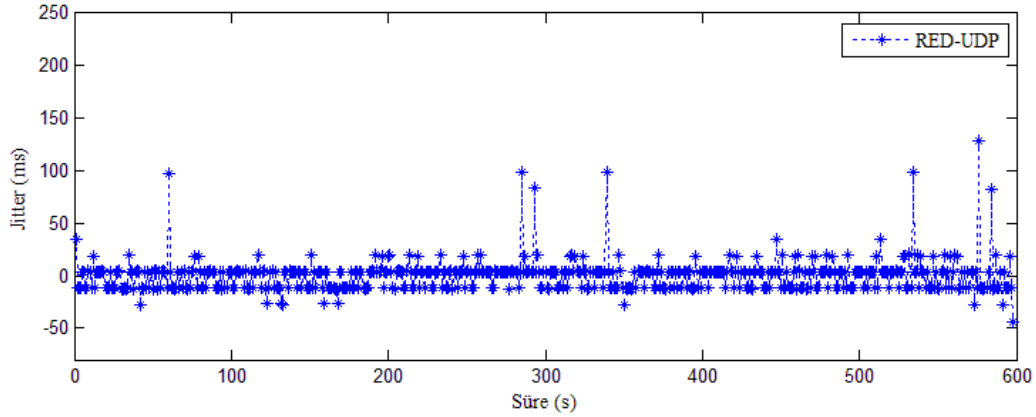
1 Mbps arka plan trafiği kullanılarak gerçekleştirilen deneme ile elde edilen jitter grafikleri Şekil 3.70 – Şekil 3.74'lerde gösterilmektedir.



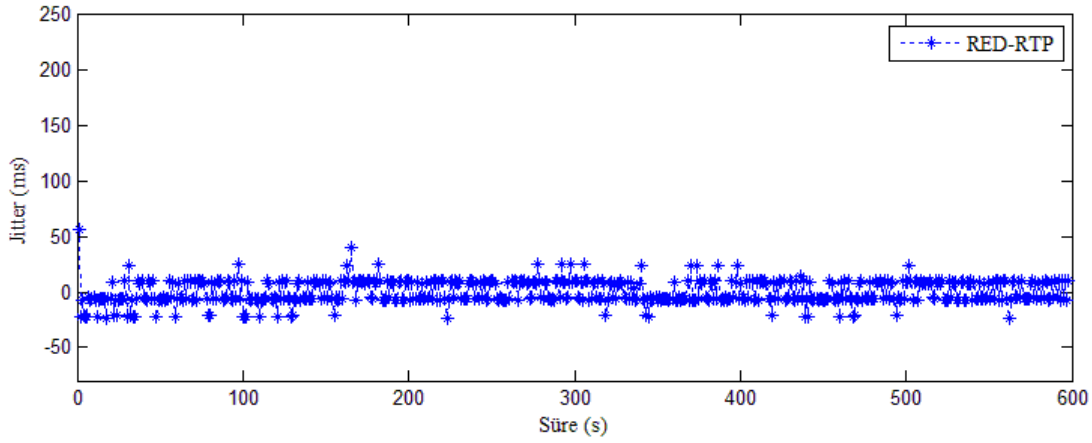
Şekil 3.70. DropTail-UDP ile elde edilen jitter



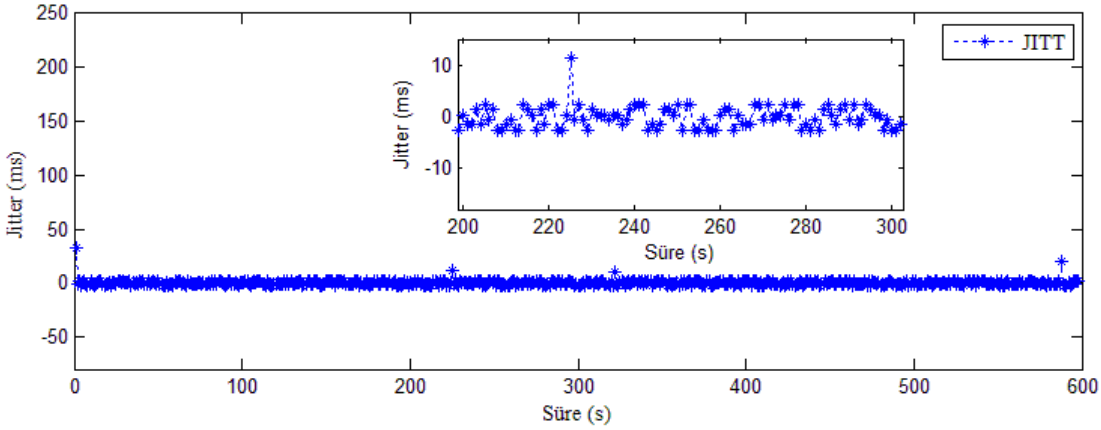
Şekil 3.71. DropTail-RTP ile elde edilen jitter



Şekil 3.72. RED-UDP ile elde edilen jitter



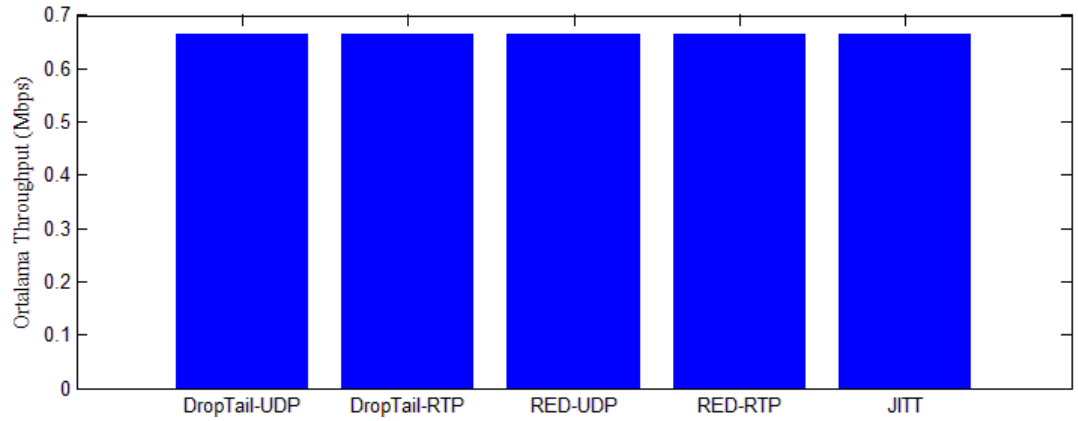
Şekil 3.73. RED-RTP ile elde edilen jitter



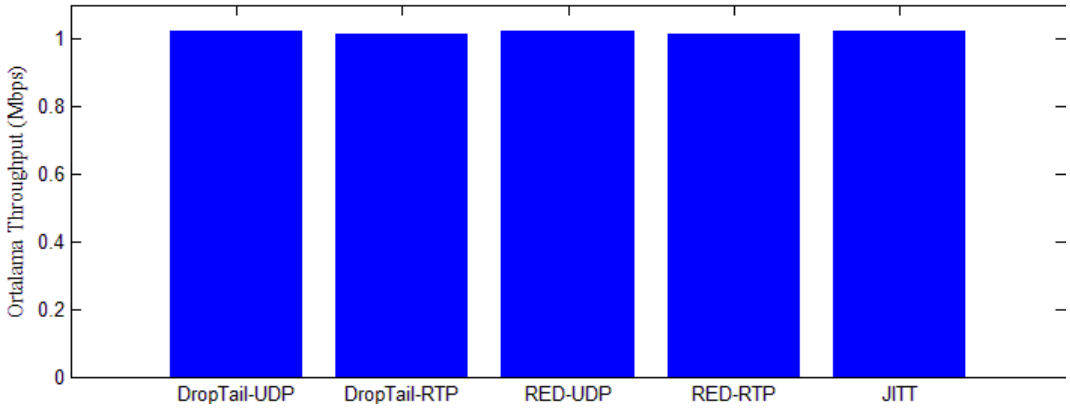
Şekil 3.74. JITT ile elde edilen jitter

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük jitter sağladığı görülmektedir.

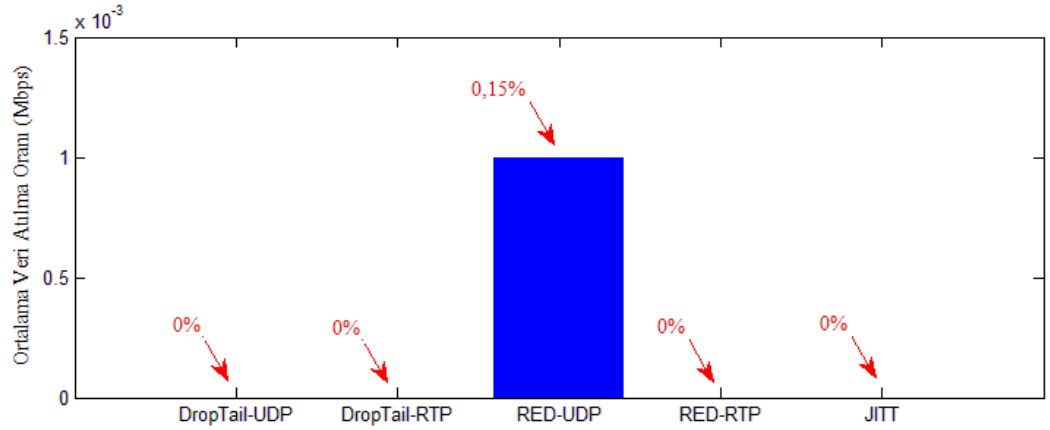
Gecikme ve jitter değerlerinin yanı sıra GZÇO trafiğinin ve arka plan trafiğinin throughput ve veri atılma değerleri incelenmiştir. Şekil 3.75'de GZÇO trafiğinin throughput değerleri; Şekil 3.76'da arka plan trafiğinin throughput değerleri, Şekil 3.77'de GZÇO trafiğinin veri atılma değerleri ve Şekil 3.78'de arka plan trafiğinin veri atılma değerleri gösterilmektedir.



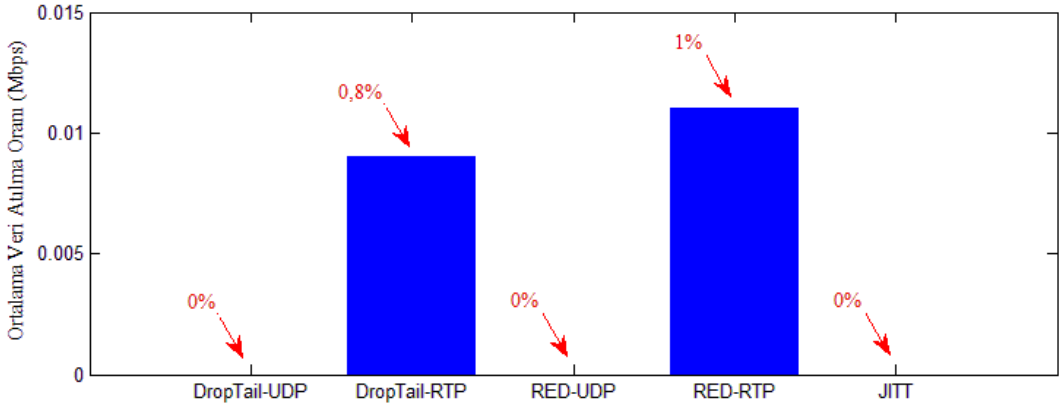
Şekil 3.75. GZÇO trafiği throughput değerleri



Şekil 3.76. Arka plan trafiği throughput değerleri

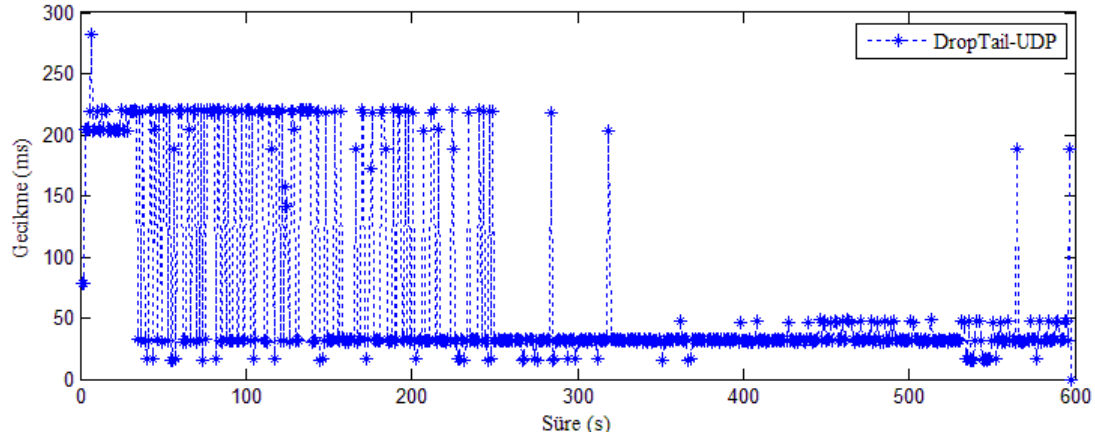


Şekil 3.77. GZÇO trafiği veri atılma değerleri

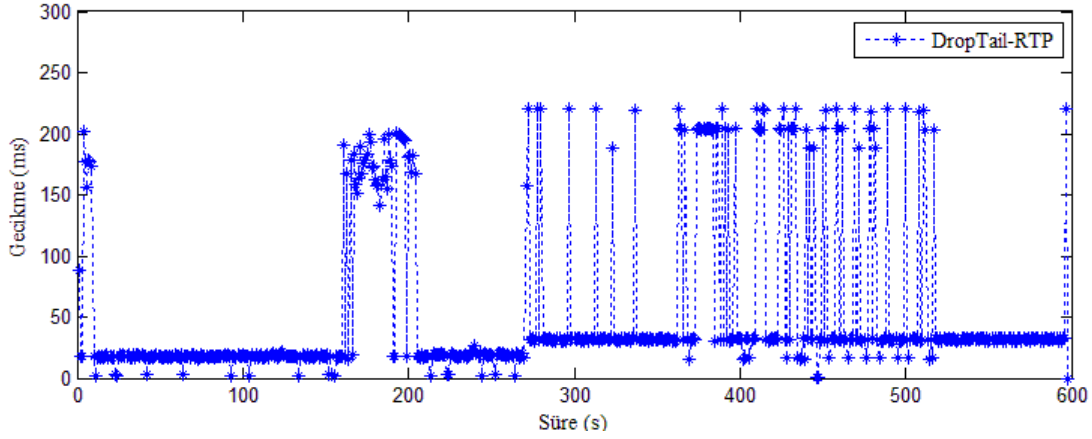


Şekil 3.78. Arka plan trafiği veri atılma değerleri

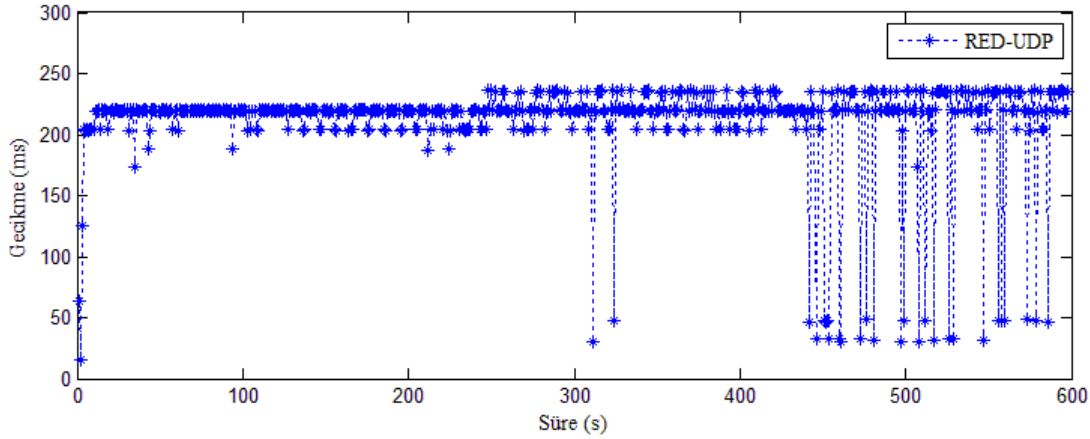
İkinci deneme olarak 2 Mbps arka plan trafiği kullanılarak gerçekleştirilen deneme ile elde edilen gecikme grafikleri Şekil 3.79 – Şekil 3.83'lerde gösterilmektedir.



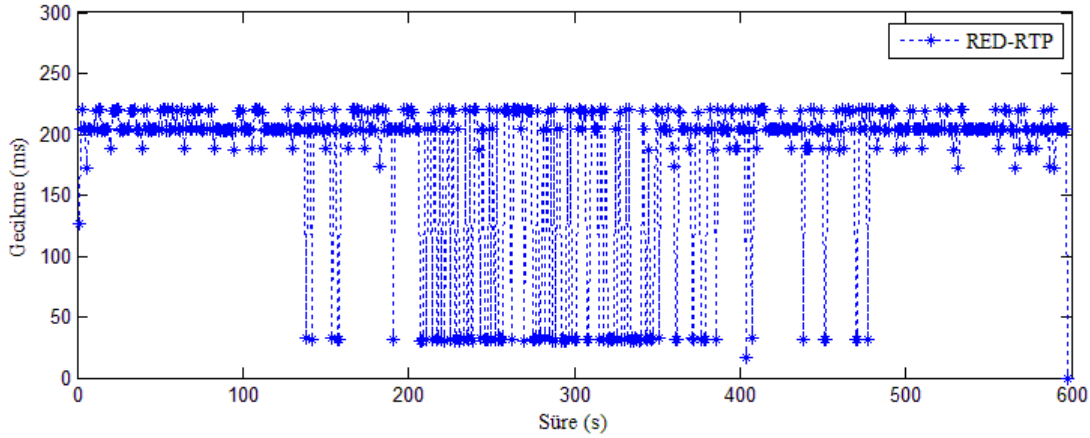
Şekil 3.79. DropTail-UDP ile elde edilen gecikme



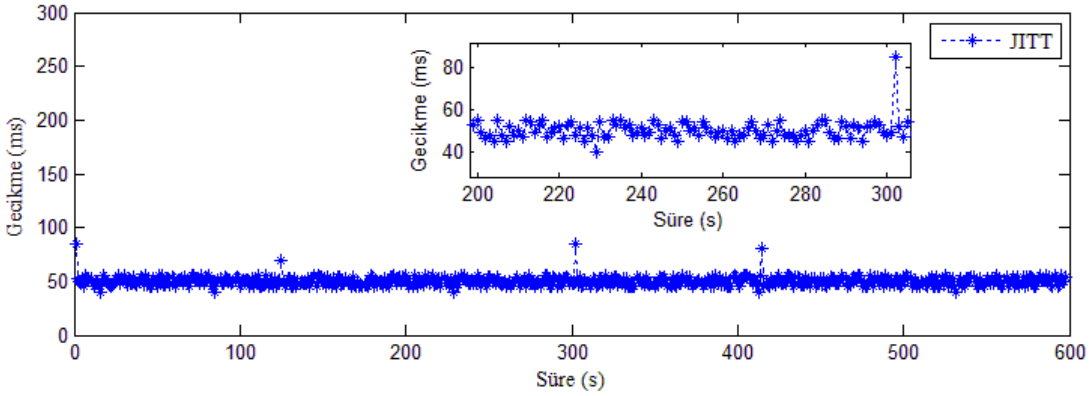
Şekil 3.80. DropTail-RTP ile elde edilen gecikme



Şekil 3.81. RED-UDP ile elde edilen gecikme



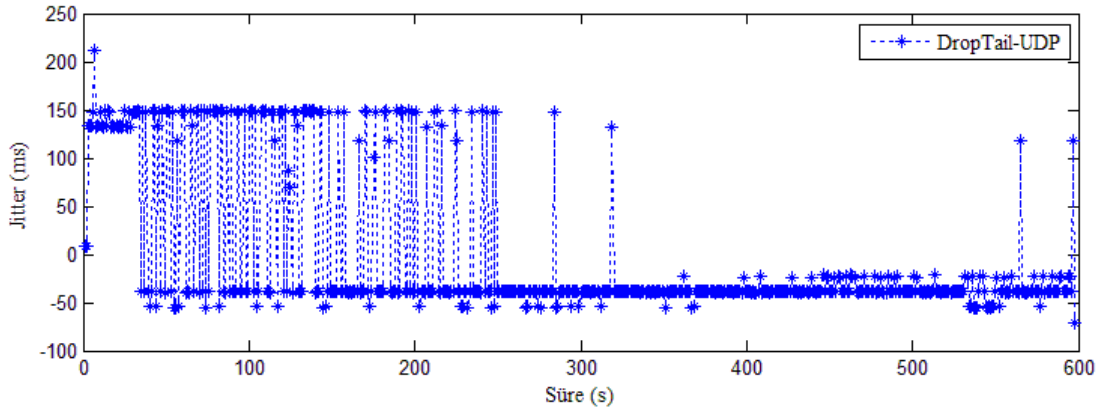
Şekil 3.82. RED-RTP ile elde edilen gecikme



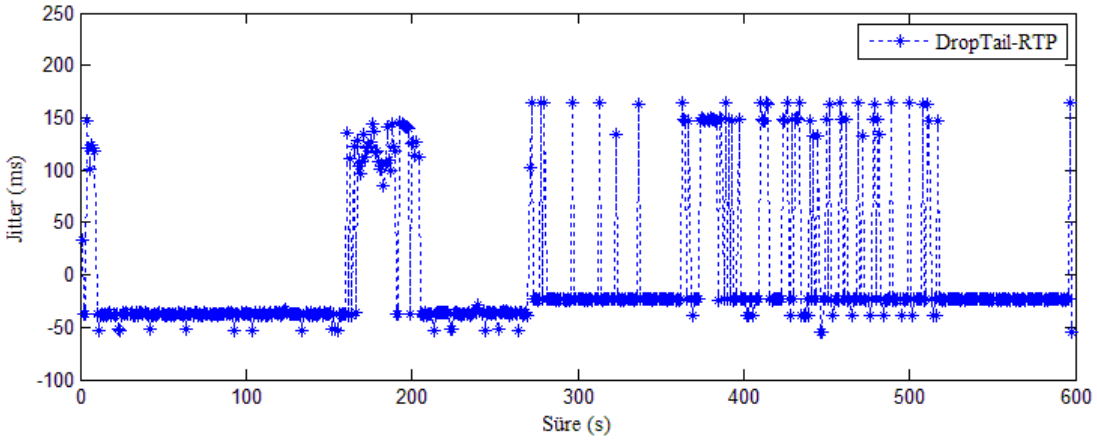
Şekil 3.83. JITT ile elde edilen gecikme

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük ve daha kararlı gecikme sağladığı görülmektedir.

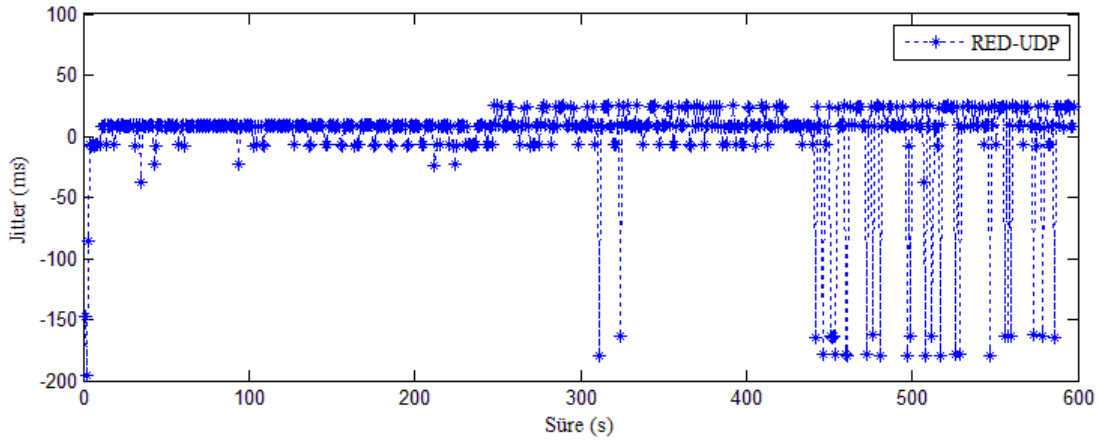
2 Mbps arka plan trafiği kullanılarak gerçekleştirilen deneme ile elde edilen jitter grafikleri Şekil 3.84 – Şekil 3.88'lerde gösterilmektedir.



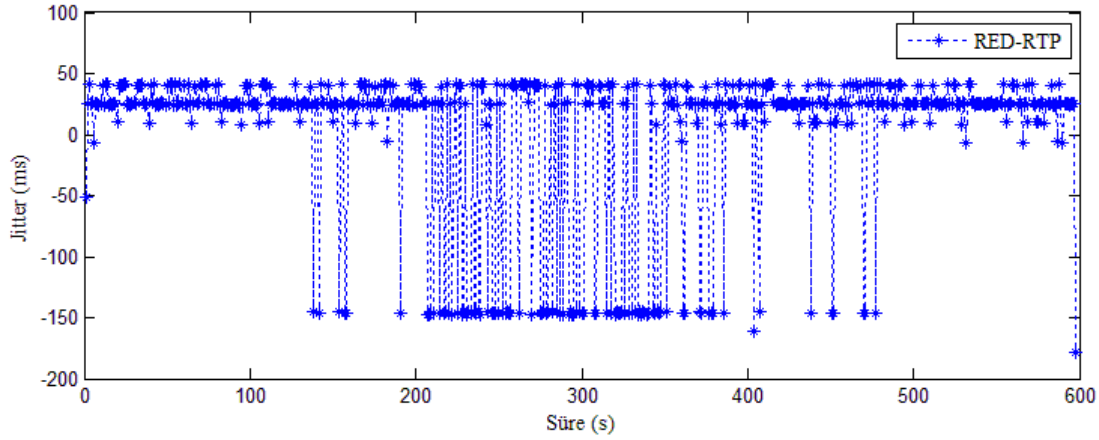
Şekil 3.84. DropTail-UDP ile elde edilen jitter



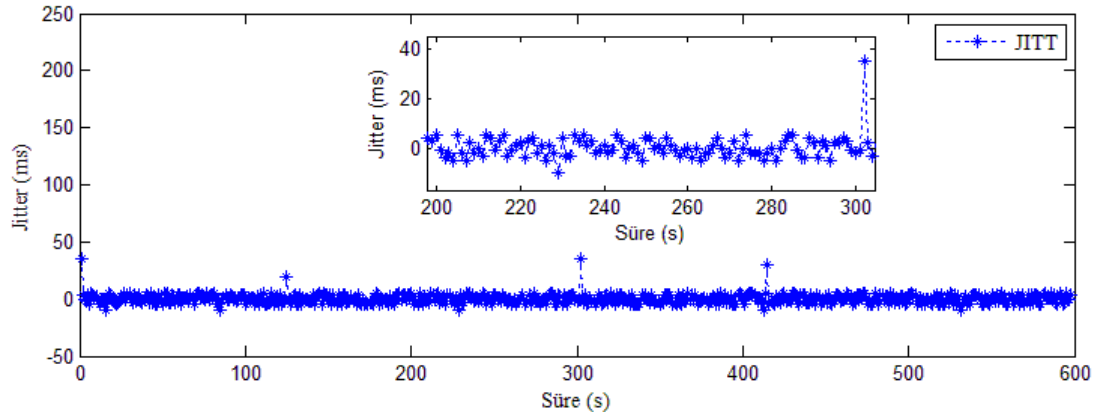
Şekil 3.85. DropTail-RTP ile elde edilen jitter



Şekil 3.86. RED-UDP ile elde edilen jitter



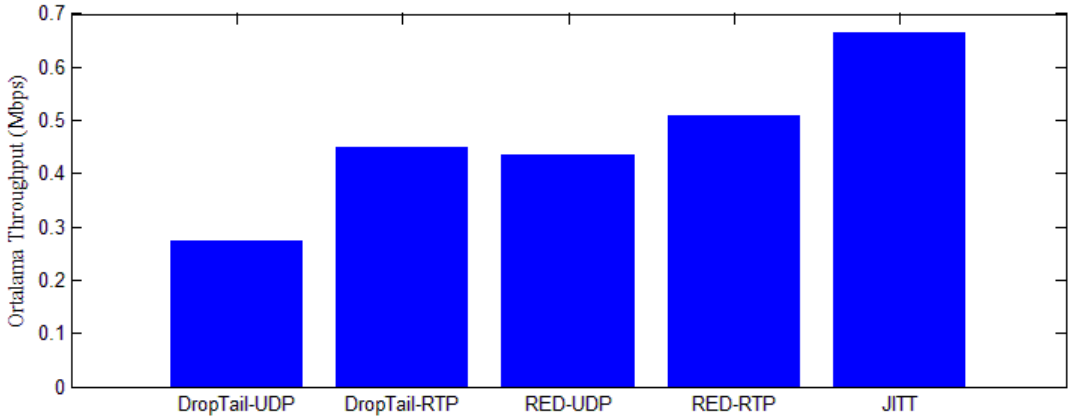
Şekil 3.87. RED-RTP ile elde edilen jitter



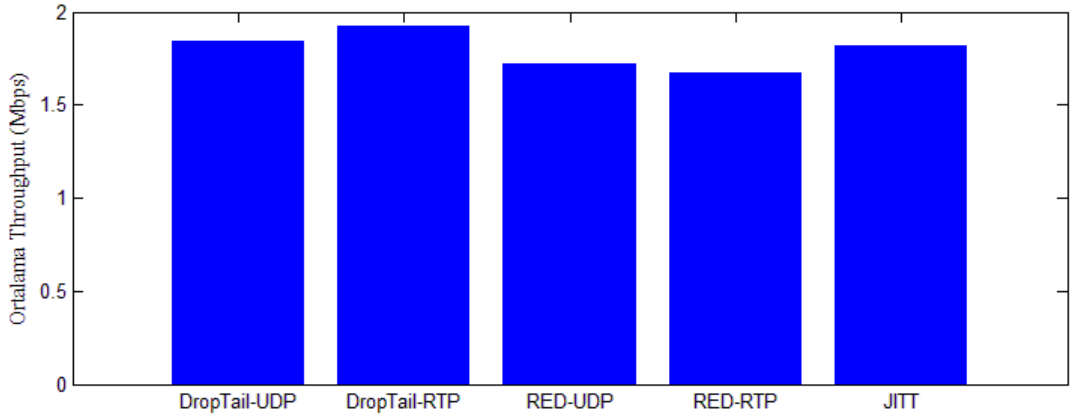
Şekil 3.88. JITT ile elde edilen jitter

Yukarıdaki grafiklerde JITT'in diğer protokol-kuyruk yapısı çiftlerine göre daha düşük jitter sağladığı görülmektedir.

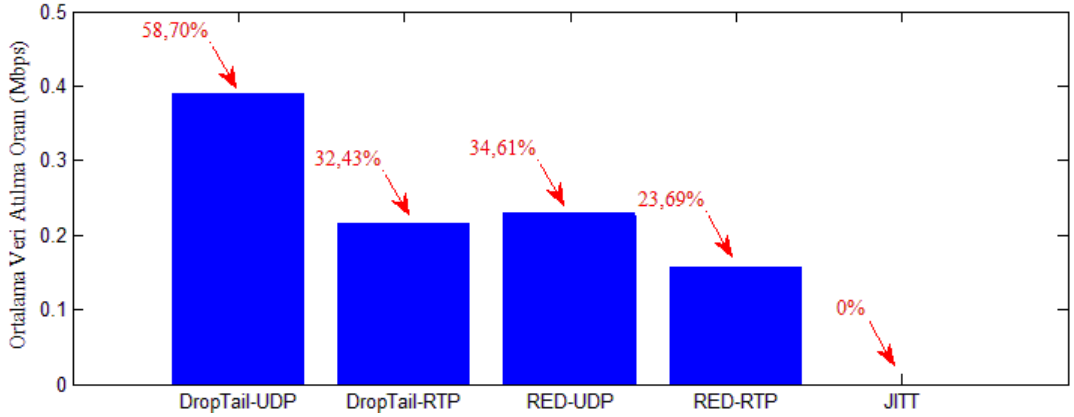
Gecikme ve jitter değerlerinin yanı sıra GZÇO trafiğinin ve arka plan trafiğinin throughput ve veri atılma değerleri incelenmiştir. Şekil 3.89'da GZÇO trafiğinin throughput değerleri; Şekil 3.90'da arka plan trafiğinin throughput değerleri, Şekil 3.91'de GZÇO trafiğinin veri atılma değerleri ve Şekil 3.92'de arka plan trafiğinin veri atılma değerleri gösterilmektedir.



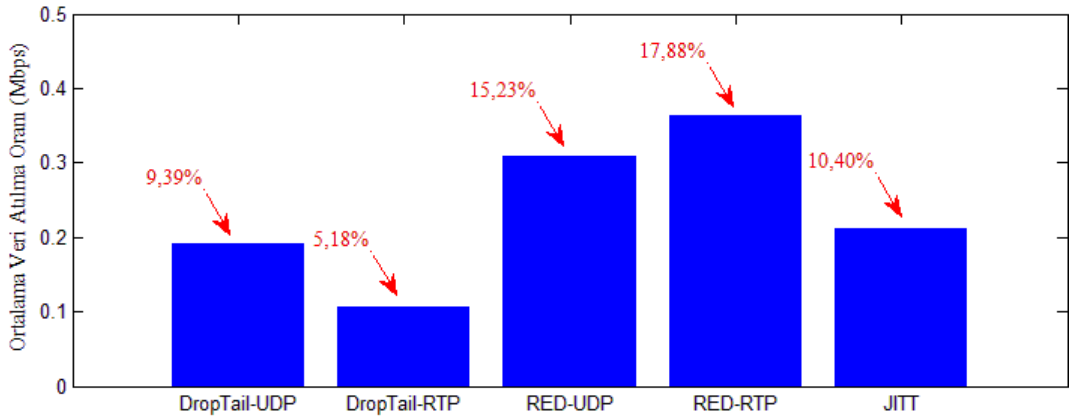
Şekil 3.89. GZÇO trafiği throughput değerleri



Şekil 3.90. Arka plan trafiği throughput değerleri



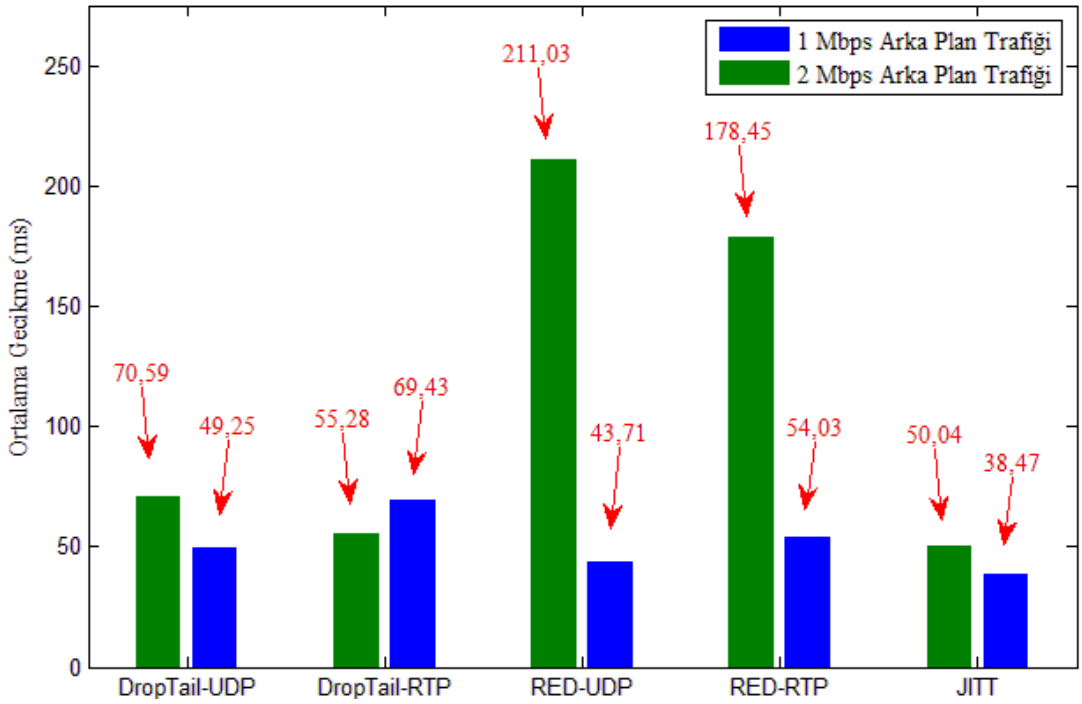
Şekil 3.91. GZÇO trafiği veri atılma değerleri



Şekil 3.92. Arka plan trafiği veri atılma değerleri

Ayrıca, 1 ve 2 Mbps arka plan trafikleri için gerçekleştirilen denemelerden elde edilen ortalama gecikme değerleri elde edilmiştir. Bu değerler Şekil 3.93'de gösterilmektedir. Şekil 3.93'den de görüldüğü gibi JITT, diğer kuyruk yapısı-ulaştırma protokolü çiftlerine göre her iki denemede de daha düşük gecikme sağlamıştır.

Son olarak, diğer kuyruk yapısı-ulaştırma protokolü çiftlerine göre daha iyi sonuçlar veren DropTail-RTP ve JITT'in detaylı olarak karşılaştırılması amacıyla bazı denemeler gerçekleştirilmiştir. Gerçekleştirilen denemelerde Şekil 3.64'de gösterilen topoloji kullanılmış fakat yönlendiriciler arası bant genişliği 2,5 Mbps yerine 10 Mbps olarak alınmıştır.



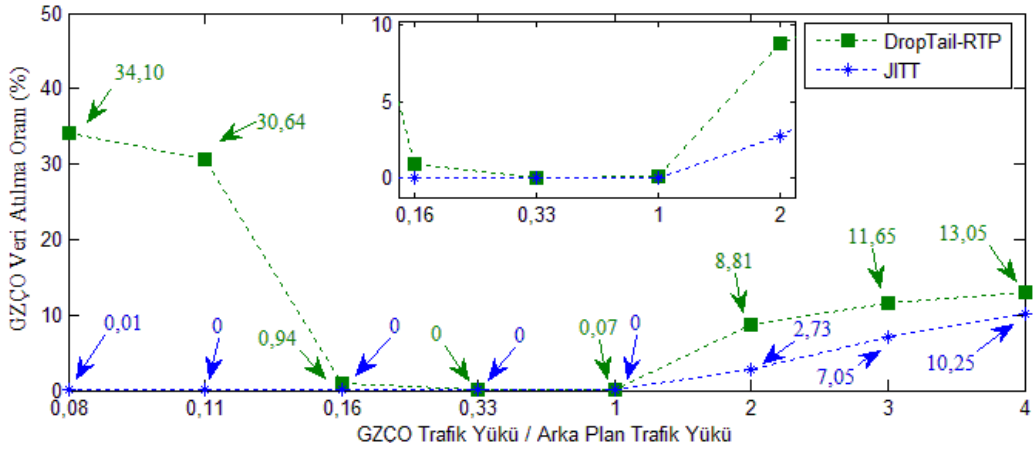
Şekil 3.93. Ortalama gecikme değerleri

Yukarıda anlatılan denemelerin haricinde farklı GZÇO trafik yükü / arka plan trafik yükü oranları için denemeler yapılmıştır. Her bir GZÇO trafik yükü / arka plan trafik yükü oranı için kullanılan trafik yükleri Çizelge 3.10'da gösterilmektedir. Uygulamada JITT'ten sonra en iyi sonucu veren kuyruk yapısı-protokol çifti DropTail-RTP olduğu için, gerçekleştirilen denemelerde DropTail-RTP ve JITT'in farklı yük oranları altındaki GZÇO veri atılma oranları, arka plan veri atılma oranları ve ortalama gecikmeler karşılaştırılmıştır. Şekil 3.94'de GZÇO veri atılma oranları, Şekil 3.95'de arka plan veri atılma oranları ve Şekil 3.96'da GZÇO ortalama gecikme değerleri gösterilmektedir. Şekil 3.94'de, JITT'in çoğu durumda GZÇO veri atılmasını önlediği; yalnızca GZÇO veri yükünün artması ile birlikte bağlantı zamanlayıcısının bant genişliğini eşit olarak paylaştırma davranışı nedeni ile GZÇO paketlerinden atılmalar yaşandığı görülmektedir. Buna karşılık DropTail-RTP; trafik yükünün yüksek olduğu durumlarda çok daha fazla GZÇO veri atılmasına neden olmuştur. Şekil 3.95'de de JITT'in, arka plan trafiğini, DropTail-RTP'ye göre daha iyi koruduğu görülmektedir. 0,08 oranından daha düşük değerlerde arka plan trafğinin fazlalığı nedeni ile tıkanıklık olacaktır. Önceki uygulama denemelerinde 2

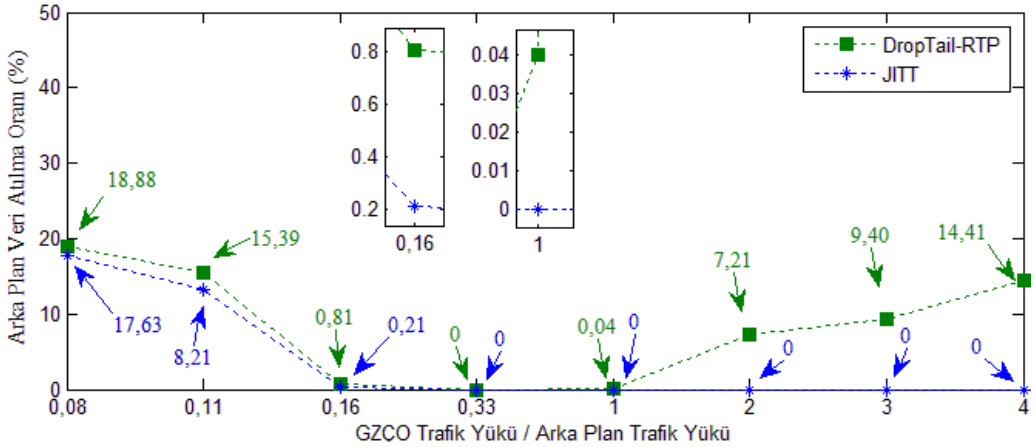
Mbps arka plan trafiği kullanılarak bu tıkanıklık durumu zaten test edilmiştir. Bu nedenle 0,08 oranından daha düşük oranlar için denemeler gerçekleştirilmemiştir.

Çizelge 3.10. GZÇO trafik yükü / arka plan trafik yükü oranları ve toplam trafik yükü

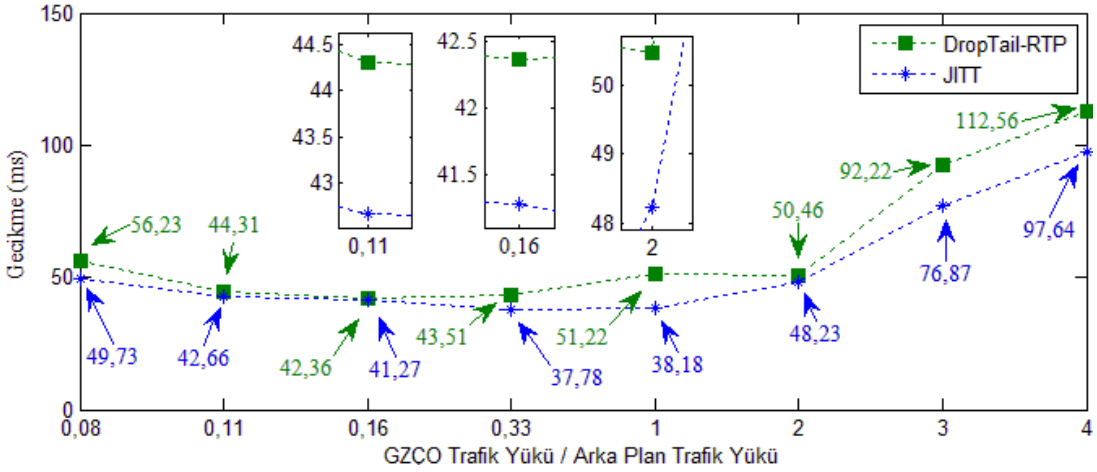
GZÇO trafik yükü / arka plan trafik yükü oranı	*GZÇO trafik yükü*	*Arka plan trafik yükü*	*Toplam yük*
0,08	665 Kbps	8 Mbps	8,665 Mbps
0,11	665 Kbps	6 Mbps	6,665 Mbps
0,16	665 Kbps	4 Mbps	4,665 Mbps
0,33	665 Kbps	2 Mbps	2,665 Mbps
1	2 Mbps	2 Mbps	4 Mbps
2	4 Mbps	2 Mbps	6 Mbps
3	6 Mbps	2 Mbps	8 Mbps
4	8 Mbps	2 Mbps	10 Mbps



Şekil 3.94. DropTail-RTP ve JITT için GZÇO veri atılma oranları



Şekil 3.95. DropTail-RTP ve JITT için arka plan veri atılma oranları



Şekil 3.96. DropTail-RTP ve JITT için ortalama gecikme değerleri

Son olarak Şekil 3.96'da GZÇO akışlarının ortalama gecikme değerleri gösterilmektedir. Bu değerlere bakıldığında, her ne kadar yakın değerlerde olsa da JITT'in daha düşük ortalama gecikme sağladığı görülmektedir. Denemelerin birçoğunda birden fazla GZÇO akışı olduğu için her bir akışın jitter grafiği ayrı ayrı çizdirilmemiştir. Jitter hesabı anlık gecikmelerin ortalama gecikmeden farkı alınarak yapıldığı için ve bu durumda ortalama jitter sıfıra çok yakın değerlerde olacağı için ortalama jitter grafiği çizdirilmemiştir. Ayrıca, benzetim denemelerinde ve önceki uygulama denemelerinde JITT'in diğer bütün kuyruk yapısı-ulaştırma protokolü çiftlerine göre çok daha düzgün jitter sağlamasından dolayı, jitter grafikleri çizdirilmemiştir.

Sonuç olarak, bütün parametreler (GZÇO ve arka plan paket atılma oranları, gecikme) göz önünde bulundurulduğunda JITT'in daha başarılı sonuçlar verdiği söylenebilir.

4. GELİŞTİRİLEN ULAŞTIRMA PROTOKOLÜ VE KUYRUK MODELİNİN MATEMATİKSEL ANALİZİ

Kuyruk yapılarının kuyruk teorisinde birçok gösterimi bulunmaktadır. Bir kuyruk sisteminin temel karakteristikleri [59]'da şu şekilde sıralanmaktadır:

1. Girişlerin varış deseni: Sisteme gelen işlerin varış tutumları. (Stokastik veya Deterministik)
2. Hizmet Deseni: Sistemde bulunan işlere verilen hizmetin tutumu. (Stokastik veya Deterministik)
3. Hizmet eden ünitelerin sayısı: Hizmet işlemini yerine getiren ünitelerin sayısı
4. Sistemin kapasitesi: Hizmet verilmek üzere alınabilecek maksimum iş sayısı. (Sonsuz veya sabit bir değer)
5. Kuyruk disiplini: Sistemde bekleyen işlerin, işlere hizmet vermek üzere nasıl alınacağı. (İlk giren – ilk çıkar, son giren – ilk çıkar veya rastgele)

Örneğin girişlerin varış deseni ve hizmet deseni stokastik olan; hizmet eden ünite sayısı bir olan ve sonsuz kapasiteli bir kuyruk sistemi M/M/1/∞ şeklinde gösterilmektedir. JITT kuyruğundan bahsedecek olursak girişlerin varış deseninin stokastik, hizmet deseninin deterministik, 1 hizmet ünitesi bulunan ve sınırlı kapasiteli bir model olduğunu söyleyebiliriz. Bu bağlamda JITT kuyruk modelinin gösterimi M/D/1/C şeklinde olacaktır.

4.1. Bekleme Zamanı Dağılımları

JITT'in gerek benzetim oramında gerekse gerçek ortamda diğer ulaştırma protokolü – kuyruk modeli çiftlerine göre gecikme, jitter ve ortalama gecikmeden maksimum sapma parametreleri açısından daha iyi sonuçlar verdiği ilgili bölümlerde anlatılmıştır. Bu bölümde JITT paketlerinin bekleme zamanı analizi gerçekleştirilerek, yukarıda bahsedilen parametreler için elde edilen benzetim ve gerçek ortam sonuçlarının arkasında yatan matematiksel temeller irdelenmiştir.

JITT kuyruğundaki paket bekleme sürelerinin dağılımını incelemeden önce genel olarak kuyruk yapılarındaki bekleme sürelerinin dağılımlarından bahsetmek faydalı olacaktır. Bir iş için bir kuyruk sisteminde 2 bekleme süresinden bahsedilebilir. Bunlar, kuyrukta bekleme süresi W_q ve hizmet süresi t'dir. Toplamda bir işin kuyrukta bekleme süresi $W = W_q + t$ olarak ifade edilir. Bilgisayar haberleşmesinde kuyruk sistemine dâhil olan her bir iş bir paket olduğu için bundan sonra iş yerine paket ifadesi kullanılacaktır. Bir paket için W_q değeri, paketin kuyruğa vardığı anda kuyrukta olan paket sayısına bağlı olarak aşağıdaki şekilde hesaplanabilir.

$$W_q = t_1' + t_2 + \cdots t_n \tag{4.1}$$

Burada t_1' kuyruğa en son dahil olan paketin hizmet süresi, t_n n. paketin hizmet süresi, n kuyruktaki toplam paket sayısıdır.

Haberleşme ağlarında paketlerin yönlendiricilere varış deseni Poisson dağılımı ile ifade edilir [59]. Buna göre bit cinsinden paket boyutu φ, hizmet oranı olan çıkış bant genişliği bps cinsinden B_{out} olarak gösterilirse, n. paket için hizmet süresi $t_n = \frac{\varphi_n}{B_{out}}$ olarak hesaplanabilir. Örnek olarak 1000 byte büyüklüğündeki bir paketin 1 Mbps bant genişliğine sahip bir hattan gönderilmesi için gereken süre $t = \frac{1000 \times 8}{1.000.000} =$ 8ms olarak bulunur. Bir GZÇO akışında üretilen paketlerin boyutu sabittir ya da birbirlerine yakın değerlerde değişmektedir. Yönlendiricinin çıkış bant genişliği de sabit olduğu için aynı akıştaki her bir paket için t süresi eşit olarak alınabilir. Bu durumda, eğer kuyruk uzunluğu zaman göre değişmiyorsa, yeni gelen her bir paket için hesaplanacak W_q süresi de sabit olacağından toplam bekleme süresi W da sabit olur. Ancak pratikte kuyruk uzunluğunu sabit tutmak mümkün olmadığından her bir paketin karşılaşacağı W süresi de farklı olacaktır. M/D/1/C kuyruk yapıları için analitik bir çözümleme Brun ve Garcia tarafından [60] çalışmasında yapılmıştır. Buna göre M/D/1/C kuyruk modelinde ortalama bekleme süresi W_C aşağıdaki şekilde hesaplanmaktadır.

$$W_C = \left(C - 1 - \frac{\sum_{k=0}^{C-1} b_k - C}{\rho b_{C-1}}\right) t \quad (4.2)$$

Burada $\rho = \frac{\lambda}{\mu}$, λ giriş varış oranı, μ hizmet oranı, $b_i = \sum_{k=0}^{i} \frac{(-1)^k}{k!} (i-k)^k \, e^{(i-k)\rho} \rho^k$ olarak alınmıştır. Eş. 4.2'den de görüldüğü üzere M/D/1/C kuyruk yapısına sahip bir sistemde ortalama bekleme süresi kuyruk kapasitesi ve anlık kuyruk uzunluklarına bağlıdır. Daha basit bir anlatım için P_i ve P_{i+1} aynı akışa ait olan iki paket olsun ve bu paketlerin yönlendirici kuyruğuna eklenmeleri ile birlikte kuyrukta bekleyen paket sayıları sırası ile $k \neq m$ olmak üzere k ve m olsun. Bu durumda P_i için $(W_q)_{P_i} = t'_1 + t_2 + \cdots t_k$ ve P_{i+1} için $(W_q)_{P_{i+1}} = t'_1 + t_2 + \cdots t_m$ olur. t süreleri bütün paketler için eşit olarak alındığından $(W_q)_{P_i} - (W_q)_{P_{i+1}} = |k-m| \times t \neq 0$ olarak elde edilir. Bu da bir akış için uçtan uca gecikmenin zamana göre salınım göstermesi ve jitter'in sıfırdan farklı çıkmasına neden olur.

Daha önceki başlıklarda bahsedildiği üzere bir JITT akışındaki paketlerin istenen maksimum uçtan uca gecikme süreleri JITT paket başlığındaki Delay alanı ile verilmektedir. Buna göre yol üzerindeki bir yönlendirici bir JITT paketinin kendi kuyruğunda ne kadar süre bekleyeceğini

$$\tau = \frac{\text{Delay}}{\text{hopcount}} \quad (4.3)$$

eşitliği ile hesaplamaktadır. Burada hopcount, yol üzerinde bulunan toplam yönlendirici sayısıdır. φ ilgili paketin bit cinsinden boyutu ve B_{out} çıkış bant genişliği olarak alınırsa, JITT kuyruk yapısının uygulandığı bir yönlendirici bir JITT paketinin gönderilme zamanının gelip gelmediğini Eş. 4.4 ile hesaplar.

$$\tau - t \geq \Delta \quad (4.4)$$

Burada Δ, paketin kuyruğa dâhil edilmesinden itibaren geçen süredir. Karar verme işleminin ardından ilgili paket derhal gönderilme işlemine tabi tutulmakta ve t kadar

bir süre içerisinde gönderilmektedir. Bir JITT akışını aşağıdaki biçimde gösterebiliriz.

$$F = \{P_1, P_2 \dots P_n\} \tag{4.5}$$

Burada n, akışta toplam paket sayısıdır. Aynı akışa ait bütün paketler için Delay ve hopcount değerleri aynıdır. Böylece τ değerleri de aynı olur. F akışındaki her bir paketin kuyruktan alınma zamanının gelip gelmediğini tespit etmek için kullanılan Eş. 4.4 ile F akışındaki her bir paket için bekleme süreleri $T_{bekleme} = \{(\tau - t_1), (\tau - t_2), \dots, (\tau - t_n)\}$ olarak hesaplanır. *F* akışındaki her bir paket için hizmet süreleri $T_{hizmet} = \{(t_1), (t_2), \dots, (t_n)\}$ olduğundan

$$T_{bekleme} + T_{hizmet} \cong \{(\tau), (\tau), \dots, (\tau)\} \tag{4.6}$$

bulunur. Böylece aynı akışa ait olan her bir JITT paketi, kuyrukta bekleyen paket sayısından bağımsız olarak τ bekleme süresine sahip olur. Bu durum bir JITT akışındaki bütün paketler için geçerli olduğundan JITT akışının gecikmesi sabit tutulmaktadır.

Önceki bölümlerde jitter parametresinin tanımı bir paketin gecikme süresinin aynı akıştaki ortalama gecikmeden farkı olarak verilmiştir.
F akışı için jitter:

$$J = \{\tau_1 - \tau_{ort}, \tau_2 - \tau_{ort}, \cdots, \tau_n - \tau_{ort}\} \tag{4.7}$$

Burada τ_{ort}, F akışındaki paketlerin ortalama gecikme süresidir. JITT akışındaki her bir paketin kuyrukta bekleme süresi Eş. 4.3 ile $\sim\tau$ olarak hesaplandığından her bir paketin bekleme süresi $\tau_1 \cong \tau_2 \cong \cdots \cong \tau_n$ olacaktır. Buradan bir F akışı için ortalama gecikme

$$\tau_{ort} \cong \frac{\sum_1^n \tau_n}{n} \tag{4.8}$$

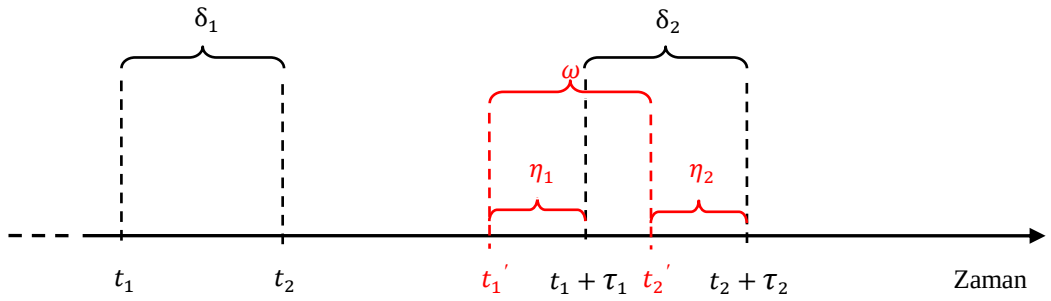
olarak hesaplanır. n adet paketin her birinin bekleme süreleri yaklaşık olarak eşit olduklarından $\tau_{ort} \cong \tau_i, 1 \leq i \leq n$ olarak seçilebilir. Böylece F akışı için jitter

$$J \cong \{\tau_1 - \tau_{ort}, \tau_2 - \tau_{ort}, \cdots, \tau_n - \tau_{ort}\} \cong \{0,0, \ldots, 0\} \quad (4.9)$$

olarak bulunur. JITT bir kuyruk modeli olduğundan fiziksel ortamdan kaynaklanan gecikme değişimleri ve veri bağı katmanından kaynaklanan hatalar ve yeniden gönderimlerin neden olabilecekleri düşük gecikme süreleri ile ilgilenmediği için yukarıdaki eşitliklerde "≅" işareti kullanılmıştır.

4.2. Gönderme Zamanı Analizi

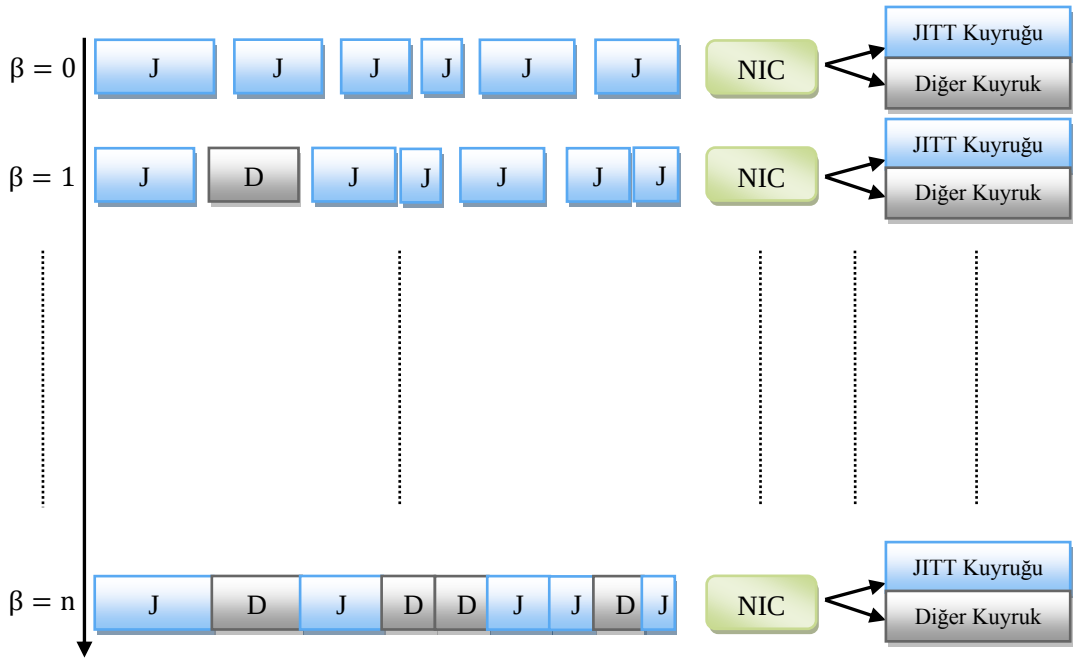
Bu başlıkta JITT kuyruk yapısında ard arda alınan paketlerin boyutlarının, paketlerin kuyrukta maksimum bekleme sürelerinin, giriş ve çıkış bant genişliklerinin ve diğer trafiklerin oranlarının, JITT paketlerinin kuyruktan alınma zamanlarını nasıl etkilediği incelenecektir. $P_1 \prec P_2$ olmak üzere P_1 ve P_2 JITT kuyruğundaki iki paket olsun ve P_1'in son bitinin alınmasının ardından hiçbir bekleme olmadan P_2'nin alınmaya başladığı kabul edilsin. P_1 paketinin kuyruğa eklenme zamanı t_1, istenen maksimum bekleme süresi τ_1, hizmet süresi olan $\frac{\varphi_1}{B_{out}}$ değeri η_1, kuyruktan alınma zamanı t_1' ve P_2 paketinin kuyruğa eklenme zamanı t_2, istenen maksimum bekleme süresi τ_2, hizmet süresi olan $\frac{\varphi_2}{B_{out}}$ değeri η_2, kuyruktan alınma zamanı t_2' olsun. P_1 ve P_2 paketlerinin alınma zamanları farkı $\delta_1 = t_2 - t_1 = \frac{\varphi_2}{B_{in}}$ ve gönderilme zamanları farkı $\delta_2 = (t_2 + \tau_2) - (t_1 + \tau_1)$ olarak hesaplanabilir. Bu durumda P_1 ve P_2 paketlerinin kuyruktan alınma zamanları farkı $\omega = t_2' - t_1'$ olacaktır. Yukarıda bahsedilen durumlar Şekil 4.1'de gösterilmektedir.



Şekil 4.1. JITT kuyruk modelinde paket zamanlaması

ω değerinin φ_1, φ_2, B_{in}, B_{out}, τ_1 ve τ_2 değerlerine bağlı olduğu açıktır. $\varphi_1 = \varphi_2$, $B_{in} = B_{out}$, $\tau_1 = \tau_2$olduğu durumda $\omega = \delta_1 = \delta_2$olur. En kötü durumda bazı koşulların sağlanması ile $t_2' = t_1'$ olabilir. Bu da P_2 paketinin gönderilme zamanının η_1 kadar ötelenerek $t_2 + \tau_2 + \eta_1$ olması demektir. Bu durumun bir JITT akışının gecikmesini sürekli olarak etkileyebilmesi, farklı JITT akışlarına ait paketlerin aralarında hiçbir gecikme olmadan yönlendiriciye varması ile mümkündür. Bu ön koşulun sağlanmasından sonra $B_{out} < B_{in}$ ve $\varphi_2 > \varphi_1$koşullarının da aynı anda sağlanması ile $\eta_2 = \delta_1 + \eta_1$ durumunda $t_2' = t_1'$ olabilir. Bunlardan başka $\tau_2 \neq \tau_1$ ön koşulu ile $\tau_2 = (t_1 + \tau_1 + \eta_2) - \eta_1$ özel durumunda da $t_2' = t_1'$ olur. $t_2' = t_1'$ durumunun gerçekleşme ihtimalinin analizi için Şekil 4.2'deki yönlendirici giriş paket yoğunluğunun incelenmesi gerekmektedir. Bir yönlendiriciye giren JITT akışları birim zamanda $\propto$ kadar paket üretsin. b_j, anlık JITT trafik oranını göstermek üzere $1 \leq \alpha \leq m$ için $\lim_{\propto \to m} f(b_j) = B_j$ olur. Burada m, yönlendiricinin ağ arayüz kartına (NIC) birim zamanda gelebilecek maksimum JITT paketi sayısı, B_j, yönlendiricinin ağ arayüz kartına (NIC) birim zamanda gelebilecek maksimum JITT trafik oranıdır. Bir yönlendiriciye giren Diğer trafik akışları birim zamanda β kadar paket üretsin. b_d, anlık diğer trafik oranını göstermek üzere $1 \leq \beta \leq n$ için $\lim_{\beta \to n} f(b_d) = B_d$ olur. Burada n, yönlendiricinin ağ arayüz kartına (NIC) birim zamanda gelebilecek maksimum diğer trafik paketi sayısı, B_d, yönlendiricinin ağ arayüz kartına (NIC) birim zamanda gelebilecek maksimum diğer trafik oranıdır. Toplam olarak $B_{in} = B_j + B_d$ elde edilir. Şekil 4.2'de $\alpha = m$ alınarak β değerinin değişmesi ile yönlendiricinin NIC aygıtına ard arda varan paketlerdeki JITT ve diğer trafik paket oranlarının değişimi gösterilmiştir. β'nın küçük değerlerinde

yönlendiricinin NIC aygıtına ard arda varan paketlerin büyük çoğunluğu JITT paketidir. Ancak bu durumda her bir JITT paketinin arasında oransal olarak $\frac{B_{in}-B_j}{B_{in}}$ kadar süre boşluk kalacaktır. Bu durum da $t_2' = t_1'$ durumunun gerçekleşme ihtimalini düşürmektedir. β'nın büyük değerlerinde ise yönlendiricinin NIC aygıtına ard arda varan paketlerin türü değişkenlik gösterecektir. Bu durumda da JITT paketlerinin aralarında belirli süreler boşluk bırakacağından $t_2' = t_1'$ durumunun gerçekleşme ihtimali düşmektedir.



Şekil 4.2. Yönlendirici girişi paket yoğunluğu

Yönlendiricinin NIC aygıtına ard arda varan paketlerin tamamının JITT paketleri olması ve paketler arasında hiç süre kalmaması durumu $B_j = B_{in}$koşulunun sağlanması ile mümkündür ki bu durum da internet ortamında mümkün gözükmemektedir. Bütün koşullar gerçekleşip $t_2' = t_1'$ durumu oluşsa bile $\eta_1 = \frac{\varphi_1}{B_{out}}$ olduğundan t_2' değerindeki artışın yönlendiricinin çıkış bant genişliği ile ters orantılı olduğu görülmektedir. GZÇO akışlarında paket boyutları çok büyük olmamakla birlikte ortalama olarak $\varphi_1 = 8.000$ bit ve $B_{out} = 10$ Mbps olan bir durumda bir

yönlendirici üzerinde P_2 paketinin t_2' süresine etki edecek η_1 süresi $\eta_1 = \frac{8.000}{10.000.000} =$ 0,8 ms olacaktır ki bu süre bir akış için ihmal edilebilir bir süredir.

5. SONUÇ

Bu tezde, JITT olarak isimlendirilen ve paket anahtarlamalı ağlarda hizmet kalitesini arttırmaya yönelik olarak bir ulaştırma protokolü ve kuyruk yapısı geliştirilmiş ve geliştirilen protokol ve kuyruk yapısına JITT ismi verilmiştir. JITT protokolü ve kuyruk yapısı ns2 benzetim aracı üzerinde kodlanmış, gerçek ortamda uygulanmış ve UDP, RTP gibi ulaştırma katmanı protokolleri ile DropTail ve RED gibi kuyruk yapılarıyla karşılaştırılmıştır.

Gerçekleştirilen benzetimlerde JITT ile gecikme 50-75 ms bandında tutulmuştur. Diğer kuyruk yapısı-ulaştırma protokolü çiftleri ile yapılan denemelerde gecikmelerin 50-300 ms bantlarında değiştiği görülmüştür. Gecikmeye bağlı olarak jitter istenilen seviyelerde tutulmuştur. JITT ile ortalama gecikmeden maksimum sapma %4,5 seviyelerinde ölçülürken, diğer kuyruk yapısı-ulaştırma protokolü çiftleri ile yapılan denemelerde %20 seviyelerinde ölçülmüştür. JITT, GZÇO veri atılmasını önlemiştir. Buna karşılık diğer kuyruk yapısı-ulaştırma protokolü çiftleri %3'lük bir GZÇO veri atılmasına neden olmuşlardır. JITT ile arka plan trafiği veri atılması %1,5 seviyelerinde ölçülmüştür. Diğer kuyruk yapısı-ulaştırma protokolü çiftleri ile yapılan denemelerde ise %7 seviyelerinde ölçülmüştür. Ölçeklenebilirlik senaryosunda artan GZÇO bağlantı sayısına rağmen JITT ile arka plan trafiği korunmuştur. Sonuç olarak JITT ile gecikme, jitter ve ortalama gecikmeden maksimum sapma parametrelerinde 4 kat kadar bir iyileşme elde edilmiştir.

Gerçekleştirilen uygulamalarda JITT ile 1 Mbps arka plan trafiği için gecikme 40-50 ms bandında; 2 Mbps arka plan trafiği için 40-60 ms bandında tutulmuştur. Diğer kuyruk yapısı-ulaştırma protokolü çiftleri ile yapılan denemelerde gecikmelerin 100-200 ms bantlarında değiştiği görülmüştür. Gecikmeye bağlı olarak jitter istenilen seviyelerde tutulmuştur. JITT ile ortalama gecikmeden maksimum sapma %17 seviyelerinde ölçülürken, diğer kuyruk yapısı-ulaştırma protokolü çiftleri ile yapılan denemelerde %111seviyelerinde ölçülmüştür. JITT, GZÇO veri atılmasını önlemiştir. Buna karşılık diğer kuyruk yapısı-ulaştırma protokolü çiftleri %18'lik bir GZÇO veri atılmasına neden olmuşlardır. JITT ile arka plan trafiği veri atılması

%2,5 seviyelerinde ölçülmüştür. Diğer kuyruk yapısı-ulaştırma protokolü çiftleri ile yapılan denemelerde ise %7,5 seviyelerinde ölçülmüştür. 2 Mbps arka plan trafiği ile gerçekleştirilen denemede yönlendiriciler arasındaki bağlantının bant genişliği 2.5 Mbps olduğu için arka plan trafiğinde paket atılmaları yaşanmış; ancak toplamda yaklaşık 2.5 Mbps gibi bir throughput elde edilebilmiştir. Diğer kuyruk yapısı-ulaştırma protokolü çiftleri ile yapılan denemelerde gerek arka plan trafiğinde gerekse GZÇO trafiğinde büyük paket atılma oranları gözlemlenmiş ve böylece düşük throughput değerleri elde edilmiştir. Değişik GZÇO/Arka Plan Trafiği oranlarında, önceki uygulama sonuçları ile uyumlu sonuçlar elde edilmiştir. Gecikme ve jitter ölçütlerinde JITT açık bir biçimde diğer protokol-kuyruk yapısı çiftlerine göre üstünlüğünü ortaya koymuştur. Sonuç olarak JITT ile gecikme ve jitter parametrelerinde 3,5; ortalama gecikmeden maksimum sapma parametresinde ise 6 kat kadar bir iyileşme elde edilmiştir.

Yukarıda bahsedildiği gibi JITT ile elde edilen düşük ve kararlı gecikmenin sebebi, paketlerin kuyrukta bekleme sürelerinin kuyrukta bekleyen diğer paketlerin sayısından ve paket boyutlarından bağımsız olarak belirlenebilmesidir. Düşük ve kararlı gecikme değerlerinin elde edilmesinin yanı sıra hattın toplam bant genişliğinin etkin ve adil kullanılmasının sebebi ise, JITT kuyruk yapısının JITT harici trafikler için de zamanlama yapması ve aynı zamanda bağlantı zamanlayıcısı ile uyumlu çalışabilmesidir. Gerçekleştirilen benzetim ve uygulamalar, JITT'in analitik modeli ile uyumlu sonuçlar vermiştir.

Gerçekleştirilen benzetim ve uygulamalarda JITT'in tutarlı sonuçlar verdiği gözlemlenmiştir. Diğer protokol-kuyruk yapısı çiftleri ile gerçekleştirilen benzetimlerden elde edilen gecikme ve jitter değerleri ile aynı protokol-kuyruk yapısı çiftleri ile gerçekleştirilen uygulamalardan elde edilen gecikme ve jitter değerleri arasında 1,5 kat (benzetim sonuçlarında daha yüksek) kadar bir fark bulunmaktadır. İlk bakışta bu bir tutarsızlık gibi görünse de bu farkın nedeni GZÇO paket atılma oranları göz önünde bulundurulduğunda anlaşılmaktadır. Diğer protokol-kuyruk yapısı çiftleri ile gerçekleştirilen uygulamalarda GZÇO paket atılma oranlarının, aynı protokol-kuyruk yapısı çiftleri ile gerçekleştirilen benzetimlerdeki GZÇO paket

atılma oranlarından daha yüksek olduğu görülmektedir. Yani, diğer protokol-kuyruk yapısı çiftlerinin uygulama ortamında daha düşük gecikme sağlamalarının nedeni daha yüksek GZÇO paket atılma oranı sağlamış olmalarıdır.

Mevcut GZÇO uygulamalarında alıcı tarafın, gönderici taraftan gelen paketleri tamponlaması ve oynatma işlemini bu tampondan yürütmesi jitter'i absorbe etmek için kullanılan bir yöntemdir. Tamponlama işlemi aynı zamanda, kaybolan paketlerin yeniden gönderilebilmesi için gönderici tarafa süre sağlamaktadır. Bununla birlikte, tamponlama işlemi başlangıç oynatma süresini arttırmaktadır [17]. Tampon boyutu ile oynatmaya başlama süresi arasında sıkı bir ilişki vardır. Tamponlanan paketlerin miktarı belirli bir seviyeye gelince alıcı taraf verileri oynatmaya başlar. Eğer kullanılan tampon büyük seçilirse gecikmelerden kaynaklanan etkiler yumuşatılmış olur. Ancak, uçtan uca gecikmeler küçük tutulmak isteniyorsa tampon kullanımına dikkat edilmelidir [17]. Dolayısı ile oynatma tamponunun büyük olması iletişimin kalitesini gecikme açısından düşürmektedir. Mevcut uygulamalarda büyük tampon boyutu kullanımının nedeni gecikmenin kararlı olmamasıdır. JITT, gecikmeyi kararlı tutarken jitter'i de düşük tutarak daha küçük tampon boyutları kullanılmasını sağlayabilir.

Son olarak, kuyruk yapılarının analizleri kararlı hal (steady state) durumunda gerçekleştirilmektedir. Kuyruk yapılarının kararlı hal durumu, tıkanıklık olmadığı durumlar olarak açıklanabilir [59,60]. Bu açıdan JITT'in gerçek performans değerlendirmesi, tıkanıklık olmayan durumlardaki davranışına göre belirlenmelidir. Tıkanıklık olmadığı durumlarda, gerek düşük trafik yükü gerekse yüksek trafik yükü altına JITT, kararlı gecikme ve jitter sağlarken, paket atılma oranlarını da düşük tutmuştur. Her ne kadar JITT, tıkanıklık durumlarında da başarılı sonuçlar vermiş olsa da, özünde JITT bir tıkanıklık denetimi veya tıkanıklık giderme mekanizması değildir. Tıkanıklık durumundan kaçınma veya tıkanıklık durumunu kontrol altına almak için birtakım başka mekanizmaların JITT ile birlikte kullanılması mümkündür.

KAYNAKLAR

1. İnternet: International Telecommunication Union "ITU-T Recommendation E.800: Definitions of Terms Related to Quality of Service" http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-E.800-200809-I!!PDF-E&type=items (2008).

2. Masip-Bruin, X., Yannuzzi, M., Domingo-Pascual, J., Fonte, A., Curado, M., Monteiro, E., Kuipers, F., Van Mieghem, P., Avallone, S., Ventre, G., Aranda Guiterrez, P., Hollick, M., Steinmetz, R., Iannone, L., Salamatian, K., "Research challenges in QoS routing", ***Computer Communications***, 29(5): 563-581 (2006).

3. Cisco Systems, "Understanding Jitter in Packet Voice Networks (Cisco IOS Platforms)", ***Cisco Systems 18902***, ***San Francisco***, 1-7 (2006).

4. Demichelis, C., Chimento, P., "IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)", ***IETF RFC 3393, Kaliforniya***, 1-21 (2002).

5. Almes, G., Kalidindi, S., Zekauskas, M., "A One-way Delay Metric for IPPM", ***IETF RFC 2679, Kaliforniya***, 1-20 (1999).

6. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W., "An Architecture for Differentiated Services", ***IETF RFC 2475, Kaliforniya***, 1-36 (1998).

7. Nichols, K., Jacobson, V., Zhang, L., "Two-Bit Differentiated Services Architecture for the Internet", ***IETF RFC 2638, Kaliforniya***, 1-26 (1999).

8. Braden, R., Clark, D., "Integrated Services in the Internet Architecture: An Overview", ***IETF RFC 1633, Kaliforniya***, 1-33 (1994).

9. Stewart, R., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., Paxson, V., "Stream Control Transmission Protocol (SCTP)", ***IETF RFC 2960, Kaliforniya***, 1-134 (2000).

10. Ong, L., Yoakum, J., "An Introduction to the Stream Control Transmission Protocol (SCTP)", ***IETF RFC 3286, Kaliforniya***, 1-10 (2002).

11. Argyriou, A., "A novel end-to-end architecture for H.264 video streaming over the Internet", ***Telecommunication Systems***, 28(2): 133-150 (2005).

12. Schulzrinne, H., Casner, S., Frederick, R., Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", ***IETF RFC 1889, Kaliforniya***, 1-75 (1996).

13. Schulzrinne, H., Casner, S. ve Frederick, R., Jacobson, V., "RTP: A Transport Protocol for Real-Time Applications", ***IETF RFC 3550, Kaliforniya***, 1-104 (2003).

14. Shafqaat, A., Gohar, N. D., Atif, K., "A Dynamic Congestion Control Mechanism for Real-Time Streams over RTP", ***IEEE ICACT***, Kore Cumhuriyeti, 961-966 (2007).

15. Huang, C. M., Lin, C. W., Chuang, C. Y., "A Multilayered Audiovisual Streaming System Using the Network Bandwidth Adaptation and the Two-Phase Synchronization", ***IEEE Transactions On Multimedia***, 11(5): 797-809 (2009).

16. Kalman, M., Steinbach, E., Girod, B., "Adaptive Playout For Real-Time Media Streaming", ***IEEE ISCAS***, Arizona, 145-148 (2002).

17. Schaar, M. V. D., Chou, P. A., "Multimedia over IP and Wireless Networks: Compression, Networking, and Systems", ***Academic Press***, San Diego, 527-556 (2007).

18. Huang, C. M., Chang, C. K., "An Adaptive Flow Control With The Re-Transmission Policy Over The Server–Proxy–Client Networking Environment", ***The Journal of Systems and Software***, 69(1-2): 15-27 (2004).

19. Khlifi, H., Grégoire, J. C., "ARTP: A *Buffer-Aware Rate Control Protocol For Media Streaming*", ***Computer Networks***, 51(6): 1601-1615 (2007).

20. Floyd, S., Jacobson, V., "Random Early Detection Gateways For Congestion Avoidance", ***IEEE/ACM Transactions on Networking***, 1(4): 397-413 (1993).

21. Feng, W. C., Shin, K. G., Kandlur, D. D., Saha, D., "The BLUE Active Queue Management Algorithms", ***IEEE/ACM Transactions on Networking,*** 10(4): 513-528 (2002).

22. Feng, W. C., Kapadia, A., Thulasidasan S., "GREEN: Proactive Queue Management Over a Best-Effort Network", ***GLOBECOM***, Taipei, 1774-1778 (2002).

23. Athuraliya, S., Low, S. H., Li, V. H., Qinghe, Y., "REM: Active Queue Management", ***IEEE Network***, 15(3): 48-53 (2001).

24. Floyd, S., Jacobson, V., "Link-Sharing and Resource Management Models for Packet Networks", ***IEEE/ACM Transactions on Networking***, 1(4): 365-386 (1995).

25. Parris, M., Jeffay, K., Smith, D. F., "Lightweight Active Router-Queue Management For Multimedia Networking", ***Multimedia Computing and Networking (MMCN)***, San Jose, 162–174 (1999).

26. Pan, R., Prabhaker, B., Psounis, K., "CHOKe: a Stateless Active Queue Management Scheme For Approximating Fair Bandwidth Allocation", ***IEEE INFOCOM***, Tel-Aviv, 942–951 (2000).

27. Yamaguchi, T., Takahashi, Y., "A Queue Management Algorithm For Fair Bandwidth Allocation", ***Computer Communications,*** 30(9): 2048-2059 (2007).

28. Papadimitriou, P., Tsaoussidis, V., "SSVP: A Congestion Control Scheme for Real-Time Video Streaming", ***Computer Networks***, 51(15): 4377-4395 (2007).

29. Demers, A., Keshav, S., Shenkar, S., "Analysis and Simulation of A Fair Queuing Algorithm", ***ACM SIGCOMM Computer Communication Review***, 19(4): 1-12 (1989).

30. Parekh, A. K., Gallager, R. G., "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case", ***IEEE Transactions on Networking***, 1(3): 344-357 (1993).

31. McKenney, P., "Stochastic Fairness Queuing", ***IEEE INFOCOM***, Florida, 733-740 (1991).

32. Shreedhar, M., George, V., "Efficient Fair Queuing Using Deficit Round-Robin", ***IEEE Transaction on Networking***, 4(3): 375-385 (1996).

33. Long, F., Feng, G., Tang, J. H., "A New Joint Packet Scheduling/Admission Control Framework for Multi-service Wireless Networks", ***Journal of Communications and Networks***, 7(4): 408-416 (2005).

34. Wu, E. H. K., Lai, H. T., Tsai, M. F., Chou C. F., "Low Latency And Efficient Packet Scheduling For Streaming Applications", ***Computer Communications***, 29(9): 1413-1421 (2006).

35. Pyun, K., Song, J., Lee, H. K., "The Service Curve Service Discipline For The Rate-Controlled EDF Service Discipline in Variable-Sized Packet Networks", ***Computer Communications***, 29(18): 3886-3899 (2006).

36. Lim, T. M., Lee, B. S., Yeo, C. K., "Edge-to-Edge Quality-of-Service Domain", ***IEICE Transactions on Communications***, 89(5): 1554-1569 (2006).

37. Yi, D. H., Kim, J. W., "A Frame-based Packet Scheduling Scheme for Real-time Applications: 2-tier Hierarchical Frame Queuing", ***IEEE CCNC***, Las Vegas, 238-242 (2008).

38. Xie, Y., Tu, X., Liu, H., Li, J., Li, T., Xie, J., "Insertion Based Packets Scheduling for Providing QoS Guarantee in Switch Systems", ***IEEE ICCCAS***, Fujian, 453-456 (2008).

39. Tsao, S. C., Lai, Y. C., Tsa, L. C., Lin, Y. D., "On Applying Fair Queuing Discipline to Schedule Requests At Access Gateway For Downlink Differential QoS", ***Computer Networks***, 52(18): 3392-3404 (2008).

40. Deering, S. ve Hinden, R., "Internet Protocol, Version 6 (IPv6)", ***IETF RFC 2460, Kaliforniya***, 1-39 (1998).

41. İnternet: NSNAM "NS-3: Network Simulator 3" http://www.nsnam.org/tutorials/NS-3-LABMEETING-1.pdf (2010).

42. Issariyakul, T., Hossain, E., "Introduction to Network Simulator NS2", ***Springer***, New York, 99-137 (2008).

43. Fall, K., Varadhan, K., "The ns Manual", ***ISI***, 45-62 (2010).

44. Domoxoudis, S., Kouremenos, S., Loumos, V., Drigas, A., "Measurement, Modeling and Simulation of Videoconference Traffic from VBR Video Encoders", ***HET-NETS***, Ilkley, 28-36 (2004).

45. Sreenan, C. J., Chen, J. C., Agrawal, P., Narendran, B., "Delay Reduction Techniques for Playout Buffering", ***IEEE Transactions on Multimedia***, 2(2): 88-100 (2000).

46. İnternet: The International Computer Science Institute "Measurement Studies of End-to-End Congestion Control in the Internet" http://www.icir.org/floyd/ccmeasure.html (2010).

47. Drigas, A. Kouremenos, S., Bakopoulos, Y., Loumos, V., "A Study of H.263 Traffic Modeling in Multipoint Videoconference Sessions Over IP Networks", ***Computer Communications***, 29(3): 372-391 (2006).

48. Wiegand, T., Sullivan, G. J., Bjontegaard, G., Luthra, A., "Overview of the H. 264/AVC Video Coding Standard", ***IEEE Transactions on Circuits and Systems for Video Technology***, 13(7): 560-576 (2003).

49. Gide, A., "Implementing Mobile TV", ***Focal Press***, Boston, 477-495 (2010).

50. Nguyen, D. T., Ostermann, J., "Congestion Control for Scalable Video Streaming Using the Scalability Extension of H.264/AVC", ***IEEE Journal of Selected Topics in Signal Processing***, 1(2): 246-253 (2007).

51. Lu, W. Z., Gu, W. H., Yu, S. Z., "One-Way Queuing Delay Measurement and Its Application on Detecting DDoS Attack", ***Journal of Network and Computer Applications***, 32(2): 367-376 (2009).

52. Choi, J. H., Yoo, C., "One-Way Delay Estimation and Its Application", ***Computer Communications***, 28(7): 819-828 (2005).

53. Ogawa, K., Sawai, A., Iida, N., Bandai, M., Watanabe, T., "Measurement Method of One-Way Differential Delay for End-to-End Path Selection on Multihoming", ***Electronics and Communications in Japan***, 90(6): 29-42 (2007).

54. Luong, D. D., Birio J., "Hop-by-Hop versus End-to-End Active Measurements", ***Computer Communications***, 25(10): 954-963 (2002).

55. Matera, F., Matteotti, F., Pasquali, P., Rea, L., Tarantino, A., Baroncini, V., Del Prete, G., Gaudino, G., "Comparison Between Objective and Subjective Measurements of Quality of Service Over an Optical Wide Area Network", ***Transactions on Emerging Telecommunications Technologies***, 19(3): 233-245 (2008).

56. Aoki, M., Oki, E., Rojas-Cessa, R., "Measurement Scheme for One-Way Delay Variation with Detection and Removal of Clock Skew", ***ETRI Journal***, 32(6): 854-862 (2010).

57. Gurewitz, O., Cidon, I., Sidi, M., "One-Way Delay Estimation Using Network-Wide Measurements", ***IEEE Transactions on Information Theory***, 52(6): 2710-2724 (2006).

58. Cisco Systems, "Measuring Delay, Jitter, and Packet Loss with Cisco IOS SAA and RTTMON", ***Cisco Systems 24121***, ***San Francisco***, 1-8 (2005).

59. Medhi, J., "Stochastic Models in Queueing Theory", ***Academic Press***, San Diego, 47-164 (2002).

60. Brun, O., Garcia, J. M., "Analytical Solution of Finite Capacity M/D/1 Queues", ***Journal of Applied Probability***, 37(4): 1092-1098 (2000).

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: ŞİMŞEK, Mehmet
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 22.09.1981 Artvin
Medeni hali	: Evli
Telefon	: 0 (312) 582 31 41
Faks	: 0 (312) 230 65 03
e-mail	: msimsek.ceng@gmail.com

Eğitim

Derece	**Eğitim Birimi**	**Mezuniyet tarihi**
Yüksek lisans	Gazi Üniversitesi / Bilgisayar Mühendisliği	2006
Lisans	Selçuk Üniversitesi / Bilgisayar Mühendisliği	2004
Lise	Ankara Başkent Lisesi	1998

İş Deneyimi

Yıl	**Yer**	**Görev**
2005-	Gazi Üniversitesi	Uzman

Yabancı Dil

İngilizce

Yayınlar

1. Şimşek, M., Akcayol, M.A., "Kablosuz Ağlarda Sezgisel Bir Yönlendirme Protokolü ve Tıkanıklık Denetimi", ***Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi***, 23(2): 57-63 (2008).

2. Şimşek, M., Yodaş, M., Bulut, A., Doğru, İ. A., Akcayol, M. A., "3G Tabanlı Uzaktan Kontrol Edilebilen Araç Geliştirilmesi", ***Gazi Üniversitesi Mühendislik-Mimarlık Fakültesi Dergisi***, 27(1): 135-142 (2012).

3. Şimşek, M., Akcayol, M.A., "Bilgisayar Ağlarında Tıkanıklık Denetimi ve Çözüm Yöntemleri", ***Bilişim Teknolojileri Dergisi***, 1(3): 51-56 (2008).

4. Doğru, İ.A., Şimşek, M., Akcayol, M.A., "Hareketli Ad-Hoc Ağlarda Bir Hareketlilik Yönetimi Protokolü", ***G.Ü. Politeknik Dergisi***, 11(4): 313-318 (2008).

5. Dogru, I.A., Simsek, M., Akcayol, M. A., "A Comprehensive Analysis of E-Learning Studies in Turkey, Problems and Suggestions", ***IADIS International Conference e-Learning***, Rome, Italy (2011).

6. Simsek, M., Dogru, İ. A., Akcayol, M. A., "Design Of An Expert System Supported Portable Cardiotocograph (CTG)", ***IADIS e-Health***, Rome, Italy (2011).

7. Simsek, M., Toklu, S., Akcayol, M.A., Soncul, H., Ergun, M.A., "A Comprehensive Analysis of E-Government Studies in Turkey, Problems and Suggestions", ***The IADIS WWW/Internet 2009 Conference***, Rome, Italy, 365-369 (2009).

8. Dogru, I. A., Simsek, M., Toklu, S., Yildiz, O., Akcayol, M.A., "Rule-Based Mobility Management Routing for Mobile Ad Hoc Networks", ***International Conference on Wireless Networks (ICWN)***, Las Vegas, Nevada, USA, 175-179 (2008).

9. Toklu, S., Simsek, M., Yildiz, O., Bay, I., Simsek, A., Akcayol, M.A., "A Priority Based Protocol and Load Balancing for Queue Management on Wireless Networks", ***International Conference on Wireless Networks (ICWN)***, Las Vegas, Nevada, USA, 341-344 (2008).

10. Akcayol, M.A., Şimşek, M., Bay, İ., Toklu, S., Doğru, I.A., "Genetic Algorithm Based Location Optimization of Emergency Service Units", ***4th FAE International Symposium***, Lefke (2006).

11. Akcayol, M.A., Şimşek, M., Bay, İ., "Türkiye'de E-Kütüphane Çalışmalarının Durum Analizi ve Öneriler", ***Akademik Bilişim***, Gaziantep (2005).