KABLOSUZ AĞLARDA YÖNLENDİRME PROTOKOLÜ VE TIKANIKLIK DENETİMİ

Mehmet ŞİMŞEK

YÜKSEK LİSANS TEZİ
BİLGİSAYAR MÜHENDİSLİĞİ

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

ARALIK 2006
ANKARA

Mehmet ŞİMŞEK tarafından hazırlanan KABLOSUZ AĞLARDA YÖNLENDİRME PROTOKOLÜ VE TIKANIKLIK DENETİMİ adlı bu tezin Yüksek Lisans tezi olarak uygun olduğunu onaylarım.

Doç. Dr. M. Ali AKCAYOL
Tez Yöneticisi

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

Başkan: : Prof. Dr. Sezai DİNÇER

Üye : Prof. Dr. Hadi GÖKÇEN

Üye : Doç. Dr. M. Ali AKCAYOL

Tarih : 25/12/2006

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

TEZ BİLDİRİMİ

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

Mehmet ŞİMŞEK

KABLOSUZ AĞLARDA YÖNLENDİRME PROTOKOLÜ VE TIKANIKLIK DENETİMİ

(Yüksek Lisans Tezi)

Mehmet ŞİMŞEK

GAZİ ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
Aralık 2006

ÖZET

Bu tezde, bilgisayar ağlarındaki en önemli sorunlardan biri olan tıkanıklık ele alınmış ve hareketli kablosuz ağlarda tıkanıklık denetimi için bir protokol geliştirilmiştir. Önerilen protokol talep üzerine çalışmakta ve tıkanıklık olacağını sezdiği anda devreye girmektedir. Benzetim aracı olarak Network Simulator 2 (ns-2) kullanılmıştır. Geliştirilen protokol, C++ programlama dili kullanılarak kodlanmış ve kablosuz ağlarda tıkanıklık denetimi için yaygın olarak kullanılan AODV (Ad Hoc On Demand Distance Vector) protokolü ile benzetim sonuçları karşılaştırılmıştır. Benzetimde değişik yükler altında ağın çalışması incelenmiş ve elde edilen sonuçlar değerlendirilmiştir. Deneysel sonuçlar, geliştirilen protokolün tıkanıklık önlemede daha başarılı olduğunu göstermiştir.

Bilim Kodu : **902.1.063**
Anahtar Kelimeler : **Tıkanıklık, Yönlendirme, Protokoller, Kablosuz Ağlar**
Sayfa Adedi : **69**
Tez Yöneticisi : **Doç. Dr. M. Ali AKCAYOL**

ROUTING PROTOCOL AND CONGESTION CONTROL IN WIRELESS NETWORKS

(M.Sc. Thesis)

Mehmet ŞİMŞEK

GAZİ UNIVERSITY

INSTITUTE OF SCIENCE AND TECHNOLOGY

December 2006

ABSTRACT

In this thesis, a congestion prevention protocol has been developed for mobile wireless networks. Congestion is the most important problem in computer networks. The proposed protocol has worked with on-demand method and activated when it senses that a congestion will occur. Network Simulator 2.29 (ns-2) has been used as simulation tool. The developed protocol has been coded with C++ programming language and compared with simulation results of AODV (Ad Hoc On Demand Distance Vector) routing protocol. The operation of the mobile wireless network has been investigated under different overheads and obtained results have been evaluated. Experimental results show that the developed protocol is more successful for congestion prevention.

Science Code : **902.1.063**
Key Words : **Congestion, Routing, Protocols, Wireless Networks**
Page Number : **69**
Adviser : **Assoc.Prof.Dr. M. Ali AKCAYOL**

TEŞEKKÜR

Çalışmalarım boyunca değerli yardım ve katkılarıyla beni yönlendiren Hocam Doç.Dr. M. Ali AKCAYOL'a, Yüksek Lisans Öğrenimim boyunca yardımlarını esirgemeyen değerli hocalarım Doç.Dr. Şeref SAĞIROĞLU'na ve Yrd.Doç.Dr. Hasan Şakir BİLGE'ye, her zaman yanımda olan aileme ve kıymetli varlığım Aybike ERKAYA'ya teşekkürü bir borç bilirim.

İÇİNDEKİLER

Sayfa

Sayfa

ÇİZELGELERİN LİSTESİ

Çizelge **Sayfa**

ŞEKİLLERİN LİSTESİ

Şekil **Sayfa**

Şekil **Sayfa**

SİMGELER VE KISALTMALAR

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

Simgeler	**Açıklama**
α	Alçak geçiren filtre kazancı
β	Gidiş-dönüş zamanı değişimi
g	Dolu kuyruk/toplam kuyruk uzunlukları
L(i)	Bir ağdaki anlık yük miktarı
mbps	Milyon bit/saniye
π	Pi Sayısı
sn	Saniye
W(i)	Pencere büyüklüğü
Wmax	Maksimum pencere uzunluğu

Kısaltmalar	**Açıklama**
Ack	Geri bildirim Paketi
AS	Autonom System - Otonom Sistem
BGP	Border Gateway Protocol - Kenar Ağ geçidi Protokolü
BSD	Berkeley Software Distribution - Berkeley Yazılım Dağıtımı
Cwnd	Tıkanıklık Penceresi
DV	Distance Vector - Uzaklık Vektörü

Kısaltmalar	Açıklama
ELFN	Explicit Link Failure Notification - Açık Düğüm Hata Bildirimi
ERDN	Explicit Route Disconnection Notification - Yol Bağlantı Kopukluğu Bildirimi
ERSN	Explicit Route Successful Notification - Açık Yol Başarılı Bildirimi
FP	Failure Point - Hata Noktası
ICMP	Internet Control Message Protocol - İnternet Kontrol Mesaj Protokolü
IGRP	Interior Gateway Routing ProtocolİÇ Ağ geçidi Yönlendirme Protokolü
IP	Internet Protocol - İnternet Protokolü
LQ	Yerelleştirilmiş Sorgu
LS	Bağlantı Durumu
LSA	Bağlantı Durumu İlanı
NNT	Komşunun Komşuları Listesi
OSPF	Open Shortest Path First - Açık En Kısa Yol Önce
PL	Tercih Edilen Komşuların Listesi
PN	Bağlantı Kopukluğunu Yakalayan Ara Düğüm
RIP	Routing Information Protocol - Yönlendirme Bilgisi Protokolü
RFN	Yol Hatası Bildirimi
RN	Yol Bildirimi
RRN	Yeniden Yol Kurulma Bildirimi
RTT	Gidiş-Dönüş Zamanı

Kısaltmalar	Açıklama
RTO	Yeniden İletim Zaman Aşımı Aralığı
SRTT	Ortalama Gidiş-Dönüş Zamanı
TCP	Transmission Control Protocol - Aktarım Kontrol Protokolü
TCP – BUS	TCP with Buffering Capability and Sequence Information - Tamponlama Yeteneğine Sahip ve Sıra Numarası Bilgisi Kullanan TCP
TCP-ELFN	TCP with Explicit Link Failure Notification Açık Düğüm Hata Bildirimli TCP
TCP-F	Feedback Based TCP - Geri Besleme Tabanlı TCP
ToS	Type of Service - Servis Tipi
ZRP	Zone Routing Protocol - Alan Yönlendirme Protokolü

1. GİRİŞ

Bilgisayar ağlarında yönlendirme ve tıkanıklık denetimi ağın performansını etkileyen en önemli unsurlardır. Bir ağda fazla yükten dolayı paketlerin bekleme süreleri artarsa, paket kayıpları yaşanırsa ve ağın etkinliği azalırsa bu ağ tıkanmış demektir. Ağın etkinliğinin azalması, kaybolan paketlerin yeniden gönderilmesini gerektirmekte ve ağa fazladan yük getirmektedir [1-3].

Tıkanıklık kişiye göre değişen bir durumdur. Örneğin paketlerin bir ağda kaybolma ihtimali 1/500 olsun. Gönderdiği paketlerin 1/1000 oranında bir hata ile iletilmesini isteyen bir kullanıcı bu ağa tıkanık derken, gönderdiği paketlerin 1/100 oranında bir hata ile iletilmesinden memnun olan bir kullanıcı için bu ağ tıkanık değildir. Yine aynı şekilde, ağı elektronik posta göndermek için kullanan biri gönderdiği e-postanın ertesi gün iletilmesinden memnun olabilirken, gerçek zamanlı video aktarımı yapan başka bir kullanıcı aynı ağın tıkalı olduğunu söyleyebilmektedir. Bazı kaynaklarda tıkanıklık, ağ yükünün artması ile kullanıcının istediği performansın düşmesi şeklinde tanımlanmaktadır [3]. Diğer bir tıkanıklık tanımı ise şu şekildedir: Ağdaki aşırı yüklenme sonucunda kuyrukta bekleyen paketlerin sayısının anormal şekilde artması, yeni gönderilen paketlerin işleme alınamaması ve işleme alınamayan paketlerin yeniden gönderilmesi sonucunda kuyruk uzunluğunun sürekli olarak artması şeklindedir [3, 4].

Temel olarak iki çeşit tıkanıklık önleme yöntemi vardır. Bunlar, rezervasyon tabanlı ve isteğe bağlı metotlarıdır [4]. Bir kullanıcının ihtiyacı olan hizmet kalitesi için önceden ağ üzerinde kaynak ayrılır. Kullanıcı istediği zaman bu kaynakları kullanarak ağ üzerinden bilgi transfer eder. Bu yöntem rezervasyon tabanlı olarak adlandırılabilir. Alternatif olarak kullanıcı gerek duyduğu anda ağdaki kaynakları rezerve etmek isteyebilir. Fakat ağdaki mevcut yük dolayısı ile istediği kalitede hizmet alamayabilir. İkinci metot rezervasyonsuz ağlarda kullanılır. Bu durumda kullanıcılar kendilerini değişen ağ durumuna adapte etmek durumundadırlar.

İsteğe bağlı planda ise ağı kullanan kişiler sürekli olarak ağı gözlemlemeli ve tıkanıklığı önlemek için ağın durumunu değiştirmelidirler. Her iki yöntemin de avantajları ve dezavantajları vardır. Birinci yöntemde hizmet kalitesi garanti edilir fakat kullanıcı sayısı sınırlandırılmak zorundadır. İkinci yöntemde kullanıcı sayısı sınırlı değildir fakat her kullanıcı aldığı hizmetten ödün vermek durumunda kalabilir.

Literatürde ulaştırma katmanında ve yönlendirme katmanında gerçekleştirilmiş birçok tıkanıklık denetleme ve giderme yöntemi bulunmaktadır [1, 2].

Rezervasyon tabanlı ve isteğe bağlı yöntemlerde tam bir dışta bırakma söz konusu değildir. İki yöntemin bir arada uygulandığı karma sistemler de tasarlanabilir. Bazı kullanıcılar için kesin bir hizmet kalitesi sağlanması gerekiyor ise o kullanıcılara kaynaklar rezerve edilebilir. Diğer kullanıcılar ise kalan kaynakları paylaşabilirler. Bundan başka, her kullanıcı için kaynakların küçük bir kısmı rezerve edilir. Kalan kaynaklar bir havuzda toplanır ve kullanıcılar bu kaynaklara erişmek için birbirleri ile yarışabilirler [3, 4].

1988 Kasım ayında Jacobson ve Karels “Tıkanıklıktan kaçınma ve Kontrol” (Congestion Avoidance And Control) isimli makale yayınladılar. Bu çalışmalarında, BSD TCP 4.0’da yer alan yedi algoritmadan beşini tanımladılar [2].

Bu yedi algoritma:

1. Round-trip time variance estimation.
2. Exponential retransmit timer backoff.
3. Slow-Start.
4. More aggressive receiver ACK policy.
5. Dynamic window sizing on congestion.
6. Karn's clamped retransmit backoff.
7. Fast Retransmit

1994 yılında C.E. Perkins ve P. Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers" isimli bir çalışma yaptılar. Bu çalışma, hareketli kablosuz ağlarda kullanılan ilk yönlendirme protokolünü tanımlamaktaydı [5]. 1996 yılında D. B. Johnson ve D. A. Maltz, "Dynamic Source Routing Protocol in Ad-Hoc Wireless Networks" isimli çalışmalarıyla kablosuz ağlarda kullanılan dinamik kaynak yönlendirmeli protokolü geliştirdiler [3]. 1997 yılında W. R. Stevens, "TCP Slow-Start, Congestion Avoidance, Fast Retransmission, and Fast Recovery Algorithms" isimli bir makale yayınladı. Bu makalesinde tıkanıklıktan kaçınmak için TCP'de neler yapılabileceğini, tıkanıklık meydana gelmesi durumunda hızlı bir biçimde nasıl giderilebileceğini açıkladı [2]. 1999 yılında M. Allman, V. Paxson ve W. Stevens'ın "TCP Congestion Control" isimli çalışmaları ve 2000 yılında Sally Floyd'un "Congestion Control Principle" isimli çalışmaları Internet Engineering Task Force (IETF) Request For Comments (RFC) standartları arasına girdi [6, 7]. 2001 yılında Dongkyun Kim, C. K. Toh ve Yanghee Choi, "TCP-BuS: Improving TCP Performans in Wireless Ad Hoc Networks" isimli çalışmasıyla, kablosuz ağlar için performansı arttırılmış bir TCP sürümü geliştirdi. 2001 yılında K. Chandran, S. Raghunathan, S. Venkatesan, ve R. Prakash "A Feedback-Based Scheme for Improving TCP Performans in Ad Hoc Wireless Networks" isimli çalışmalarıyla, kablosuz ağlar için değiştirilmiş olan bir TCP sürümü geliştirdiler [8]. 2002 yılında Zygmunt J. Haas, Marc R. Pearlman, ve Prince Samar, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks," isimli çalışmalarıyla yeni bir kablosuz ağ yönlendirme protokolü geliştirdiler. Bu protokol, ağda meydana gelen sel paketlerinin yayılma alanını önemli ölçüde azaltarak kaynakların verimli kullanılmasını sağlıyordu [9].

Bu çalışmada, tıkanıklık belirleme ve giderme işlemi yönlendirme katmanında gerçekleştirilmekte ve kablosuz ağlarda kullanılabilecek yeni bir yönlendirme protokolü anlatılmaktadır.

2. BİLGİSAYAR AĞLARINDA YÖNLENDİRME

Yönlendirme, bir ağ üzerinde bir düğümden diğer bir düğüme ulaşmak için kullanılacak olan yolun belirlenmesi ve bu yol üzerinden iletişime geçilmesidir. Bu yolun belirlenmesi için değişik yöntemler geliştirilmiştir. Paketlerin yönlendirilmesi (geçilecek düğümlerin belirlenmesi) iki şekilde gerçeklenir: kaynakta yönlendirme ve sekmelerde yönlendirme [10, 11]. Kaynakta yönlendirmede kaynak düğüm, paketin sırası ile geçeceği düğümleri (yönlendiricileri) belirler ve bu bilgiyi pakete ekler. Yönlendiriciler bu bilginin bulunduğu özel alana bakarak paketi sırası ile geçmesi gereken düğümlere aktarırlar. Bu yöntemde, yönlendirme işlemi kaynak düğümde yapılır. Diğer düğümler paketin belirlenen yol üzerinden geçmesini sağlar. Sekmelerde yönlendirmede ise paketin üzerindeki varış düğümü adresine bakarak paketin gönderileceği bir sonraki düğümün adresi belirlenir ve paket o düğüme aktarılır. Yol üzerindeki her sekmede (düğümde) bu işlem yapılır ve paket varış düğümüne kadar ulaştırılır.
Yönlendirme Protokolleri 2 türlüdür:

- Uzaklık Vektörü Protokolü
- Bağlantı Durumu Protokolü

2.1. Uzaklık Vektörü

Uzaklık vektörü yönlendirmesi dinamik bir yönlendirme algoritmasıdır. Bu teknikte her yönlendirici bir yönlendirme tablosu tutar. Bu tabloda ağdaki her yönlendirici için bir kayıt bulunur. Her kayıtta, ilgili yönlendiricinin tablonun bulunduğu yönlendiriciye olan uzaklığı ve ilgili yönlendiriciye hangi çıkış üzerinden ulaşılacağı bilgisi tutulur. Kullanılan ölçüt gecikme, sekme sayısı, kuyruk uzunluğu vs. olabilir. Gecikmenin ölçüt olarak kullanıldığı durumlarda, her yönlendirici kendisi ile komşu yönlendiriciler arasındaki gecikmeyi doğrudan bulabilir [12-14].

Uzaklık vektörü yönlendirmesinde, her yönlendirici (periyodik olarak) her 30 saniyede bir kendi tablosunda bulunan ölçüt değerlerini komşularına gönderir ve benzer bir tabloyu da komşusundan alır. Gelen tablolardaki verilere bakarak her yönlendirici doğrudan bağlı olmadığı yönlendiriciler ile arasındaki gecikme değerlerini ve o yönlendiricilere nasıl ulaşacağını bulabilir [12-14].

Uzaklık Vektörü protokollerinde yönlendirici, kendisine doğrudan bağlı diğer yönlendiricilerle bilgi alışverişinde bulunur ve bu yolla ağdaki yolların bilgisini edinir. Uzaklık Vektörü algoritması, yönlendirme tablolarını kullanıcı protokollerinin hizmetine sunmak için oluşturur. Uzaklık Vektörü algoritmasının oluşturduğu yönlendirme tablosu içerisinde şu bilgiler bulunur;

- Hedef ağın IP adresi
- Hedef ağa olan uzaklık
- Hedef ağ yolundaki ilk komşu yönlendiricinin IP adresi
- Yol bilgisinin hangi yönlendirme protokolü tarafından oluşturulduğu
- Yol bilgisinin en son güncellendiği zamandan bu yana geçen süre
- Yol bilgisi tabloları yönlendiricilerde dinamik olarak güncellenir.

Routing Information Protocol (RIP), Interior Gateway Routing Protocol (IGRP), AppleTalk Routing (AURP) uzaklık vektörü protokolleridir.

2.2. Bağlantı Durumu

Bağlantı durumu yönlendirmesi de dinamik bir tekniktir. Amacı, topolojideki değişimlere kolayca adapte olmak ve trafikteki değişimlere göre gerektiğinde alternatif yollar bulmaktır. Burada, yönlendiriciler bağlı oldukları hatlar (komşuları ile aralarındaki) üzerindeki gecikmeleri gösteren verileri ağdaki tüm yönlendiricilere ulaşması için gönderirler. Bu verileri alan yönlendiriciler ağ topolojisini oluşturur ve

diğer yönlendiricilere en kısa yoldan ulaşmak için bu topoloji üzerinde en kısa yol algoritmasını çalıştırırlar. Uygun yolları belirlerler. Hat durumunu gösteren paketler, en kolay sel yöntemi ile yayılabilir. Bu durumda paketlerin gereksiz yere ağ içinde dolaşmasını önlemek için hat durumu paketlerinin üzerine yaş (sekme sayacı) alanı koymak anlamlıdır [14]. Bağlantı durumu protokollerinde, yönlendiricilerin birer veritabanı bulunur. Bu veritabanı yolu ile tüm ağ hakkında bilgi sahibi olur.

2.2.1. Bağlantı durumu algoritması temel özellikleri

Yol bilgisi tablolarının değiştirilmesi:

- Her bir yönlendirici kendi yol bilgisi tablolarını komşularına bildirir.
- Yol bilgisi uçtan uca veya çoklu erişim şeklinde değiştirilebilir.
- Yönlendiricinin kendi komşularına tanımlamalarını göndermesine Bağlantı Durumu İlanı, (Link State Advertisement -LSA) denir. LSA tüm arayüz bilgilerini ve her bir bağlantı için ayarlanmış edilmiş maliyet değerini içerir.
- Bir yönlendiricide her bir arayüz için COST/TOS (maliyet/servis tipi) çiftleri vardır. LSA bu çiftlerin hepsini içerir.

Yol bilgisi alanları:

- LSA yönlendiricinin alanı içinden akar. Yönlendiricinin alanı tüm Otonom Sistem (Autonom System-AS) ya da AS içinde bir başka bölüm olabilir.
- Alanlar her bir yönlendirici arabiriminde atanan bir alan numarası ile ayarlanır

Bağlantı Durumu Veritabanı:

- Domaindeki her yönlendirici aynı bağlantı durumu veritabanını içerir.
- Yönlendirici her bir alan için ayrı ayrı bağlantı durumu veritabanı bulundurur.

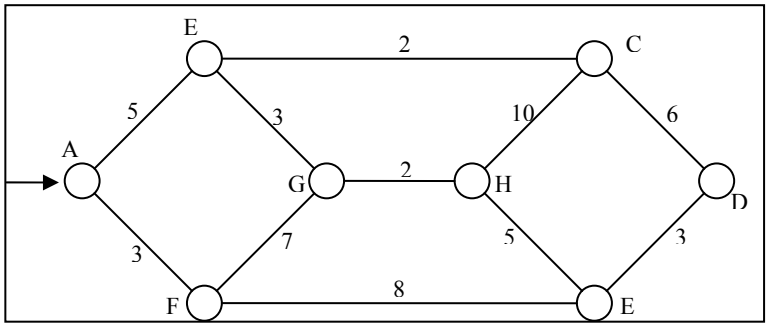
En Kısa Yol Ağacı:

- En kısa yolu hesaplayacak bir algoritma içerir. Her bir TOS için ayrı kısa yol ağaçları kullanılır.
- En kısa yol ağacı belirli bir TOS için her bir yönlendiriciye ve her bir ağa olan en kısa yolu içerir ve bu yolla diğer yönlendiriciye erişilir.

Dezavantajları ise şu şekilde sıralanabilir: Veritabanı büyük bir yer kaplayabilir. İkincisi, bir yol çöktüğü zaman yeni yolların ayarlanması gerekir ki bu da fazla hesap demektir. Open Shortest Path First (OSPF), Enhanced Interior Gateway Routing Protocol (EIGRP), Border Gateway Protocol (BGP) bağlantı durumu protokolleridir.

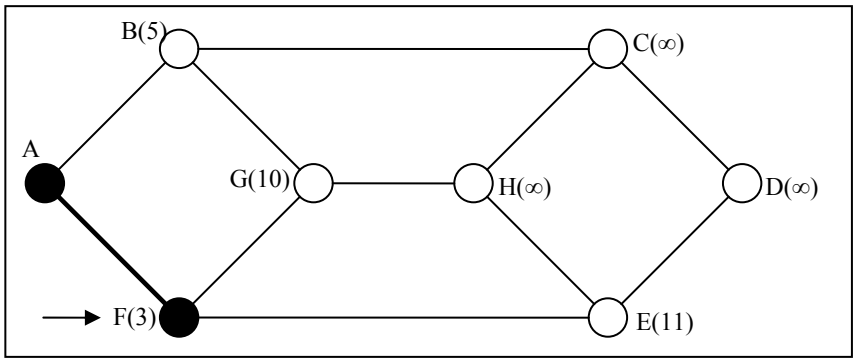
2.3. Dijkstra'nın En Kısa Yol Algoritması

Yönlendirme tekniklerinde akla gelen ilk şey, iki nokta arasındaki en kısa mesafenin bulunması olacaktır. Bilgisayar ağlarında iki nokta arasındaki en kısa yolu bulurken ölçüt olarak bağlantı noktaları arasındaki coğrafi uzaklık, geçilen düğüm (sekme) sayısı ya da hatlar üzerinde ortaya çıkan aktarım süreleri ve düğümlerdeki kuyruklarda bekleme süreleri kullanılabilir. Sonuçta, kullanılan ölçüt ne olursa olsun amaç en kısa yolun bulunmasıdır. Anlatılacak olan en kısa yol bulma algoritması Dijkstra tarafından geliştirilmiştir [12].

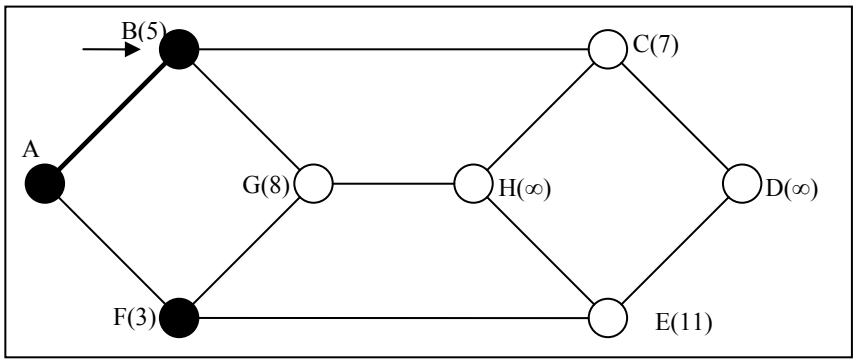


Şekil 2.1. Örnek ağ yapısı

Şekil 2.1'deki durum için önce A düğümünden başlanır ve ona komşu olan düğümler incelenip onlara geçici uzaklık değerleri atanır. Şekil 2.2(a)'da B ve F düğümleri için belirlenmiş geçici uzaklık değerleri bulabilirsiniz. Diğer düğümler henüz incelenmediği için onların geçici uzaklık değerleri sonsuz işareti ile gösterilmiştir [12].

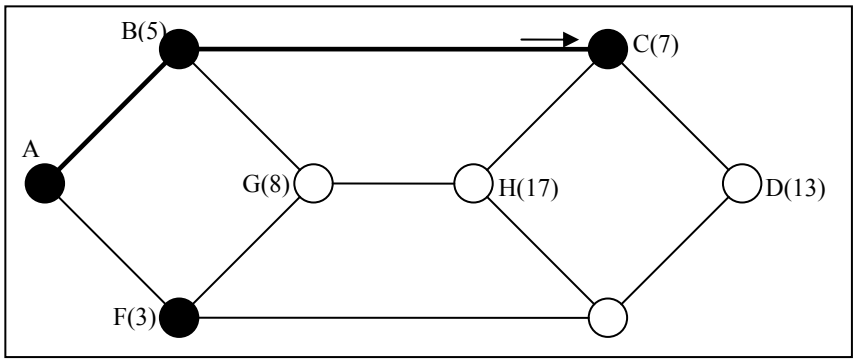


(a)

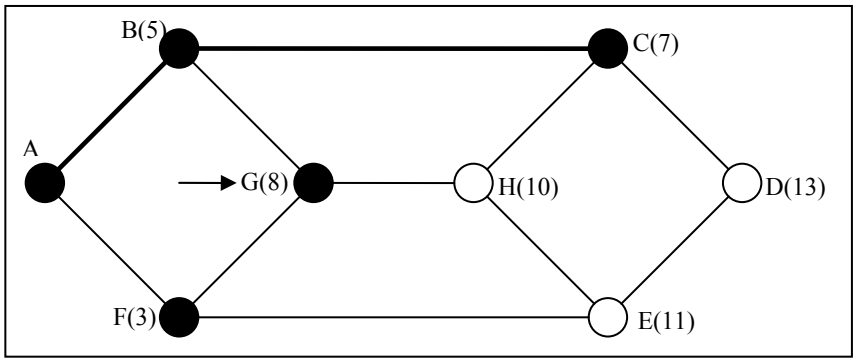


(b)

Şekil 2.2. Geçici uzaklık değerlerinin belirlenmesi
a) Geçici uzaklık değerlerinin belirlenmesi ilk adım
b) Geçici uzaklık değerlerinin belirlenmesi ikinci adım
c) Geçici uzaklık değerlerinin belirlenmesi üçüncü adım
d) Geçici uzaklık değerlerinin belirlenmesi dördüncü adım
e) Geçici uzaklık değerlerinin belirlenmesi beşinci adım
f) Geçici uzaklık değerlerinin belirlenmesi altıncı adım
g) Geçici uzaklık değerlerinin belirlenmesi yedinci adım

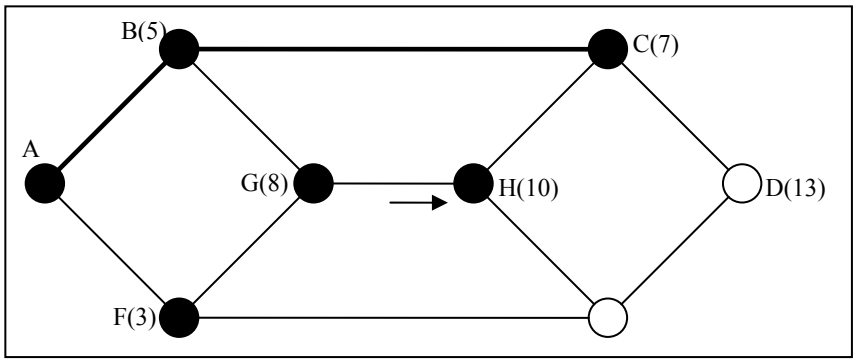


(c)

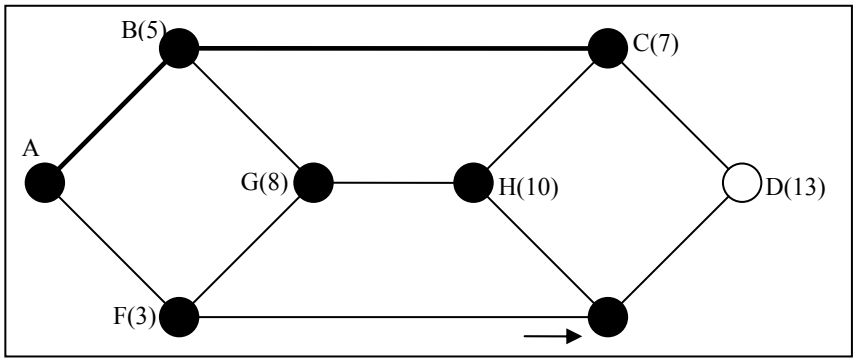


(d)

Şekil 2.2. (Devam) Geçici uzaklık değerlerinin belirlenmesi
a) Geçici uzaklık değerlerinin belirlenmesi ilk adım
b) Geçici uzaklık değerlerinin belirlenmesi ikinci adım
c) Geçici uzaklık değerlerinin belirlenmesi üçüncü adım
d) Geçici uzaklık değerlerinin belirlenmesi dördüncü adım
e) Geçici uzaklık değerlerinin belirlenmesi beşinci adım
f) Geçici uzaklık değerlerinin belirlenmesi altıncı adım
g) Geçici uzaklık değerlerinin belirlenmesi yedinci adım

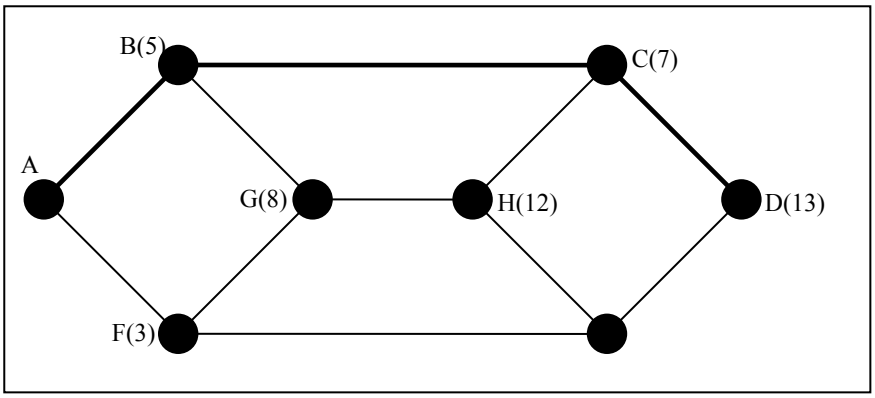


(e)



(f)

Şekil 2.2. (Devam) Geçici uzaklık değerlerinin belirlenmesi
a) Geçici uzaklık değerlerinin belirlenmesi ilk adım
b) Geçici uzaklık değerlerinin belirlenmesi ikinci adım
c) Geçici uzaklık değerlerinin belirlenmesi üçüncü adım
d) Geçici uzaklık değerlerinin belirlenmesi dördüncü adım
e) Geçici uzaklık değerlerinin belirlenmesi beşinci adım
f) Geçici uzaklık değerlerinin belirlenmesi altıncı adım
g) Geçici uzaklık değerlerinin belirlenmesi yedinci adım



(g)

Şekil 2.2. (Devam) Geçici uzaklık değerlerinin belirlenmesi
a) Geçici uzaklık değerlerinin belirlenmesi ilk adım
b) Geçici uzaklık değerlerinin belirlenmesi ikinci adım
c) Geçici uzaklık değerlerinin belirlenmesi üçüncü adım
d) Geçici uzaklık değerlerinin belirlenmesi dördüncü adım
e) Geçici uzaklık değerlerinin belirlenmesi beşinci adım
f) Geçici uzaklık değerlerinin belirlenmesi altıncı adım
g) Geçici uzaklık değerlerinin belirlenmesi yedinci adım

Daha sonra, sırası ile A'dan ulaşılan en yakın düğümden (bu örnekte F) başlayarak uzaklık belirleme işlemine devam edilir. F'nin seçiminden sonraki uzaklık değerleri Şekil 2.2(b)'de verilmektedir. B, C, G, H ve E'nin seçiminden sonraki uzaklık değerleri Şekil 2.2 (c-g)'de gösterilmektedir [12]. Yukarıdaki örnekte A düğümüne olan en kısa yollar hesaplanmıştır. Yani kaynak düğüm A'dır. Aynı yöntem kullanılarak diğer kaynak düğümler için de hesaplama yapılır [14]. Dijkstra'nın algoritması, aşağıdaki şekilde tanımlanabilir [12].

```
function Dijkstra(L[1..n, 1..n]) : array[2..n]
     array D[2..n]
     set C          { incelenecek düğümleri saklar}

     C← {2, 3, 4, 5, 6, …, n}
     for i← 2 to n
          D[i]← L[1, i]
     Repeat n-2 times
          v← C'den en küçük D[v] değerini saklayan bir
     eleman
          v← C\ {v}
          for each w elemanıdır C do
               D[w]← min(D[w], D[v]+L[v, w])
     Return D
```

2.4. Tıkanıklık Belirleme ve Çözüm Yöntemleri

Bir ağda tıkanıklığın olup olmadığı aşağıdaki unsurlara dikkat edilerek anlaşılabilir.

- Eğer düğümdeki çıkış tamponu dolu ise ve giriş tamponunda yer yoksa tıkanıklık meydana geldi denebilir.
- Düğüm, çıkış hattını gözlemler. Eğer çıkış hattı üzerinde %80 gibi bir eşik değerinin üstünde yoğunluk varsa bu hat tıkanacak demektir.
- Gidiş-Dönüş gecikme süreleri incelenebilir. Kuyruk uzunluğu arttıysa ve gidiş-dönüş gecikme süresi yükseldiyse tıkanıklık meydana gelebilir demektir.
- Düğüm bir sayaç tutar. Eğer gönderilen paket gereken süre içerisinde kabul edilmezse tıkanıklık vardır.

2.4.1. TCP'de tıkanıklık denetimi ve çözüm yöntemleri

TCP bağlantısındaki akış, paketlerin korunumu yasası'na uygun olmalıdır. Paketlerin korunumunun anlamı, akışın "eşitlik" içerisinde olmasıdır. Yani, tamamen dolu olan bir veri aktarım penceresinde eski paketin çıkmadan yeni paketin alınmamasıdır [5].

Paketlerin korunumunu başarısız kılacak 3 adet neden vardır:

1- Bağlantı, korunum ilkesini yerine getiremez.
2- Bir gönderici, eski bir paket alınmadan çıkmadan yeni bir tane gönderir.
3- Yol kaynaklarının yetersizliği yüzünden eşitlik ilkesi yerine getirilemez.

Birinci başarısızlık nedeni, bir paketin ilk defa iletilmesi veya yeniden iletilmesi durumunda ortaya çıkar. Bir düğüm tarafından alınan paketin başka bir düğüme iletilmesinde sorun varsa ve bu sırada diğer düğümden paketler gelmeye devam ediyorsa bu başarısızlık meydana gelir. Bunu, paketlerin bir düğümde yığılması olarak da değerlendirebiliriz. Korunum özelliğine bir başka açıdan bakacak olursak, gönderici aldığı ACK (acknowledge) paketlerinin hızına bakarak gönderme işlemine devam ederse sistem kendini ayarlamış olur. Fakat bu durum, sistemin kendisini sınırlamasına yol açar. Yeni bir paketin iletilmesi için bir ACK gereklidir. Bir ACK almak için de bir paketin iletilmesi gereklidir. Zamanlamaya başlamak için slow-start algoritması geliştirilmiştir. Bu algoritma şu şekilde çalışır:

- Her bir bağlantı için bir tane tıkanıklık penceresi, *cwnd* ekle.
- Bir kayıp olduğu zaman *cwnd*'yi bir pakete ayarla.
- Alınan her bir ACK'den sonra *cwnd*'yi birer paket arttır.
- Veri gönderirken, alıcının ilan ettiği pencere ile cwnd arasından en küçük olana göre işlem yap.

İkinci başarısızlık nedeni de eşitlik ilkesinin korunmamasıdır. Bu ilkeyi korumak için iyi bir gidiş-dönüş zamanlaması'na ihtiyaç vardır. İyi bir gidiş-dönüş zamanlaması kestirimi, tekrar iletim zamanlayıcısının çekirdeğini oluşturur. Bu da, bir protokolün ağır yükler altında çökmemesi için gerekli olan en temel bileşenlerdendir. Kuyruk

teorisinden şu bilinmektedir; yükle birlikte gidiş dönüş zamanı (round-trip time - RTT) ve onun değişimi hızla artmaktadır.

TCP, ortalama RTT'yi hesaplamak için bir alçak geçiren filtre tanımlamaktadır.

$$SRTT = \alpha * SRTT + (1-\alpha) * RTT \quad (2.1)$$

SRTT, hesaplanmış ortalama RTT'dir. RTT, en son geri bildirilmiş olan veri paketi ile α filtre kazancı kullanılarak hesaplanır. Önerilen α değeri 0,9 olarak sabittir. Bir defa SRTT hesabı güncellendiği zaman, tekrar iletim zaman aşımı aralığı (retransmit timeout interval - RTO), gönderilen sonraki paket için β * SRTT şeklinde ayarlanır. β parametresi RTT değişimi olarak görülür. Önerilen β=2 değeri en çok %30 yük içindir. Bu durumda bir bağlantı, iletim sırasında geciken paketler için yeniden gönderme yoluna gidecektir. Bu da, ağı faydasız olmaya iter, bant genişliğinin boş yere kullanılmasına ve paketlerin birden fazla kopyalarının ağda dolaşmasına neden olur.

Van Jacobs ve Karels, değişim kestirimi için basit bir yöntem geliştirmişlerdir. Bu yöntem β değerinin sabit olarak değil, dinamik olarak tutulması esasına dayanır. RTO, dinamik β değerine göre hesaplanır. Algoritma, SRTT'nin ortalama sapmasını hesaplar ve bu değeri RTO değerinin kestiriminde kullanır.

$$SRTT(i+1) = (1-\alpha) * SRTT(i) + \alpha * RTT(i+1) \quad (2.2)$$
$$RTO = \beta * SRTT \quad (2.3)$$

Bu durumda β dinamiktir ve SRTT'nin ortalama sapmasına bağlıdır.
SRTT(i) = (1-α) * SRTT(i-1) + α * RTT(i) denkleminde, SRTT, gerçek ortalama etrafında rasgele dalgalanmalar gösterecektir. SRTT'nin bir standart sapmasını almak muhtemel tüm olasılıkları kapsamış olur.

SRTT'nin standart sapması α * standartsapma(RTT) olacaktır. Standart sapma yerine ortalama sapmayı kullanmak daha iyidir. Çünkü Hesaplamak için daha az işlemci kaynağı gerekir ve değişim kestiriminde, standart sapmadan daha ılımlı sonuç verir.

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

$$\text{OrtalamaSapma} = \text{sqrt}(\pi)/2 * \text{StandartSapma} \sim 1.25 * \text{StandartSapma} \quad (2.4)$$

Standart sapmanın bir değeri olarak ortalama sapmayı hesaplamak işlemci açısından daha kolaydır. RTO(i)'yi (i anındaki RTO değeri) hesaplamak için kullanılan yeni algoritma RTO(i) = RTT(i) + 4 * V(i) şeklinden tanımlanmaktadır. RTT(i) i anındaki RTT değeri ve V(i) i anındaki ortalama sapmayı göstermektedir. Bu algoritma, yavaş başlama esnasında sahte yeniden iletim zaman aşımını engellemek için gerçekleştirilmiştir. Eğer RTO(i), yavaş başlama gönderiminin sonunda hesaplanmış ise sahte yeniden iletim meydana gelir. Bu durumda hesaplanan değer, sonraki gönderim için olması gereken gerçek RTT değerinden küçük veya ona eşit olacaktır. En kötü durumda, RTT her bir gönderimi iki defa yapacaktır (pencere boyutunun iki katına çıkmasından ötürü). Böylece:

$$RTT(i+1) = 2 * RTT(i) \quad (2.5)$$

$$V(i) = RTT(i) - RTT(i-1) = RTT(i) - ½ * RTT(i) = ½ * RTT(i) \quad (2.6)$$

Eğer RTO(i)'yu RTT(i) + 4 * V(i) şeklinde tanımlarsak:

$$RTO(i) = RTT(i) + 4 * V(i)$$

$$RTO(i) = RTT(i) + 4 * ½ * RTT(i) = 3 * RTT(i)$$

$$RTO(i) = 3 * RTT(i) > 2 * RTT(i) \quad (2.7)$$

Elde edilir.

2 * RTT(i) = RTT(i+1) olduğu için RTO(i) > RTT(i+1) olur ve böylece sahte yeniden gönderim engellenmiş olur.

3. başarısızlık nedeni üzerinde durulacak olursa, zamanlayıcılar iyi durumda iken bir zaman aşımı söz konusu olursa bu, bir paket kaybının olduğunu gösterir. Ancak, paket kayıplarının %99'u tıkanıklık durumunda olur. Tıkanıklıktan kaçınma stratejisinin iki temel bileşeni vardır:

- Ağ, tıkanıklık olduğu zaman veya tıkanıklığa yaklaşıldığı zaman bunu bir sinyal şeklinde uç birimlere bildirebilmelidir.
- Uç birimler bir sinyal aldıkları zaman ağın performansını düşürüp tıkanıklığı engelleyebilmeli, sinyal almadıklarında ise ağın performansını arttırabilmelidirler.

Paket kaybı genellikle ağ tıkanıklığından dolayı meydana gelir. Zaman aşımı da paket kaybından dolayı meydana gelir. Böylece, zaman aşımının ağ tıkanıklığından dolayı meydana geldiğini söyleyebiliriz. Bir ağdaki yükün ölçütü olarak, gidiş-dönüş zaman gecikmesinin neden olduğu ortalama kuyruk uzunluğunu alınırsa, tıkanmamış bir ağ bu parametrelere bağlı olarak ifade edilebilir. Bir ağdaki anlık yük L(i)=N olarak ifade edilebilir. Zamana bağlı olarak bir ağdaki yük L(i) =N+g*L(i-1) şeklinde ifade edilebilir. Bu ifade, zaman gecikmesi nedeni ile kuyrukta kalan yükün, sonraki zaman birimindeki trafiğe etkisidir. Eğer ağda tıkanıklık varsa, g değeri büyük olacaktır ve kuyruk uzunluğu üssel olarak artacaktır. Tıkanıklık durumunda pencere boyutu şu şekilde ayarlanır:

- W(i) = d * W(i-1) burada d < 1

Tıkanıklık kaybolduğu zaman ise pencere boyutu aşağıdaki gibi ayarlanır:

- W(i) = W(i-1) + μ burada μ <= Wmax

Buradaki Wmax değeri, ağın hiç yük olmaması durumundaki pencere boyutunu ifade eder. Tıkanıklıktan kaçınma algoritmasını yazan kişiler d=0.5, μ=1.0 almışlardır. Böylece, algoritma:

1. Herhangi bir zaman aşımında, cwnd'yi geçerli pencere boyutunun yarısına ayarla.
2. Her bir yeni veri için alınan ACK ile, cwnd'yi 1 arttır.
3. Veri gönderirken, cwnd ile alıcının duyurduğu pencere boyutundan küçük olanına göre gönder.

Şeklinde ifade edilir.

2.4.2. Kablosuz ağlar için değiştirilmiş TCP sürümleri

Geleneksel TCP'nin kablosuz ağlarda kullanılmasının çeşitli sakıncaları vardır. Bunlar aşağıda özetle anlatılmaktadır.

Paket Kayıplarını Yanlış Yorumlama: Geleneksel TCP, kablolu ağlar için hazırlanmıştır ve paket kayıplarını tıkanıklık belirtisi olarak değerlendirir. Bir paket kaybolduğu zaman tıkanıklık çözümü için geliştirilmiş olan algoritma işletilir. Hareketli tasarsız ağlarda, kablolu ağlara göre çok daha fazla paket kaybı yaşanmaktadır. Tasarsız ağlarda sık sık düğümlerin hareketliliğinden ötürü yol kırılmaları meydana gelmektedir, radyo sinyallerinin girişimde bulunması, tek yönlü bağlantılar gibi nedenlerden dolayı paket kayıpları yaşanmaktadır. Bu kayıpların tıkanıklık olarak değerlendirilmemesi gerekmektedir.

Sık Bağlantı Kopmaları: Hareketli tasarsız ağlarda düğümlerin hareketliliği ile ilgili olarak bir kısıtlama olmadığı için sık sık bağlantı kopmaları yaşanmaktadır. Bir bağlantı

koptuktan sonra, yeni bir yolun temin edilmesi gerekmekte ve ağ üzerinde çalıştırılan yönlendirme algoritması bunu yapmaktadır. Yeni yolun kurulma süresi eğer tekrar iletim zaman aşımı aralığı (retransmit timeout interval) RTO'dan büyükse, gönderici düğüm bir tıkanıklık olduğu düşünüp, tıkanıklık algoritması çalıştıracaktır ve kaybolan paketleri yeniden gönderecektir. Bu da bant genişliğinin azalmasına ve bataryanın boş yere kullanılmasına neden olacaktır.

Yol Uzunluğunun Etkisi: Hareketli tasarsız ağlarda yol uzunluğunun büyük olması demek, hedefe daha fazla düğüm üzerinden geçilerek gidilmesi demektir. Bu da, düğümlerin hareketliliğinden dolayı daha fazla yol kırılmasına yol açar.

Tıkanıklık Penceresinin Yanlış Kullanılması: Tasarsız ağlarda bir yol kırılması meydana geldiği zaman tıkanıklık önleme algoritması işletilir ve tıkanıklık pencere boyutu azaltılır ve RTO'yu arttırır. Yeni yol kurulduğu zaman, tıkanıklık penceresi buna hemen uyum sağlayamaz. Yeni kurulmuş olan yolun kapasitesi yüksek olmasına rağmen bunun küçük bir kısmı kullanılabilir. Bundan dolayı, sık sık yol kırılmaları hattın verimliliğini düşürür.

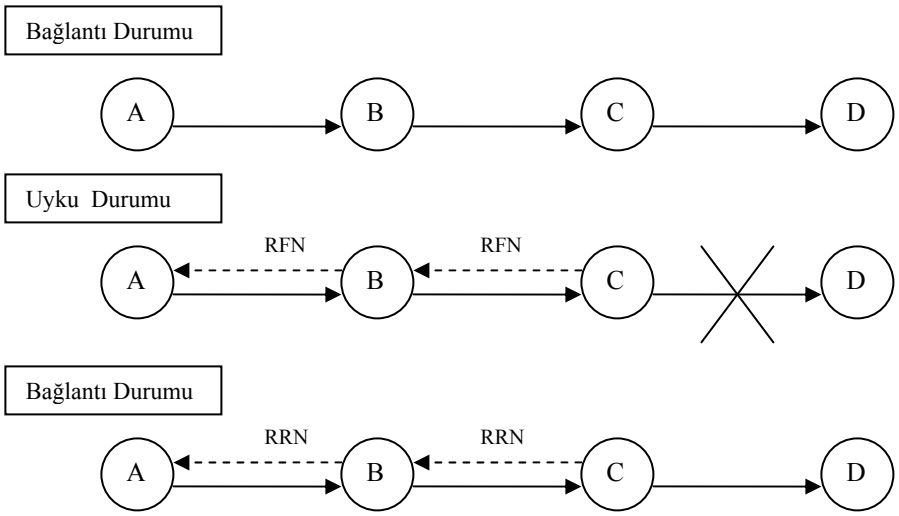
Asimetrik Bağlantı Davranışı: Hareketli tasarsız ağda kullanılan radyo kanalı farklı özelliklere sahip olabilir. Radyo dalgalarının yayılımının çevresel etkisi, konumdan dolayı meydana gelen çekişme gibi nedenlerden ötürü hat tek yönlü olabilir. Bu da, ACK paketlerinin iletilmesini engeller. Bu durumda, tıkanıklık algoritmasını çalıştırmak mümkün değildir.

Aşağıda, üç adet protokol anlatılmaktadır. Bu protokoller, geleneksel TCP'nin değiştirilmesi ile oluşturulmuştur.

Geri beslemeli TCP

Geri Beslemeli TCP (Feedback-Based TCP, TCP-F), tasarsız ağlarda performansı arttırmak için geleneksel TCP'nin değiştirilmesi ile oluşturulmuştur. Bu protokol, geri besleme tabanlı bir yaklaşım kullanır. TCP-F, oluşabilecek bağlantı kopmalarını kaynak düğüme bildirebilecek bir yönlendirme yöntemine ihtiyaç duyar. Kopan bağlantının yerine yenisinin belirlenmesi işlemi kabul edilebilir süreler içerisinde gerçekleştirilmelidir. Bu protokolün amacı, bağlantı kopmalarından meydana gelen paket kayıplarını azaltmaktır. Bir paket kaybolduğu zaman tıkanıklık önleme algoritması işletilir ve pencere boyutu küçültülür [8, 10].

TCP-F'te, bir ara düğüm, bir bağlantı kopması tespit ettiği zaman route failure notification (Yol Hatası Bildirimi-RFN) paketi yayınlar. Bu paket, gönderici düğüme doğru iletilir. RFN paketi yayınlayan düğüme failure point (hata noktası-FP) adı verilir. FP, o zamana kadar oluşturduğu tüm RFN paketlerini saklar. RFN paketini alan her ara düğüm, bir bağlantı kopukluğu meydana geldiğini anlar, yönlendirme tablolarını günceller ve artık o yola veri göndermekten sakınır. RFN paketini alan herhangi bir ara düğüm, hedefe giden başka bir yola sahipse RFN paketini yok eder ve yeni yolu kullanmaya başlar. Bu da, kontrol yükünü azaltır. Diğer durumda RFN paketi kaynak düğüme kadar gider. Bir RFN paketi alan kaynak düğüm, uyku konumuna geçer. Uyku konumuna geçen bir düğüm, tüm sayaçları sıfırlar, tıkanıklık penceresini başlangıç konumuna ayarlar ve bir yol başarısızlığı sayacı ayarlar. Bu sayaç, yönlendirme protokolüne, ağın büyüklüğüne göre değişir ve en kötü ihtimalle yol kurma süresine ayarlanır. Sayaç bittiği zaman düğüm kendisini uyku konumundan bağlantı konumuna alır ve iletime devam eder. Sayaç bitmeden bir yol kurulu ise, kaynak düğüme bir yeniden kurulma bildirimi paketi (reestablishment notification packet - RRN) yollanır ve gönderici düğüm uyku konumundan bağlantı konumuna geçer.



Şekil 2.3. TCP-F'nin işleyişi

Avantajları ve dezavantajları

Bu yöntemde, aynı zamanda hem bağlantı kopmalarına yönelik hem de tıkanıklık önlemeye yönelik işlemler bir arada yapılır. Bağlantı kopmasının tespiti, ara düğümlerin güvenilirliği ile yakından ilişkilidir. Bir yol kırılması, tıkanıklık gibi değerlendirildiği için, yeni yolun kurulmasından sonra o bağlantı tam kapasite ile çalıştırılamaz. Bu da bir dezavantaj olarak karşımıza çıkmaktadır.

Açık bağlantı başarısızlığı bildirimli TCP

Holland ve Vaidya, Açık Bağlantı Başarısızlığı Bildirimli TCP'yi (TCP With Explicit Link Failure Notification, TCP-ELFN) tasarsız ağlarda performansı arttırmak için geliştirmişlerdir. Bu protokol, TCP-F ile benzer özelliklere sahiptir. Fakat açık bağlantı başarısızlığı bildirimini yakalaması ve yeniden yol kurulduğunda bunu tespit etmesi açısından farklıdır. ELFN (Açık Düğüm Hata Bildirim) mesajı, bağlantı kopukluğunu tespit eden düğüm tarafından, kaynak düğüme bu kopukluğu bildirmek için üretilir. Bu,

iki yolla olur: ilk yol, ICMP paketi ile “hedefe ulaşılamıyor” mesajı yollamaktır. Diğer yol ise, Yol Hatası mesajı ile bu durumu bildirmektir [10].

Bir kaynak düğüm ELFN mesajı aldığı zaman, yeniden gönderim sayacını durdurur ve bekleme konumuna geçer. Bu süreç içerisinde, yeni bir yol kurulup kurulmadığını anlamak için sürekli olarak ağı izleyen paketler çıkarır. Bir izleme paketi için ACK gelirse, yolun yeniden kurulduğunu anlar ve gönderme işlemine kaldığı yerden devam eder.

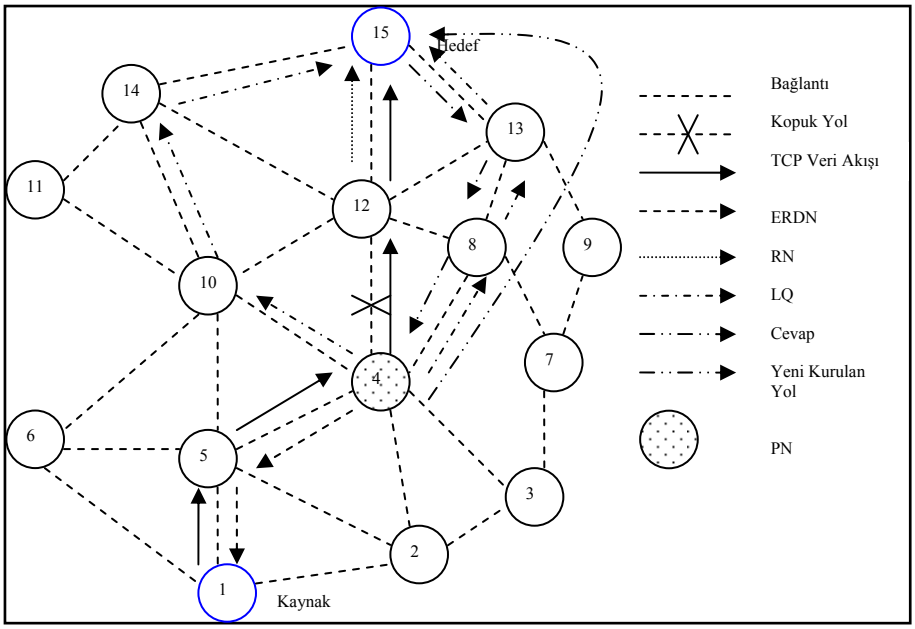
Avantajları ve dezavantajları

ELFN paketi sayesinde bağlantı kopukluğu ile tıkanıklık durumu ayırt edilebilmektedir. Bu da, gereksiz yere tıkanıklık yordamlarının çalıştırılmasını engeller. Ağın, geçici bir süre için parçalanması durumunda gereksiz yere izleme mesajları ile ağın bant genişliği düşecektir ve gereksiz yere batarya kullanılacaktır. Bu, TCP-ELFN’nin dezavantajıdır.

Tamponlama yapabilen ve sıra bilgisi kullanan TCP

Aradaki düğümün, bağlantı kopukluğunu tespit etme ve bunu kaynak düğüme geri bildirme yöntemleri konusunda Tamponlama Yapabilen ve Sıra Bilgisi Kullanan TCP (TCP With Buffering Capability and Sequence Information, TCP-BuS), TCP-F ve TCP-ELFN protokollerine benzer. TCP-F ve TCP-ELFN ile karşılaştırıldığında, yönlendirme protokolüne daha fazla bağımlıdır. TCP-BuS, dayanışma tabanlı yönlendirme protokolü üzerine kurulmuştur. Bu yüzden, bu protokolde olan belirli paketleri kullanır. Aktarımın olduğu her ara düğüm, aktarılan paketleri bir tampondan tutar. Aktarılan paket, hemen yok edilmez. Bir ara düğüm, bağlantı kopukluğu tespit ettiği zaman “açık yol bağlantı kopukluğu bildirimi” (explicit route disconnection notification-ERDN) paketi yayınlar. Bağlantı kopukluğunu yakalayan ara düğüme pivot node (PN) denir. PN’de TCP paketlerinin sıra numaraları da tutulur. ERDN paketi, kaynak düğüme doğru yayılır. Bu

paketi alan kaynak düğüm, tüm sayaçları ve pencereleri dondurur. Kopukluğun olduğu bağlantının diğer ucunda bulunan düğüm, hedef düğüme doğru bir "yol bildirimi" (route notification-RN) paketi yollar. Bu paket içerisinde, hedefe giden yolda olan tüm düğümlerin adresleri bulunur. Bu paketi alan ara düğümler, o bağlantıya ait aktarmakta oldukları paketleri yok ederler. Bundan sonra, kullanılan yönlendirme protokolünün kuralları çerçevesinde yol kurma işlemi başlar. PN tarafından yeni bir yol bulunduğu zaman, yerelleştirilmiş sorgu (localized query - LQ) ile hedef düğüme hangi sıra numaralı paketleri aldığı sorulur. Hedef düğüm, cevap paketi ile buna cevap verir ve hatalı sıra ile iletilmiş olan paketler, tampondan yeniden iletilir. Daha sonra, kaynak düğüme yollanmak üzere bir "açık yol başarılı bildirimi" (explicit route succesful notification - ERSN) paketi üretilir. Bu paketi alan kaynak düğüm, o zamana kadar gönderdiği tüm paketlerin hedef düğüme vardığına emin olur ve paket gönderme işine kaldığı yerden devam eder. Şekil 2.4'de TCP-BuS'ın çalışması gösterilmektedir [10, 15].



Şekil 2.4. TCP-Bus'ın işleyişi

Avantajlar ve dezavantajları

Tampon hafıza, sıra numarası ve seçmeli geri bildirim mekanizması çalıştırıldığından dolayı bu yöntemde yeniden iletim hızlı bir biçimde yapılabilmektedir. Yönlendirme protokolüne aşırı bağımlılığı bir dezavantaj olarak görebiliriz. Tampon görevi gören ara düğümlerden birinin başarısız olması durumunda kayıp paketler artacak ve başarı düşecektir.

3. KABLOSUZ AĞLARDA YÖNLENDİRME PROTOKOLLERİ VE TIKANIKLIK DENETİMİ

Kablosuz yönlendirme protokolleri farklı birçok kıstasa göre birçok gruba ayrılabilir. Bu sınıflandırmada kesin sınırlar yoktur. Bir yönlendirme protokolü birçok gurubun altına düşebilir. Genellikle, bu protokoller 4 guruba ayrılır [1]:

1- Yönlendirme Bilgisi Güncelleme Mekanizması
2- Yönlendirme İçin Geçici Bilgi Kullanmak
3- Yönlendirme Topolojisi
4- Belirli Kaynakların İyi Kullanımı

3.1. Yönlendirme Bilgisi Güncelleme Mekanizması Tabanlı Olanlar

Kablosuz yönlendirme protokolleri bu başlık altında 3 ana başlığa ayrılırlar.

Kaynaktan yönlendiren veya tablo kullanan yönlendirme protokolleri

Bu yöntemde her düğüm tüm ağın topolojik bilgisini bir tabloda tutar. Bu tablodaki bilgiler periyodik olarak güncellenir ve düğümler arasında değiştirilir. Bir düğüm bir yol bilgisine ihtiyaç duyduğu zaman bu tabloya bakar.

Sekmelerde yönlendiren veya talep üzerine iş yapan yönlendirme protokolleri

Bu guruptaki protokoller tüm ağın topolojik yapısını tutmazlar. İstenilen bir yolu bulmak için o anda girişimde bulunurlar. Bunun için bağlantı tabanlı bir işlem yürütülür. Periyodik olarak bilgi alış-verişi bu tip protokollerde yoktur.

Karma yönlendirme protokolleri

Bu tipteki protokoller, yukarıdaki protokollerin iyi yanlarını birleştirmişlerdir. Her düğümün bir yönlendirme alanı vardır. Bu alandaki bir düğüme erişecekleri zaman bu düğümlerin tutulduğu tabloya bakarlar. Erişmek istedikleri düğüm eğer yönlendirme alanının dışında ise bu durumda o anda bir istekte bulunurlar.

3.2. Yönlendirme İçin Geçici Bilgi Kullananlar

Kablosuz Tasarsız ağlarda düğümlerin hareketliliği ve bağlantıların kopması kablolu ağlara göre çok daha sık gerçekleşir. Bu yüzden seçilmiş olan yolun yaşam süresinin değerlendirilerek geçici bilgiler ile yönlendirme yapmak daha doğru olacaktır.

Geçmiş ve o andaki duruma bakarak yönlendirme

Bu guruptaki yönlendirme algoritmaları, bağlantıların geçmiş durumlarına ve o andaki durumlarına bakarak kara verirler. Örneğin, o andaki yollardan en kısası üzerinden paketler iletişime geçirilebilir.

Gelecekteki durumu tahmin ederek yönlendirme

Bu guruptaki yönlendirme algoritmaları, bağlantıların gelecekteki durumlarını tahmin ederek yönlendirmeye karar verirler. Gelecekteki durum tahminleri o düğümün batarya durumu, hareket yönü gibi kriterler göz önünde bulundurularak yapılır.

3.3. Yönlendirme Topolojisi Tabanlı Olanlar

Düz topoloji yönlendirme protokolleri

Bu kategorideki protokoller ağdaki her bir düğüm için (ya da en azından ağa bağlı bir düğüm için) tekil bir adres bilgisi kullanırlar.

Hiyerarşik topoloji yönlendirme protokolleri

Bu guruptaki protokoller ağ üzerinde mantıksal bir hiyerarşi kullanırlar ve bunu adresleme planı ile ilişkilendirirler. Bu hiyerarşi coğrafi bilgi tabanlı olabileceği gibi atlama sayısı ile ilgili de olabilir.

3.4. Belirli Kaynakların En İyi Kullanımı Tabanlı Olanlar

Güç-haberdar yönlendirme

Bu kategorideki algoritmalar tasarsız ağlarda çok önemli bir kaynak olan batarya'yı verimli kullanmayı amaçlarlar. Yönlendirme kararı yerel veya genel olarak batarya tüketimini en aza indirmeye yönelik verilir.

Coğrafi bilgi destekli yönlendirme

Bu guruptaki protokoller coğrafi bilgi kullanarak kontrol yükünü en aza indirmeyi amaçlarlar.

3.5. Hareketli Kablosuz Ağlarda Yönlendirme Protokolleri

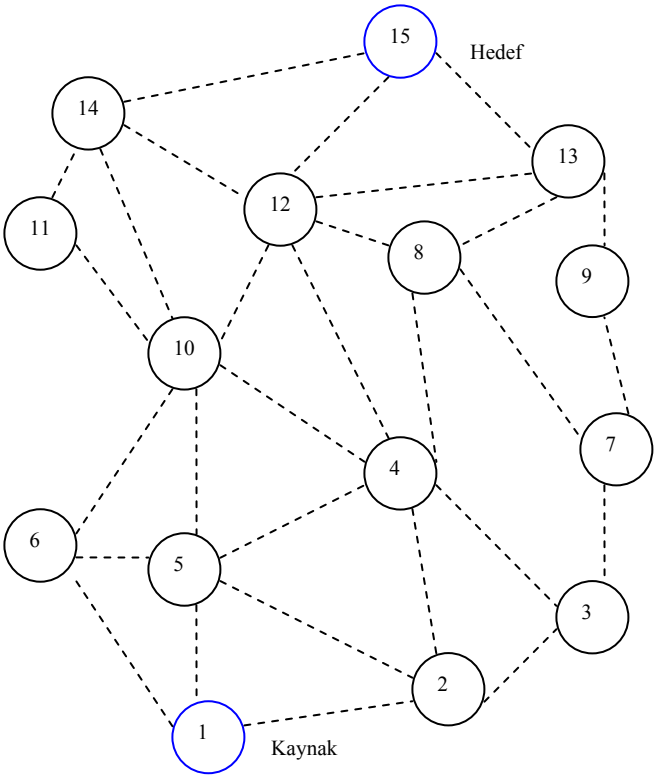
3.5.1. Hedef sıralı uzaklık vektörü yönlendirme protokolü

Hedef Sıralı Uzaklık Vektörü Yönlendirme Protokolü (Destination sequenced distance-vector routing protocol – DSDV) kablosuz ağlar için tasarlanan ilk protokollerden biridir. Her bir düğüm, ağdaki diğer düğümler için bir yol bulunan tablo tutar. Her satırda hedef düğüm, sonraki düğüm, uzaklık ve sıra numarası kayıtlıdır. Sıra numarası o yolun güncelliğini gösterir. Bir düğüm topolojide anlamlı bir değişiklik gördüğü zaman kendi tablosunu diğer düğümlere gönderir. Değişiklik büyük ise bütün tablo gönderilir, değilse sadece o parçası gönderilir. Bir düğüme giden yol ile ilgili bir değişiklik olduğu zaman bu durum komşu düğümlere daha yüksek bir sıra numarası ile bildirilir. Bu

değişikliği alan her düğüm, yönlendirme tablosunu günceller ve değişikliği kendi komşularına bildirir. Bu şekilde bütün ağ bu değişiklikten haberdar olur [5, 10].

Çizelge 3.1. 1 numaralı düğümün yönlendirme

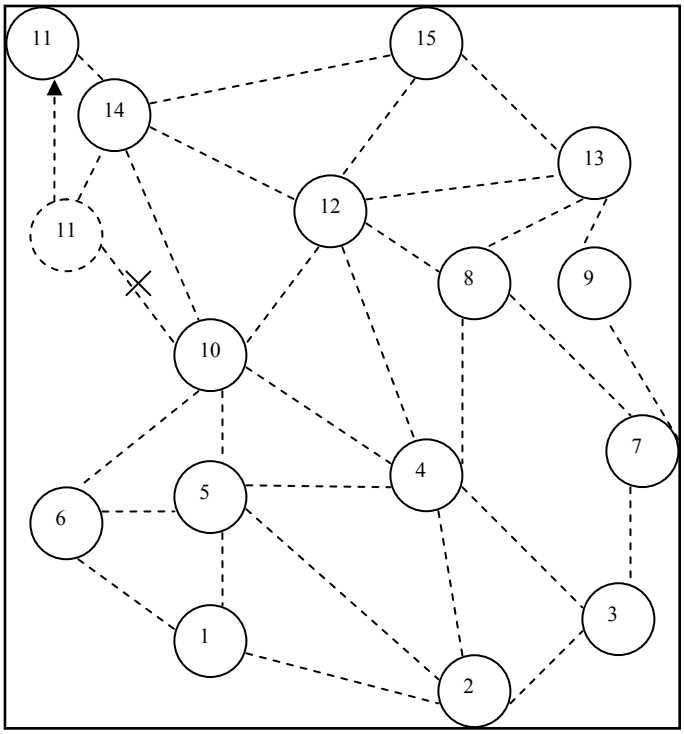
Hedef	**Sonraki Düğüm**	**Uzaklık**	**Sıra Numarası**
2	2	1	22
3	2	2	26
4	5	2	32
5	5	1	134
6	6	1	144
7	2	3	162
8	5	3	170
9	2	4	186
10	6	2	142
11	6	3	176
12	5	3	190
13	5	4	198
14	6	3	214
15	5	4	256



Şekil 3.1. Örnek ağ yapısı

Çizelge 3.2. Güncelleme sonrası 1numaralı düğümün yönlendirme

Hedef	**Sonraki Düğüm**	**Uzaklık**	**Sıra Numarası**
2	2	1	22
3	2	2	26
4	5	2	32
5	5	1	134
6	6	1	144
7	2	3	162
8	5	3	170
9	2	4	186
10	6	2	142
11	**5**	**4**	**180**
12	5	3	190
13	5	4	198
14	6	3	214
15	5	4	256

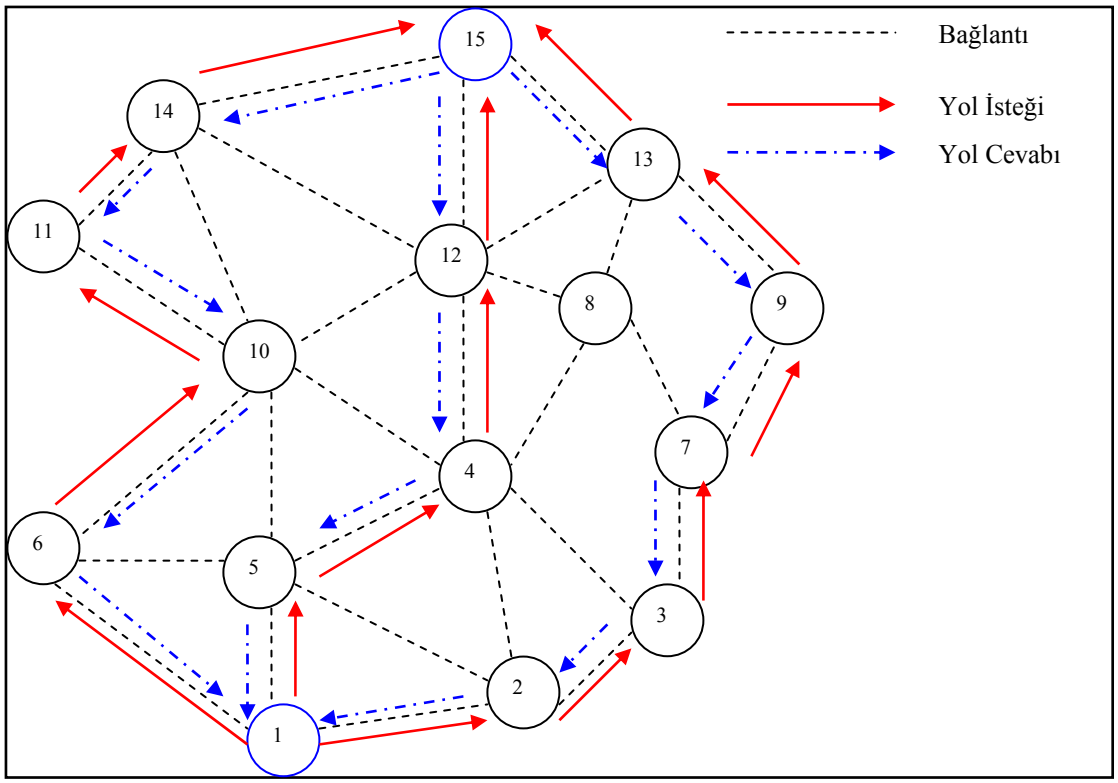


Şekil 3.2. Örnek ağ yapısı

3.5.2. Dinamik kaynak yönlendirme protokolü

Dinamik Kaynak Yönlendirme Protokolü (Dynamic source routing protocol – DSR), tablo kullanan protokollerde olan tablo güncellemelerinin ağ üzerinde taşınması ile meydana gelen yükün azaltılmasını sağlar. Bu protokolün en büyük farkı periyodik "merhaba" mesajlarının olmamasıdır. Bu ve diğer talep üzerine yönlendirme protokollerinin en büyük özelliği yol kurulması aşamasında yönlendirme isteği mesajının ağ üzerinden sel şeklinde gönderilmesidir. Hedef düğüm bir yol isteği paketi aldığı zaman buna yol cevabı paketi ile cevap verir. Kaynak düğüm tarafından yollanan yol isteği paketinin bir sıra numarası vardır. Bu numara sayesinde ara düğümler aynı paketi birden fazla yollamazlar. Ara düğümlere gelen paket eğer daha önceden yayın şeklinde gönderilmişse yeni gelen paket iptal edilir. Sıra numarası koyma mantığı, döngüleri engellemesi açısından önemlidir. Her bir yol isteği paketi gittiği yoldaki

düğümlerin kayıtlarını kendi üzerinde tutar. Bir yol bir defa kurulduktan sonra bu yol tampon'da tutulur. Başka bir yol isteği paketi gönderilirse, bu paketi alan aradaki düğüm kendi tamponunda bulundurduğu yolu kaynağa gönderir ve daha hızlı bir şekilde yol kurulmuş olur [10, 19].

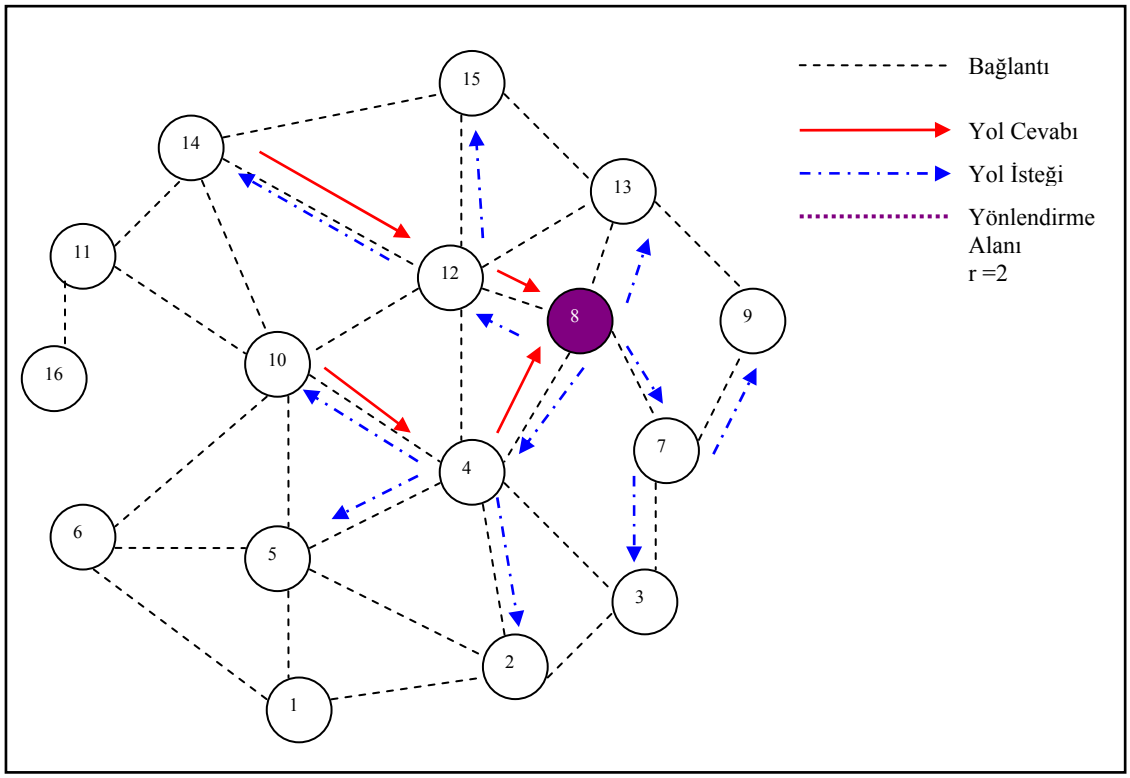


Şekil 3.3. Dinamik kaynak yönlendirme protokolü'nün çalışması

3.5.3. Alan yönlendirme protokolü

Alan Yönlendirme Protokolü (Zone routing protocol – ZRP), proactive ve reactive algoritmaların etkili taraflarını bir araya getirmiş bir algoritmadır. Bu algoritmadaki temel kavram, her bir düğüm için r-atlama sınırlı bir alan içerisinde proactive yönlendirme algoritması uygulamak, alan'lar arasında ise reactive bir algoritma uygulamaktır. Bir düğüm başka bir düğüm'e bir veri göndereceği zaman ilk olarak o

düğüm'ün kendi alanında olup olmadığını kontrol eder. Hedef düğüm kendi alanında ise paketi doğrudan yollar. Hedef düğüm kendi alanında değilse, kendi alanında olan düğüm'lere yayın olarak yol isteği paketi yollar. Paketi alan düğüm'lerden herhangi birinin alanı içersinde bu düğüm varsa, kaynak düğüme yol cevabı paketi gönderilir. Bu yol cevabı paketi içerisinde hedef düğüme giden yol bulunur. Yol cevabı paketini alan kaynak düğüm, verileri yollamaya başlar. Paketi alan düğüm'lerden hiçbirinin alanı içersinde bu düğüm yoksa bu düğümler de yayın olarak kendi alanlarındaki komşularına yol isteği paketi yollarlar. Bu işlem, hedef düğüm bulunana kadar devam eder [9, 10].



Şekil 3.4. Alan yönlendirme protokolünde 8 ve 16 düğümleri arasında yol bulma

3.5.4. Tercih edilen bağlantı tabanlı yönlendirme protokolü

Tercih Edilen Bağlantı Tabanlı Yönlendirme Protokolü'nde (Preferred link-based routing protocols – PLBR), alınan yol isteği paketlerinden yalnızca sağlam bir bağlantıdan gelenin dikkate alınması esastır. Kablolu ağlar da buna benzer yöntemler kullanırlar. Kablolu ağlarda en az bekleme süresi, en az maliyet gibi çeşitli kriterler göz önünde bulundurularak, mevcut yollardan bir tanesi seçilir. Tasarsız ağlarda bir tane yol seçmek uygun olmayabilir. Çünkü ağ topolojisi ve bağlantı kalitesi sürekli değişmektedir. Sisodia ve arkadaşları, komşu düğümler arasından bazılarını seçmek için iki tane algoritma geliştirmişlerdir. Seçilen bu komşuların bulundukları listeye Preffered List - PL adı verilir. Birinci yol, en fazla komşusu olan düğümleri bu listeye almaktır. Böylece, an az yayın ile en çok düğüme ulaşmak mümkün olacaktır. İkinci yol ise, yolun sağlamlığına göre düğüm seçmektir. Her bir düğüm, kendi komşularının ve onların komşularının bilgilerinin kayıtlı olduğu bir tablo tutar (Neighbor's Neighbor Table-NNT). Bağlantı kurulma aşamasında, hedef düğüm ile ilgili kayıt, kaynak düğümün NNT'sinde varsa, bu düğüm ile doğrudan haberleşmeye geçilir. Hedef düğümün kaydı NNT'nin içerisinde yoksa kaynak düğüm, içerisinde kaynak düğümünün adresi, hedef düğümün adresi, tekil sıra numarası, geçilen yolların listesi ve PL'nin bulunduğu bir yol isteği paketi yayınlar. Bu paketi alan düğümlerin herhangi bir tanesinin NNT'sinde hedef düğüm varsa, hedef düğüme doğru yol isteği paketini gönderir. NNT'nin içerisinde hedef düğüme giden birden fazla yol varsa bunlardan bir tanesi rasgele seçilir. Eğer yol isteği paketini alan hiçbir düğümün NNT'sinde hedef düğüme ait bilgi yoksa yol isteği paketini yönlendirirler [10, 22].

4. GELİŞTİRİLEN YÖNLENDİRME PROTOKOLÜ İLE TIKANIKLIK DENETİMİNİN GERÇEKLEŞTİRİLMESİ

Bu bölümde, önerilen tıkanıklık belirleme yöntemi detayları ile anlatılmıştır. Yöntem temel olarak 3 kısımdan oluşmaktadır: aktarım yolu üzerindeki her düğümün arayüzünde oluşan kuyruğun uzunluğunun tıkanıklık oluşturup oluşturmayacağı açısından değerlendirilmesi, hedef düğüme giden ve kuyruk uzunluğu daha az olan bir başka düğümün belirlenmesi ve yük dağılımı yapılması. Bu işlemler, yalnızca kaynakta veya hedefte yapılmamakta, aktarım yolu üzerinde bulunan tüm düğümlerde yapılmaktadır.

4.1. Tıkanıklığa Karar Verilmesi ve Tıkanıklığın Önlenmesi

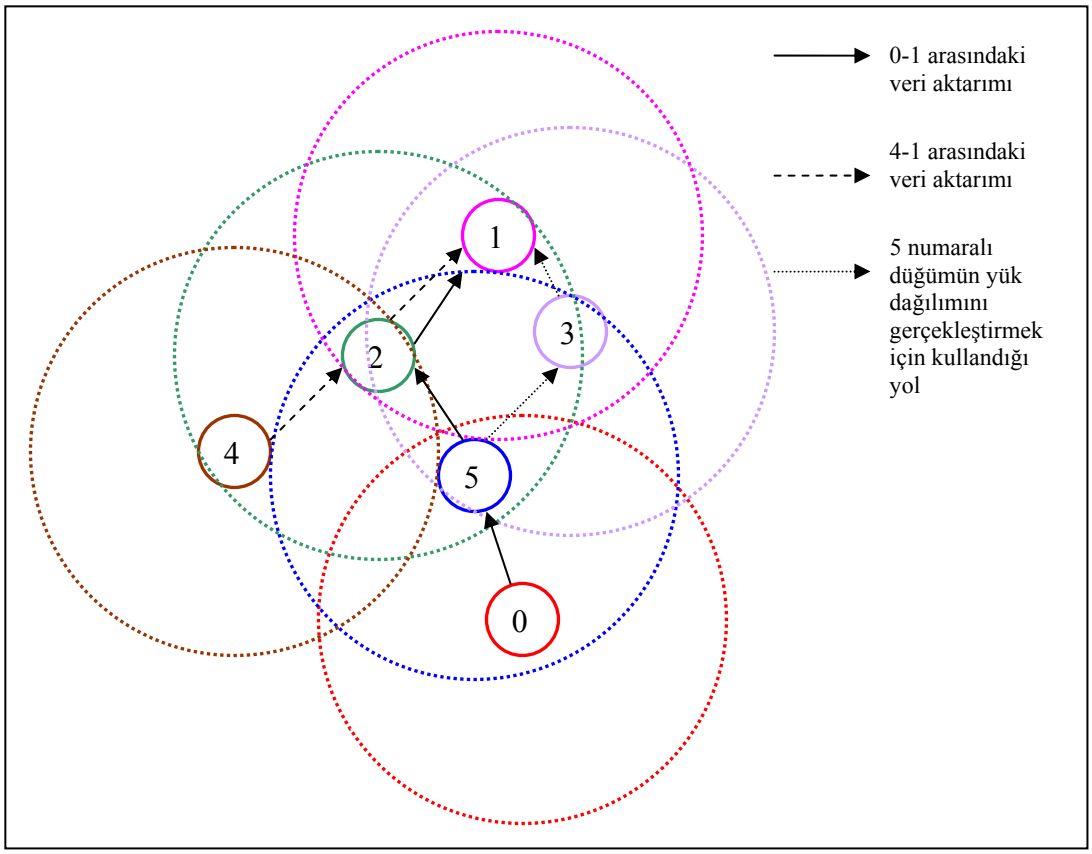
Tıkanıklığın tarifi gereği, kuyruk uzunluğunun sürekli artması ve kuyruk sınır değerine yaklaşması tıkanıklığın meydana geleceğinin habercisidir. Hareketli kablosuz ağlarda düğümlerin hareketleri, veri alma-gönderme talepleri, düğümler arasındaki bağlantıların durumlarının tahmin edilmesi son derece zor olduğu için tıkanıklığın nerede meydana geleceğini kestirmek de mümkün olmamaktadır. Bu yüzden, kaynakta veya hedefte yapılacak tıkanıklık önleme ve giderme yöntemleri en iyi sonucu sağlayamamaktadır. Tıkanıklığı önlemekte kullanılan çok yollu yük dağılımı yöntemi, kaynak ve hedef düğüm için fazladan hesap yükü getirmekte, düğümlerin hareketli oluşu ve topolojinin sık sık değişiyor olması ilgili düğümlerin yapacakları hesapları daha da zorlaştırmaktadır. Önerilen yöntemde her bir düğüm kendi arayüzlerindeki kuyruk uzunluğuna göre tıkanıklık olup olmayacağına karar verip buna göre davranmaktadır. Kuyruğun anlık uzunluğu (KAU), Kuyruk maksimum uzunluğu'nun (KMU) %80'ine kadar ulaşmışsa tıkanıklık meydana gelecek demektir ve düğüm bu durumda hedefe giden başka bir yol arar veya var olan yollar arasından bir tanesini seçer.

```
if   ( KAU <   (KMU*8/10) )
     {
           Tıkanıklık yok
     }
else      {
           Tıkanıklık meydana gelecek
          }
```

Yol seçme kıstası, ilgili yola bakan arayüzdeki kuyruk uzunluğudur. Bu arayüz tıkanmamışsa ve tıkanmaya yaklaşmamışsa veri paketlerinin bir kısmı yeni bulunan yola bakan arayüz üzerinden yollanmaya başlanır. İlk belirlenen yola bakan arayüzdeki kuyruk uzunluğu eşik değerin altına düştüğü zaman bu arayüz üzerinden paket gönderilmeye başlanır. Aksi durumda, gönderilmesi gereken her paket, kuyruk uzunluğu eşik değerin altında olan arayüz üzerinden gönderilir. Aşağıdaki kodda yük dağılımı işlemi gösterilmiştir. K, birinci arayüzün kuyruk uzunluğunu, E_D kuyruk uzunluğu eşik değerini ifade etmektedir.

```
While (Tüm paketler bitene kadar)
     {    if (K1<E_D)
               { Paketi ilk arayüzden yolla}
          else {
                  yeni_yol ara;
           if  (kuyruk uzunluğu eşik değerden az olan bir
yol varsa)
                      Paketi ikinci arayüzden yolla;
                 else Paketi ilk arayüzden yolla;
                  }
     }
```

Şekil 4.1'de örnek bir topoloji üzerinde, önerilen tıkanıklık önleme yönteminin nasıl çalıştığı görülmektedir. 0 numaralı düğüm ile 1 numaralı düğüm veri aktarımı yaparken, 4 numaralı düğüm 1 numaralı düğüme veri yollamaya başlamaktadır. 5 numaralı düğümün 2 numaralı düğüme bakan arayüzündeki kuyruk uzunluğu, eşik değeri aştığı için 5 numaralı düğüm hedefe giden ikinci bir yol belirler ve buradan aktarıma devam eder.



Şekil 4.1. Örnek topoloji

4.2. Benzetimin Gerçekleştirilmesi

4.2.1. Benzetim ortamı

Yeni tıkanıklık önleme yöntemini test etmek ve performansını diğer yöntemlerle karşılaştırmak için ns-2 2.29 sürümü kullanılmıştır. Benzetim aracı olarak ns-2'nin seçilmesinin nedeni, ortamın esnekliği, literatürdeki çoğu çalışmanın bu benzetim aracı ile yapılmış olmasıdır. Ayrıca, AODV, DSR, TORA gibi protokollerin ns-2 içerisinde tanımlanmış olarak gelmeleri test sürecinin kısalması açısından önemlidir.

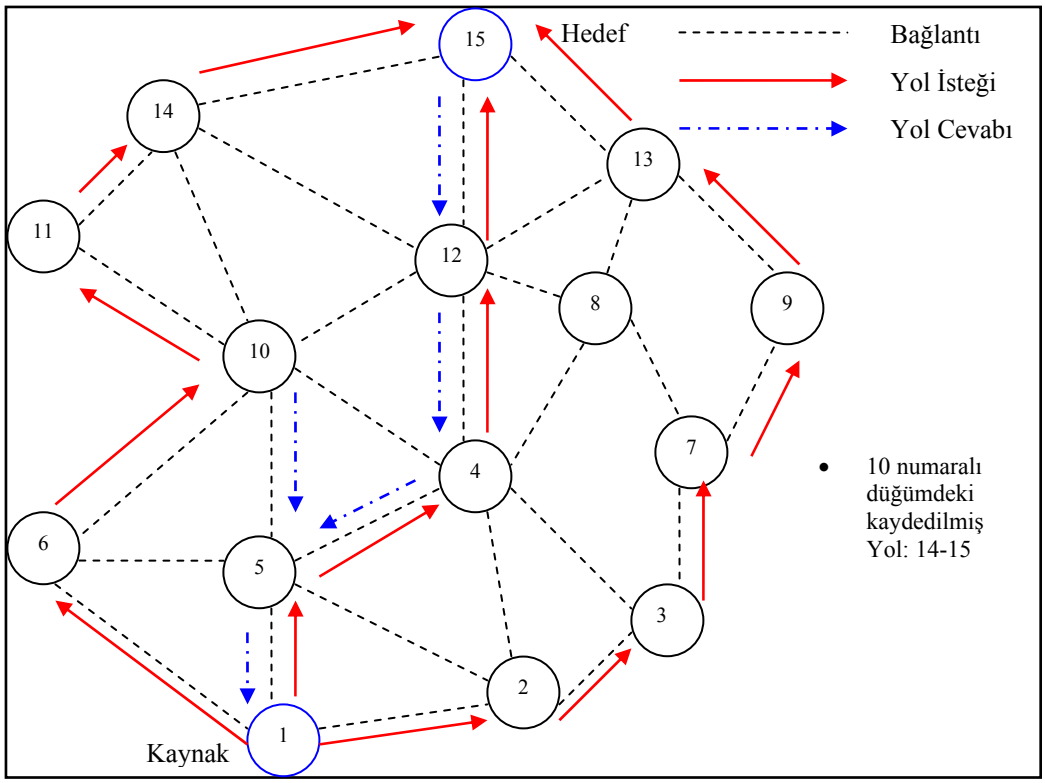
4.2.2. ns-2 ağ benzetim aracı

ns-2, Berkeley Üniversitesi tarafından geliştirilen bir kesikli olay benzetim aracıdır. Başka birçok tipteki ağ yapısının yanında, hem kablosuz ağlar hem de kablolu ağlar için destek vermektedir. Scripting dili olarak OTCL (Object Tool Command Language) kullanan C++ dilinde yazılmış nesne tabanlı bir araçtır. Değişik senaryolar ile deneyleri, var olan protokoller üzerine taşımak için düğümler, bağlantılar, protokoller, trafik vb. gibi çeşitli bileşenlerin tanımlandığı ve çalıştırıldığı bir OTCL scripti yazılabilir. Bununla birlikte var olan bir yönlendirme algoritmasını değiştirmek veya yeni bir yönlendirme algoritması geliştirmek için C++ programlama dili kullanılmak zorundadır. ns-2 her olayı benzetim süresince trace formatında rapor eder. Bu izler daha sonra istenen istatistikleri elde etmek için analiz edilebilirler.

4.2.3. Benzetimde kullanılan AODV yönlendirme Protokolü

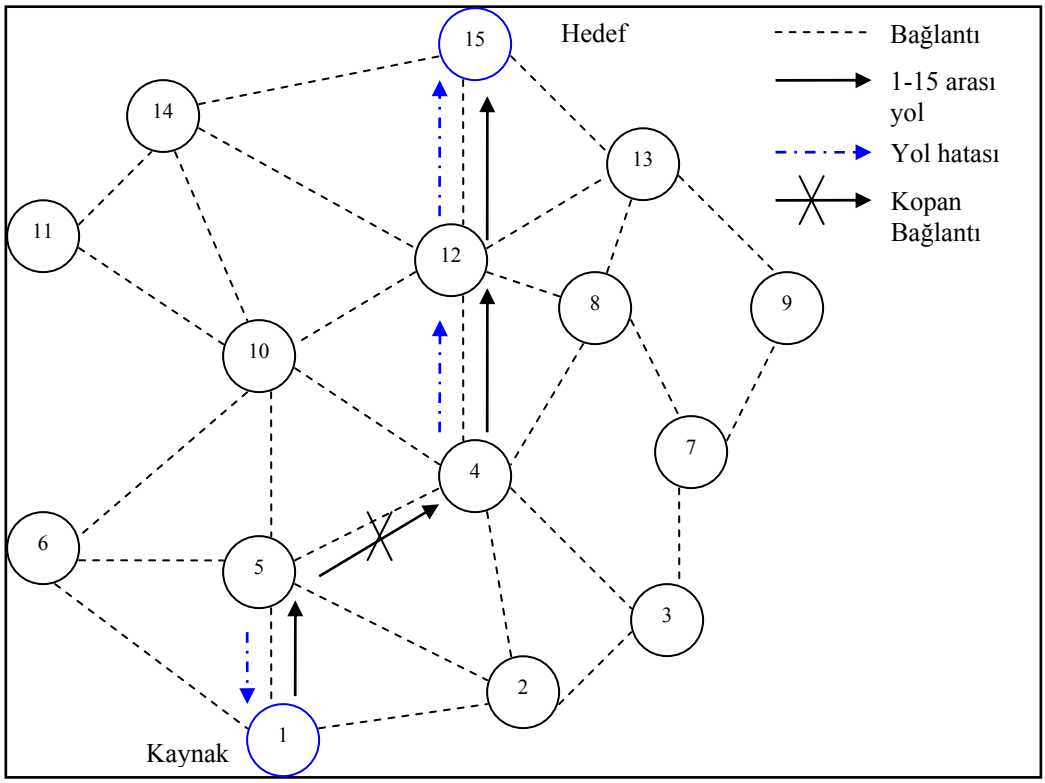
Benzetimde kullanılan Tasarsız Talep Üzerine Uzaklık Vektörü Kullanan Yönlendirme Protokolü (Ad Hoc On Demand Distance Vector – AODV) yol bulmak için isteğe bağlı olarak harekete geçer. Bir düğümü bir başka düğümle iletişime geçmek istediği zaman yol kurma isteği yayınlar. İsteğe karşılık gelen ilk cevabın geldiği yol kullanılarak haberleşmeye başlanır. Elde edilen yolların güncelliklerini belirlemek için her yola bir sıra numarası atanır. Sıra numarası en büyük olan yol en güncel yol anlamına gelmektedir. AODV'de paketlerin takip edeceği yol kaynakta belirlenmez. Bunun yerine kaynak düğüm ve her ara düğümde yalnızca bir sonraki düğümün adresi tutulur. İsteğe bağlı bir yönlendirme protokolünde yol, istenen hedef için kullanılabilir olmadığı zaman kaynak düğüm ağa bir yol istek mesajı yayınlar. Bir yol istek mesajı'nda kaynak tanımlayıcı (Src ID), hedef tanımlayıcı (Dest ID), kaynak sıra numarası (SrcSeqNum), hedef sıra numarası (DestSeqNum), yayın tanımlayıcı (Bcast ID), ve yaşam süresi (TTL) alanları bulunur. DestSeqNum, kaynağın talep kabul ettiği yolun tazeliğini gösterir. Bir ara düğüm bir yol istek mesajı aldığı zaman bu mesajı ya komşularına iletir ya da hedefe

doğru geçerli bir yola sahipse bir yol cevap mesajı hazırlar. Ara düğümdeki yolun geçerliliği, ara düğümdeki sıra numarası ile yol istek mesajı paketinde yer alan DestSeqNum alanının karşılaştırılması ile elde edilir. Bcast ID ve Src ID alanları aynı olan birden fazla yol istek mesajı alınırsa, tekrar eden paketler var demektir ve fazla paketler yok edilir. Hedefe doğru geçerli yollara sahip olan düğümler ya da hedefin kendisi kaynağa yol cevap mesajı yollayabilir. Her ara düğüm kendisine gelen yol istek mesajını yönlendirirken pakete önceki düğümün adresini ve kendi Bcast ID'sini girer. Bu sayede ara düğümde aktif bir yol kaydedilir ve kaynaktan yönlendirmeye gerek kalmaz. Bir ara düğüm yol cevap mesajı aldığı zaman, kendisine mesajı gönderen düğümü, hedefe giden bir sonraki atlama noktası olarak kaydeder. Şekil 4.2'de 1 numaralı düğüm kaynak düğüm olarak 15 numaralı hedef düğüm için bir yol istek mesajını tüm komşularına sel şeklinde gönderir. Yol istek mesajının DestSeqNum değerinin 3 olduğu ve SrcSeqNum değerinin 1 olduğu kabul edilmektedir. 2, 5 ve 6 numaralı düğümler yol istek mesajını aldıkları zaman hedefe doğru olan yollarını kontrol ederler. Hedefe doğru bir yolları yoksa, yol istek mesajını komşularına iletirler. 3 ve 10 numaralı düğümlerin hedefe doğru daha önceden kaydedilmiş yol bilgileri olduğunu ve 3 numaralı düğümdeki DestSeqID değerinin 1, 10 numaralı düğümdeki DestSeqID değerinin 4 olduğunu varsayalım. Kaynaktan belirtilen DestSeqID değeri 3 olduğu için yalnızca 10 numaralı düğümdeki yol kabul edilebilir durumdadır. Hedef düğüme bir başka düğüm üzerinden de ulaşılabiliyorsa (bu örnekte 4 numaralı düğüm) hedef düğüm kaynak düğüme yol cevap mesajı gönderir. Bir yol istek mesajı ile birden fazla yol bulunabilir. Bu yollardan en kısa olan seçilir. AODV kopan bir bağlantıyı yerel olarak onaramaz. Bağlantı koptuğu zaman, kopukluğun olduğu yerdeki düğümler kaynak ve hedef düğümleri sonsuz atlama sayısına sahip yol hata mesajı ile bilgilendirir. Bu mesajı alan kaynak ve hedef düğümler ilgili girişleri silerler. Kaynak düğüm, hedefe giden bir başka yol bulabilmek için yeni bir yol istek mesajı yayınlar.



Şekil 4.2. AODV'de yol kurma

Şekil 4.3'de 4 numaralı düğüm ile 5 numaralı düğümler arasındaki bağlantı koptuğu zaman kaynak ve hedef düğümleri bilgilendirmek adına 4 ve 5 numaralı düğümler tarafından yol hata mesajı oluşturulur ve ilgili düğümlere yollanır. Kaynak düğüm yeni bir BcastID ile önceki DestSeqID ile yol bulma işlemini yeniden başlatır.



Şekil 4.3. AODV'de yol hatası

4.2.4. Benzetim metodolojisi

Düğüm sayısı, veri trafiğinin miktarı, veri paketlerinin boyutu ve bağlantı kapasite sabiti burada belirlenir. Temel hareketlilik desenini değiştirerek farklı test senaryoları elde etmek mümkündür. Böylece, önerilen yöntemin farklı senaryolarda nasıl çalıştığını görmek mümkün olmaktadır.

4.2.5. Performans ölçütleri

Çeşitli hareketli tasarsız ağlar yönlendirme protokollerinin performanslarını değerlendirmek için literatürde yaygın olarak kullanılan üç parametre vardır. Bunlar:

Paket Dağıtım Oranı: Uygulama katmanının CBR kaynakları tarafından oluşturulan paket sayısı ile son hedefteki CBR havuzu (sink) tarafından alınan paket sayısı arasındaki orandır.

Yönlendirme Yükü: Benzetim boyunca iletilen yönlendirme paketlerinin toplam sayısıdır.

Ortalama Gecikme: Kaynak düğümde IP katmanına bir veri paketi verildiği zaman ile hedefin IP katmanına paketin ulaştığı zaman arasındaki ortalama gecikme süresidir.

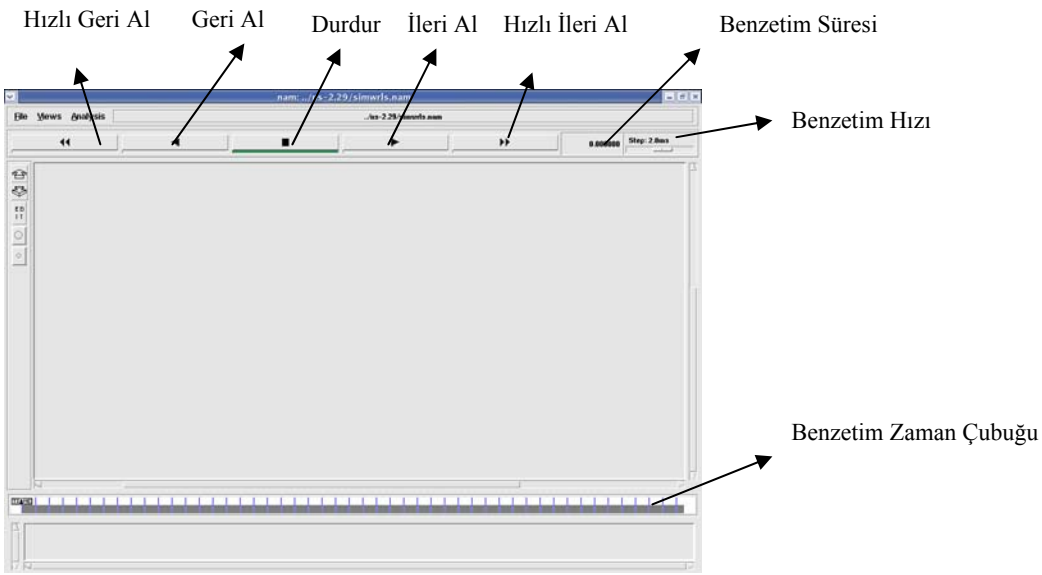
Paket dağıtım oranı önemli bir ölçüttür. Ağ katmanının en üstünde çalışan iletim protokolleri tarafından anlaşılacak kayıp oranını tanımlar. Böylece paket dağıtım oranı ağın destekleyebileceği maksimum işlem hacmini dolaylı olarak gösterir. Benzetim ölçütlerini tanımlarken de bahsettiğimiz gibi paket dağıtım oranı bütün düğümlerde alınan veri paketlerinin toplam sayısının CBR kaynakları tarafından dışarı gönderilen veri paketlerinin toplam sayısına bölümüyle hesaplanır. Paket dağıtım oranı bir ad hoc yönlendirme protokolünün performans değerlendirmesi için önemli bir ölçüt oluşturur; çünkü verilen benzer senaryolarda hedeflere başarılı olarak dağıtılan veri paketlerinin sayısı çoğunlukla yol elverişliliğine bağlıdır. Yol elverişliliği ise dolaylı olarak bir mobil senaryosu içinde temel yönlendirme algoritmasının ne kadar etkili olduğuna bağlıdır. Yönlendirme yükü, paketleri yönlendirirken dışarı gönderilen toplam yönlendirme paketlerin sayısı şeklinde hesaplanabilir. Burada yönlendirme taşması için benzetim gidişatı boyunca ağ katmanında gönderilen yönlendirme paketlerinin toplam sayısı temel alınmıştır.

Daha fazla yönlendirme taşması olursa, hedeflerine ulaşmakta olan veri paketlerinin sahip olacağı gecikme miktarı tıkanıklık, çarpışma ve kuyruk gecikmesi olacağından artacaktır. Ama yönlendirme taşması azaltılırsa doğal olarak daha iyi paket dağıtım sürelerine ulaşılacaktır.

Ortalama gecikme bir veri paketinin kaynak düğümden gönderilip hedef düğümde alınmasına kadar geçen süredir. Bu ölçüt sel yönlendirme algoritmasının ne kadar etkili olduğunun bir ölçümüdür, çünkü her şeyden önce gecikme seçilen yolun en iyiliğine, arayüz kuyruklarının yaşadıkları gecikmelerine ve çarpışmalar yüzünden olan fiziksel katmandaki yeniden iletimlerin neden olduğu gecikmeye bağlıdır.

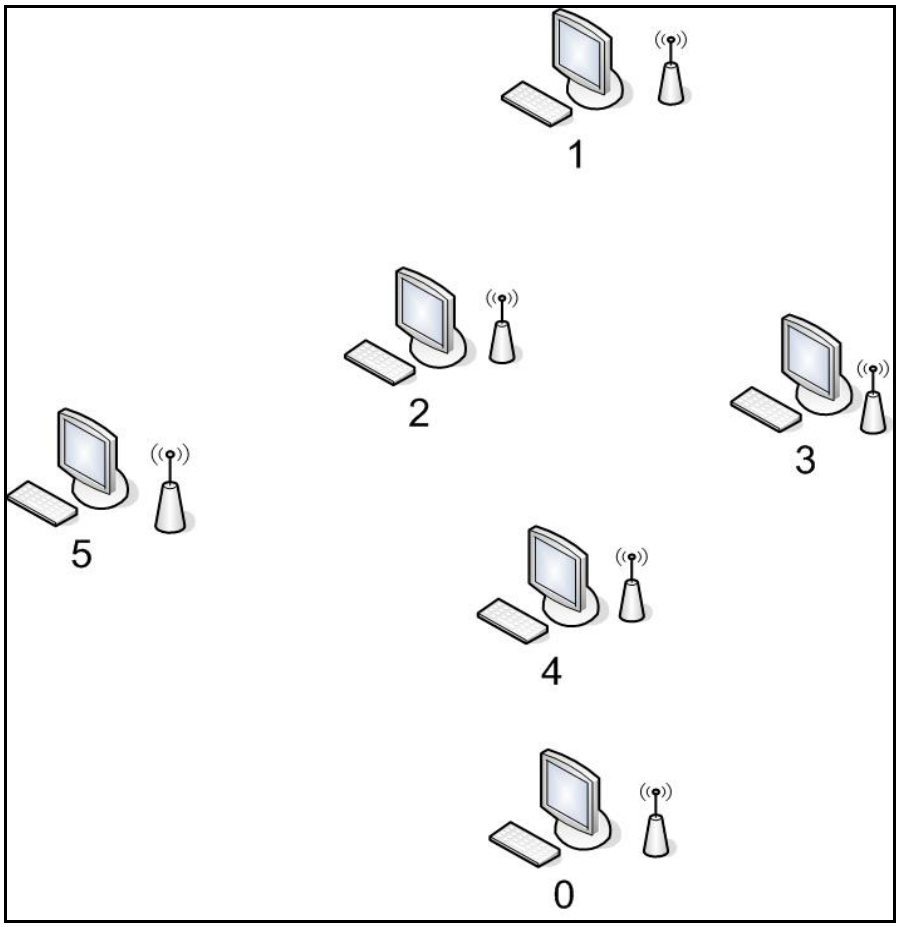
4.2.6. Gerçekleştirilen benzetim

ns-2 ile gerçekleştirilen benzetimler sonucunda meydana gelen ağ yapısını ve trafiği görebilmek için uzantısı "nam" olan bir takip dosyası üretilir. Bu dosyada, benzetimde belirtilen zaman aralıklarında benzetimde gerçekleşen olayların kayıtları tutulur. Örnek olarak; iletilen paketlerin, atılan paketlerin hangi sürelerde gerçekleştikleri, bağlantıların hangi sürelerde koptukları, düğümler hareketli ise düğümlerin hangi hızda nereden nereye doğru hareket ettikleri gibi bilgiler bu dosyada yer alır. Network Animator (nam) olarak isimlendirilen yardımcı bir program bu dosyaları okuyarak görsel hale getirir. nam programının ekran görüntüsü ve programda yer alan düğmelerin ne işe yaradıkları aşağıdaki şekilde gösterilmektedir.



Şekil 4.4. nam programının ekran görüntüsü

Geliştirilen protokolü test etmek için iki adet senaryo üretilmiştir. Bu senaryolarda birbirleri ile eşdeğer bilgisayarların kablosuz ve altyapısız haberleştikleri varsayılmıştır. Benzetim ortamına aktarılan senaryolarda bu bilgisayarlardan düğüm olarak bahsedilmektedir. İlk senaryo biraz daha basit yapıdadır ve daha az düğüm sayısına sahiptir. İlk senaryo ile gerçekleştirilen benzetimde 6 tane eş değer düğüm kullanılmış ve bu düğümlerden dört tanesi haberleştirilmiştir. İlk senaryo için oluşturulan temsili görünüş Şekil 4.5'de, her bir düğümün oluşturulma parametreleri gösterilmektedir.

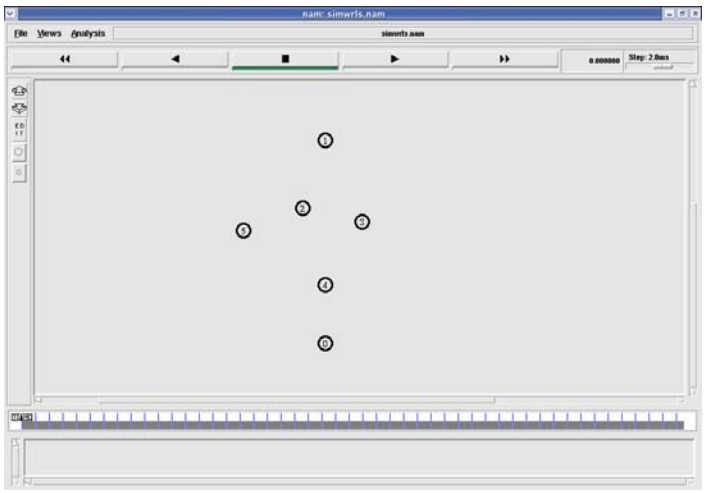


Şekil 4.5. İlk senaryonun temsili görünümü

```
set val(chan)           Channel/WirelessChannel  ;# kanal tipi
set val(prop)           Propagation/TwoRayGround ;#radio-yayılımmodeli
set val(netif)          Phy/WirelessPhy          ;# ağ arayüz tipi
set val(mac)            Mac/802_11               ;# MAC tipi
set val(ifq)            Queue/DropTail           ;# arayüz kuyruk tipi
set val(ll)             LL                    ;# veri bağı katmanı tipi
set val(ant)            Antenna/OmniAntenna      ;# anten modeli
set val(ifqlen)         15               ;# arayüz kuyruğu max.uzunluğu
set val(nn)             6                ;# düğümlerin sayısı
set val(rp)             AODV             ;# yönlendirme protokolü
set val(x)              1000             ;# benzetim alanı X boyutu
set val(y)              1000             ;# benzetim alanı Y boyutu
set val(stop)           150              ;# benzetim süresi
```

Yukarıdaki parametrelerden `ifqlen` parametresi, tıkanıklık belirleme esnasından kullandığımız arayüz kuyruk maksimum uzunluğudur. Benzetimin herhangi bir anında bu değer 12'den büyük olduğu zaman (maksimum kuyruk uzunluğunun %80'inden fazla) arayüzün sahibi olan düğüm, hedefe giden bir başka yol bularak buradan iletişime geçer. Bu değer eşik değerden aşağı düştüğü zaman da tekrar önceki yol kullanılmaya başlanır.

Benzetimin başlangıç ekran görüntüsü Şekil 4.6'da gösterilmektedir. Benzetim aracının genel yapısı şekilde açıklanmıştır.



Şekil 4.6. Benzetimde kullanılan topoloji

Benzetimdeki düğümlerin topolojide nerede bulunacakları, benzetim aracına bildirilmelidir. Eğer düğümler hareketli olarak oluşturulacaklarsa, düğümlerin varacakları noktanın koordinatları, harekete başlama zamanları ve hareketleri esnasındaki hızları da benzetim aracına bildirilmelidir. Üç boyutlu topolojiler de tanımlanabilir. Bunun için Z_ parametresi sıfırdan başka bir değere ayarlanabilir. Gerçekleştirilen benzetimde düğümlerin iki boyutta hareket ettikleri kabul edilerek Z_ parametresi sıfır olarak ayarlanmıştır. Şekil 4.6'daki topolojiyi oluşturan OTCL kodu aşağıdaki gibidir.

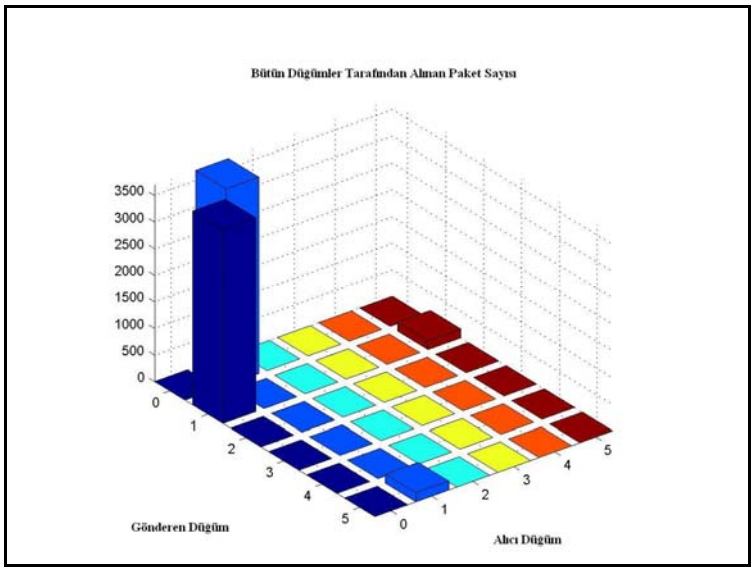
```
$node_(0) set X_ 200.0
$node_(0) set Y_ 50.0
$node_(0) set Z_ 0.0
$node_(1) set X_ 200.0
$node_(1) set Y_ 500.0
$node_(1) set Z_ 0.0
$node_(2) set X_ 150.0
$node_(2) set Y_ 350.0
$node_(2) set Z_ 0.0
$node_(3) set X_ 280.0
$node_(3) set Y_ 320.0
$node_(3) set Z_ 0.0
$node_(4) set X_ 200.0
$node_(4) set Y_ 180.0
$node_(4) set Z_ 0.0
$node_(5) set X_ 100.0
$node_(5) set Y_ 50.0
$node_(5) set Z_ 0.0
```

Benzetimde ulaştırma katmanında TCP kullanılmıştır. TCP'de tıkanıklık önleme ve giderme için kullanılan yöntemler olmasından dolayı bu protokol tercih edilmiştir. TCP kullanılmasının bir diğer nedeni de karşılaştırma kıstaslarından biri olan tıkanıklık penceresi boyutunun yalnızca TCP bağlantılarında yer almasıdır. Veri trafiğini yaratmak için uygulama katmanında FTP (File Transfer Protocol) kullanılmıştır. İlk senaryo için; benzetimin 10. saniyesinde 0 numaralı düğüm ile 1 numaralı düğüm arasında kurulan TCP bağlantısı üzerinden FTP trafiği başlamaktadır. İlk TCP bağlantısında veri aktarımı için belirlenen yol "*0-4-2-1*" şeklindedir. Benzetimin 15. saniyesinde 5 numaralı düğüm ile 1 numaralı düğüm arasında kurulan TCP bağlantısı üzerinden FTP trafiği

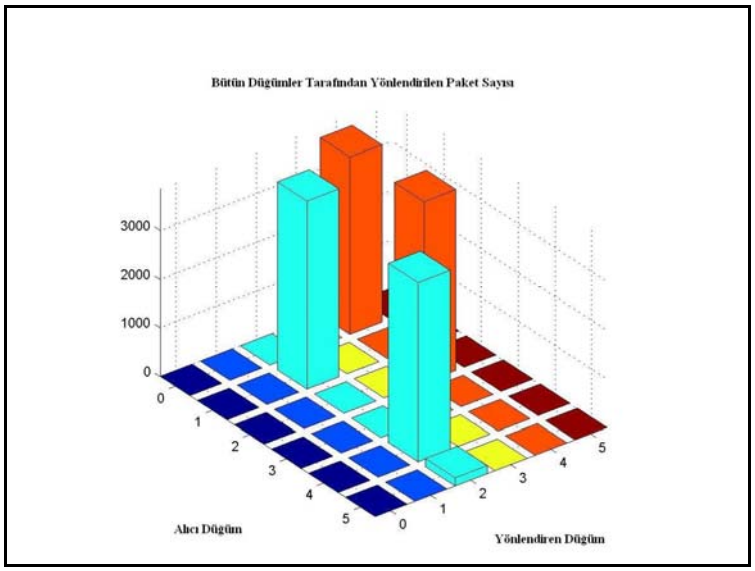
başlamaktadır. İkinci TCP bağlantısında veri aktarımı için belirlenen yol "*5-2-1*" şeklindedir. Bahsedilen veri trafiğini oluşturan OTCL kodları aşağıda verilmektedir.

```
set tcp(0) [new Agent/TCP/Newreno]
set sink(0) [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp(0)
$ns attach-agent $node_(1) $sink(0)
$ns connect $tcp(0) $sink(0)
set ftp [new Application/FTP]
$ftp attach-agent $tcp(0)
$ns at 10.0 "$ftp start"
set tcp(1) [new Agent/TCP/Newreno]
set sink [new Agent/TCPSink]
$ns attach-agent $node_(5) $tcp(1)
$ns attach-agent $node_(1) $sink(1)
$ns connect $tcp(1) $sink(1)
set ftp [new Application/FTP]
$ftp attach-agent $tcp(1)
$ns at 15.0 "$ftp start"
```

İlk senaryoda ağda alınan tüm paket sayısının düğümlere göre grafiği Şekil 4.7'de, ağdaki düğümler tarafından yönlendirilen tüm paket sayısının grafiği Şekil 4.8'de gösterilmektedir.

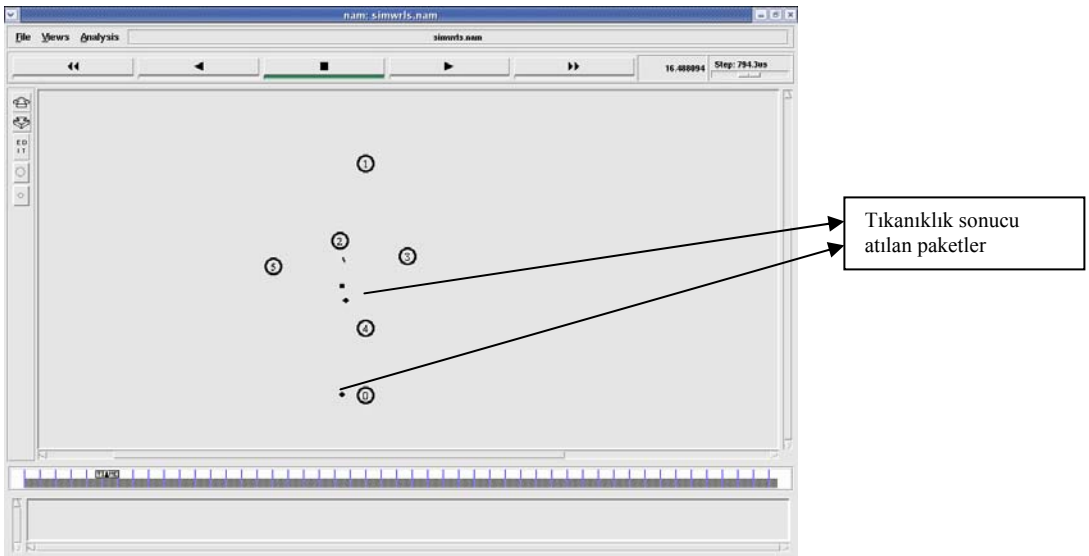


Şekil 4.7. İlk senaryoda düğümlerin aldıkları paket sayıları dağılımı



Şekil 4.8. İlk senaryoda ağda yönlendirilen paket sayısının düğümlere göre dağılımı

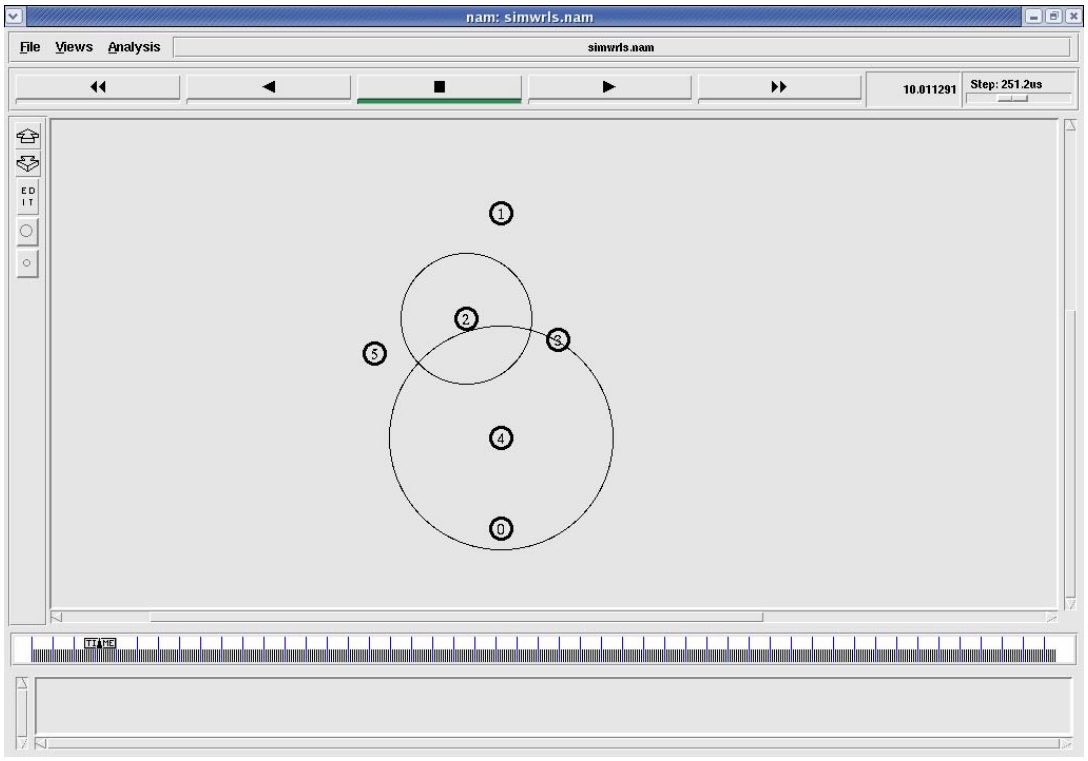
İlk olarak, AODV protokolü bahsedilen senaryo ile denenerek sonuçlar elde edilmiştir. Benzetimde oluşturulan ikinci trafiğin başlaması ile birlikte 2 numaralı düğümün kuyruk uzunluğu artmış ve bir süre sonra tıkanıklık meydana gelmiştir. Tıkanıklık sonucu meydana gelen paket atılma durumu Şekil 4.9'da gösterilmektedir.



Şekil 4.9. Tıkanıklık sonucu paketlerin atılması

Daha sonra, geliştirilen protokol kullanılarak benzetim aynı şartlarda tekrar gerçekleştirilmiştir. Gerçekleştirilen benzetimde düğümler, gönderdikleri yol istek mesajları ile hedefe gidecek yolu temin etmektedirler. Bunun için, kaynak düğümler yol istek paketlerini sel biçiminde ağa yollarlar. En kısa sürede yol cevap paketi gelen yol, veri aktarımında kullanılacak yol olarak seçilir.

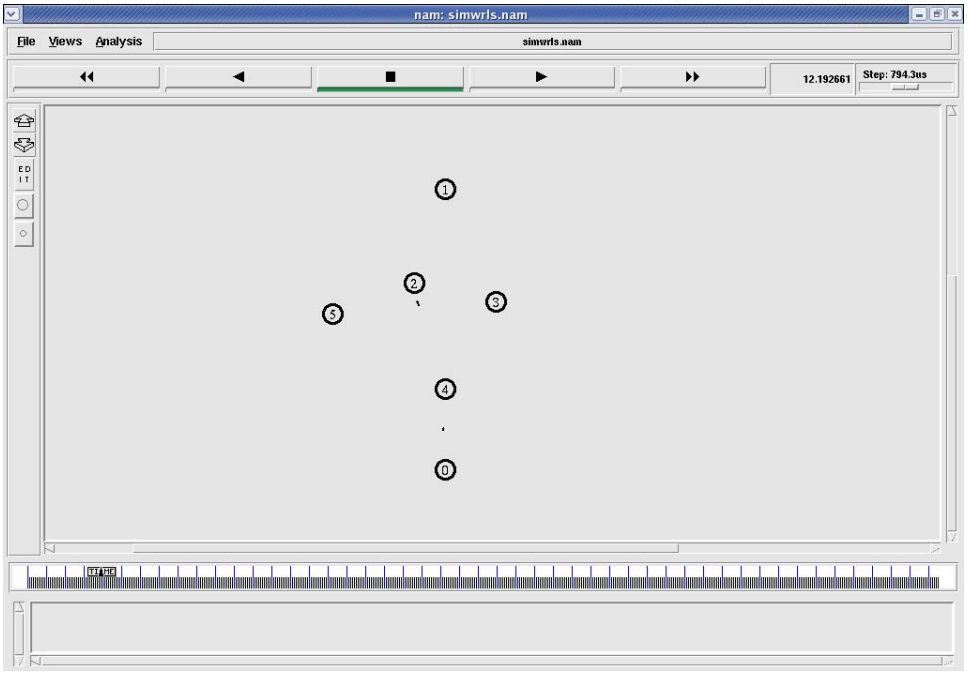
Benzetimin yol belirleme aşamasındaki ekran görüntüsü Şekil 4.10'deki gibidir.



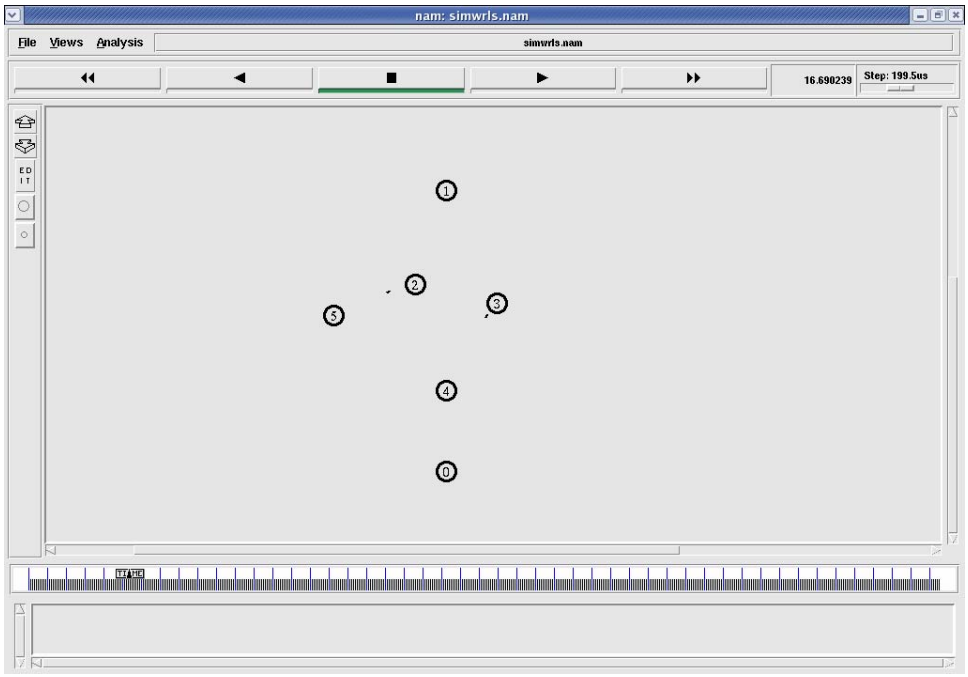
Şekil 4.10. Yol kurulma aşaması

Yol kurulduktan sonra ilgili düğümler veri aktarımına başlamaktadır. Benzetimin 15. saniyesinde 5 numaralı düğüm yol kurma talebinde bulunur ve bulduğu yol üzerinden, yani 2 numaralı düğüm üzerinden veri aktarımına başlar. Bu saniyeden sonra 4 numaralı düğümün 2 numaralı düğüme bakan arayüzündeki kuyruk uzunluğu hızla artmakta ve belirlenen eşik değerini geçmektedir. Önerilen yöntem gereği 4 numaralı düğüm, hedefe

giden bir başka yol aramakta ve 3 numaralı düğümü bir sonraki düğüm olarak seçip paketleri hedefe göndermeye başlamaktadır. 4 numaralı düğüm ile 2 numaralı düğüm arasındaki kuyruk uzunluğunun değeri eşik değerin altına düştüğü zaman trafik tekrar bu arayüzden aktarılmaya devam edilmektedir. Kuyruk uzunluğunun eşik değerinin altında olduğu durumdaki ekran görüntüsü Şekil 4.11'de gösterildiği gibidir. Kuyruk uzunluğunun eşik değerini aştığı zaman 4 numaralı düğümün davranışı sonucu elde edilen ekran görüntüsü Şekil 4.12'de gösterildiği gibidir.



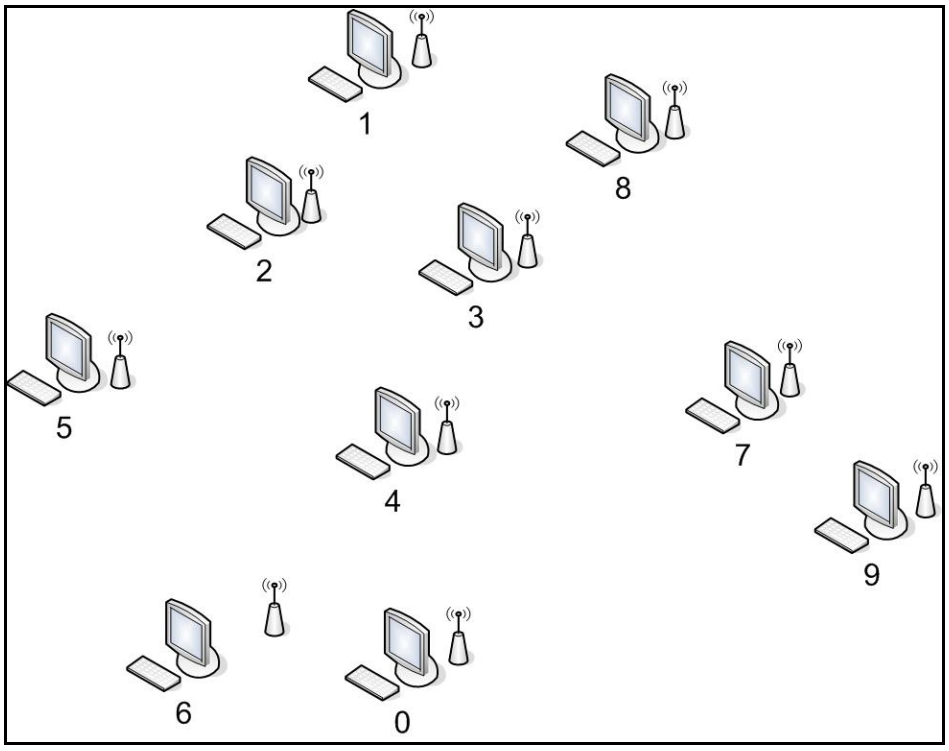
Şekil 4.11. Kuyruk uzunluğu eşik değerinin altındayken ekran görüntüsü



Şekil 4.12. Kuyruk uzunluğu eşik değerine varmışkenki ekran görüntüsü

Şekil 4.12'de de görüldüğü gibi 4 numaralı düğüm tıkanıklık olacağına karar verdiği zaman trafiği 3 numaralı düğümün üzerine dağıtmıştır. Böylece, 2 numaralı düğümün giriş kuyruk uzunluğu daha fazla artıp maksimum değere ulaşmadan ve paket atılmaları meydana gelmeden duruma müdahale edilmiştir.

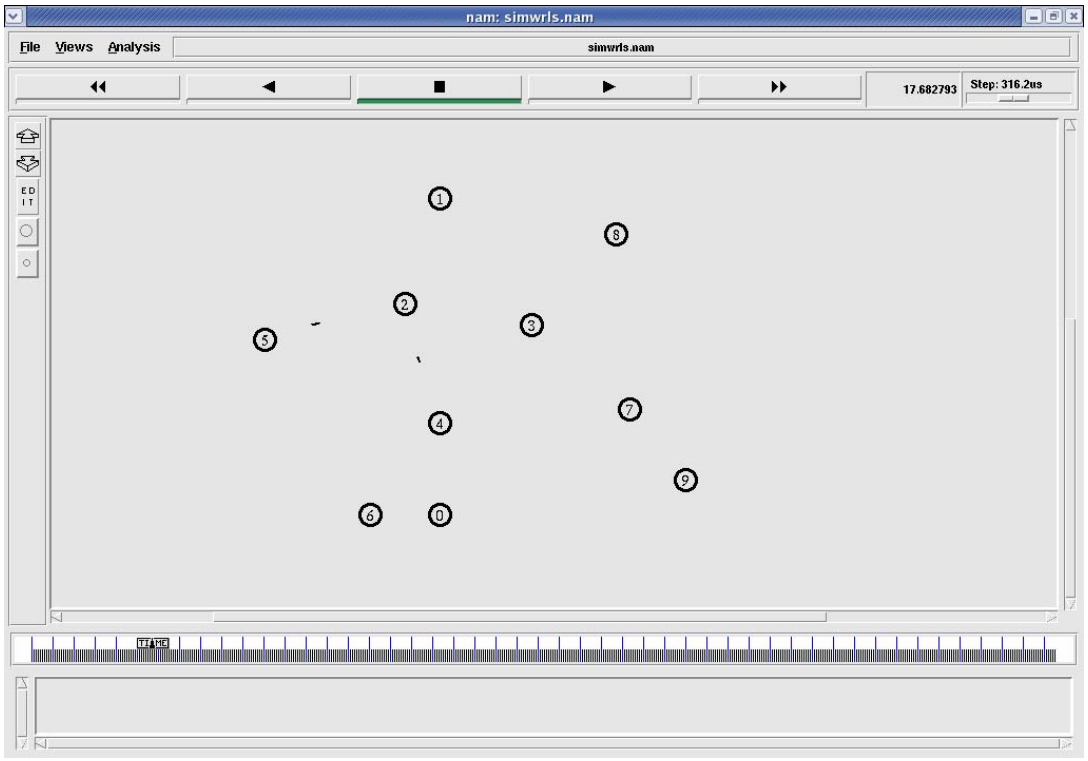
İkinci senaryoda 10 adet düğüm kullanılmıştır. Bu senaryoda daha fazla TCP bağlantısı oluşturularak FTP ile veri aktarımı gerçekleştirilmiştir. Benzetimdeki tüm düğümlerin benzetimin değişik zamanlarında kullanılması ile tıkanıklığa son derece müsait bir ortam yaratılmıştır. Geliştirilen protokolün böyle bir duruma vereceği cevap, ağdaki her derece tıkanıklığa vereceği cevabı tahmin etmek açısından önemlidir. Oluşturulan senaryonun temsili yapısı Şekil 4.13'de gösterilmektedir.



Şekil 4.13. İkinci senaryonun temsili görünümü

Yukarıda temsili görünümü verilen benzetimi meydana getiren başlangıç kodları aşağıda, benzetimin ekran görüntüsü ise Şekil 4.14'de gösterilmektedir.

```
set val(chan)           Channel/WirelessChannel  ;# kanal tipi
set val(prop)           Propagation/TwoRayGround ;#radio-yayılımmodeli
set val(netif)          Phy/WirelessPhy          ;# ağ arayüz tipi
set val(mac)            Mac/802_11               ;# MAC tipi
set val(ifq)            Queue/DropTail           ;# arayüz kuyruk tipi
set val(ll)             LL                    ;# veri bağı katmanı tipi
set val(ant)            Antenna/OmniAntenna      ;# anten modeli
set val(ifqlen)         15              ;# arayüz kuyruğu max.uzunluğu
set val(nn)             10                   ;# düğümlerin sayısı
set val(rp)             AODV            ;# yönlendirme protokolü
set val(x)              1000            ;# benzetim alanı X boyutu
set val(y)              1000            ;# benzetim alanı Y boyutu
set val(stop)           150             ;# benzetim süresi
```

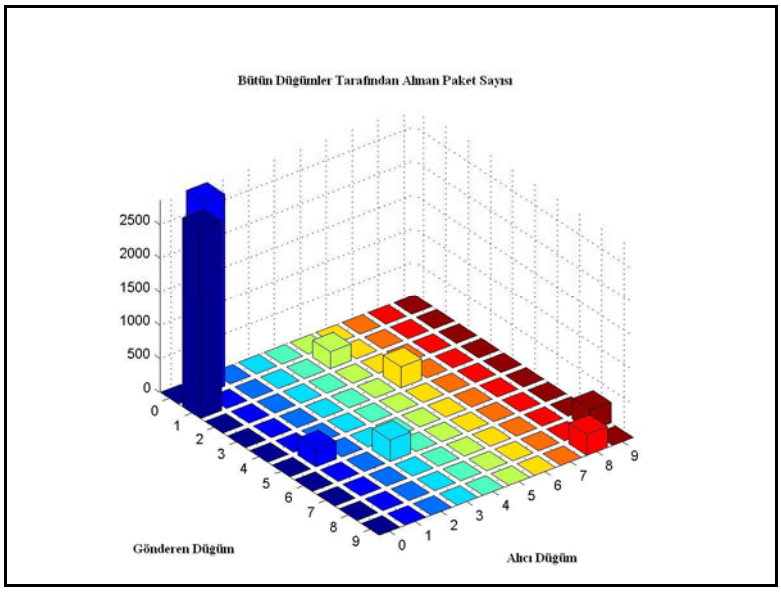


Şekil 4.14. İkinci senaryo için benzetim ekranı

Benzetimin 10. saniyesinde 0 numaralı düğüm 1 numaralı düğüme doğru veri aktarımına başlamaktadır ve bu durum benzetimin 145. saniyesine kadar devam etmektedir. Benzetimin 15. saniyesinde 5 numaralı düğüm de 1 numaralı düğüme doğru veri aktarımına başlamaktadır ve bu aktarım 25. saniyeye kadar devam etmektedir. Benzetimin 20. saniyesinde 6 numaralı düğümden 3 numaralı düğüme doğru veri aktarımı başlamaktadır ve bu aktarım 40. saniyeye kadar devam etmektedir. Son olarak, benzetimin 25. saniyesinde 9 numaralı düğümden 8 numaralı düğüme doğru veri aktarımı başlamakta ve bu aktarım 50. saniyede sona ermektedir. Bu trafikleri meydana getiren OTCL kodu aşağıda gösterilmektedir. Her bir aktarım için başlangıçta oluşturulan yollar sırasıyla: 0-4-2-1, 5-2-1, 6-4-3, 9-7-3-8 şeklindedir.

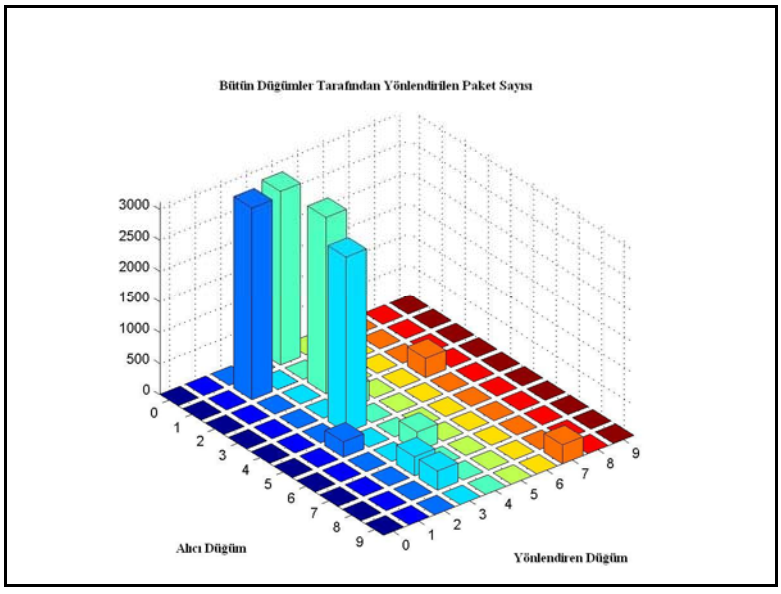
```
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 1
set sink [new Agent/TCPSink]
$ns attach-agent $node_(0) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 10.0 "$ftp start"
$ns at 145.0 "$ftp stop"
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 2
set sink [new Agent/TCPSink]
$ns attach-agent $node_(5) $tcp
$ns attach-agent $node_(1) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 15.0 "$ftp start"
$ns at 25.0 "$ftp stop"
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 3
set sink [new Agent/TCPSink]
$ns attach-agent $node_(6) $tcp
$ns attach-agent $node_(3) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 20.0 "$ftp start"
$ns at 40.0 "$ftp stop"
set tcp [new Agent/TCP/Newreno]
$tcp set class_ 4
set sink [new Agent/TCPSink]
$ns attach-agent $node_(9) $tcp
$ns attach-agent $node_(8) $sink
$ns connect $tcp $sink
set ftp [new Application/FTP]
$ftp attach-agent $tcp
$ns at 25.0 "$ftp start"
$ns at 50.0 "$ftp stop"
```

İkinci senaryoda ağda alınan tüm paket sayısının düğümlere göre grafiği Şekil 4.15'de, ağdaki düğümler tarafından yönlendirilen tüm paket sayısının grafiği Şekil 4.16'da gösterilmektedir.



Şekil 4.15. İkinci senaryoda düğümlerin aldıkları paket sayıları dağılımı

Üstteki şekilde de görüldüğü gibi 0 ve 1 numaralı düğümler en çok paket alan düğümlerdir. 0 numaralı düğüm, 1 numaralı düğümün oluşturduğu veri paketlerini, 1 numaralı düğüm de 0 numaralı düğümün oluşturduğu ACK paketlerini almıştır.



Şekil 4.16. İkinci senaryoda ağda yönlendirilen paket sayısının düğümlere göre dağılımı

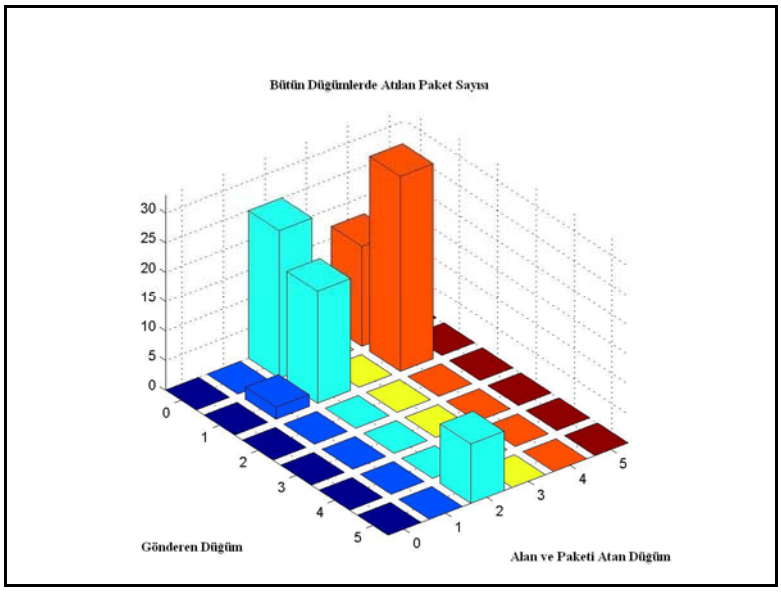
İkinci senaryo ile ilk olarak AODV protokolü test edilmiştir. AODV protokolü ile gerçekleştirilen benzetim sonucunda ağda yüksek oranda paket kaybı yaşandığı görülmüştür. 9 numaralı düğüm ile 8 numaralı düğüm arasında ve 6 numaralı düğüm ile 3 numaralı düğüm arasında oluşan trafik, 0 numaralı düğüm ile 1 numaralı düğüm arasında oluşan trafiğin aktığı düğümlerden geçmekte olduğu için genel olarak 4 ve 2 numaralı düğümlerin davranışları bundan etkilenmektedir. Ayrıca, düğümlerin veri iletmek için yaydıkları radyo sinyalleri de ortamda artmakta ve birbirleri üzerinde bozunuma neden olma ihtimalleri yükselmektedir. Gerçekleştirilen benzetimin detayları Deneysel Sonuçlar bölümünde anlatılacaktır.

Daha sonra, geliştirilen protokol ikinci senaryo ile test edilmiştir. Bu testte, ağda yaşanan paket kayıplarının belirgin bir biçimde azaldığı gözlemlenmiştir. 5-1 düğümleri arasındaki trafik ve 0-1 düğümleri arasındaki trafikten dolayı 2 numaralı düğüm tıkanmaya başlamıştır. Bu durumda 2 numaralı düğümün giriş kuyruk uzunluğu artmıştır. Geliştirilen yöntem gereği 4 numaralı düğüm 1 numaralı düğüme giden yeni bir yol bularak trafiği bu yola paylaştırmıştır. Bu senaryoda 6–3 düğümleri arasında ve 9–8 düğümleri arasında TCP bağlantılarının oluşturulma nedeni, 3 numaralı düğümün her iki bağlantıda da kullanılıyor olması ve bu durumun yük dağılımına etki edip etmediğinin test edilmesidir. Sonuçta bu durumun çok fazla önemli olmadığı görülmüştür. Yapılan benzetimlerle ilgili detaylı sonuçlar Deneysel Sonuçlar başlığı altında anlatılmaktadır.

5. DENEYSEL SONUÇLAR

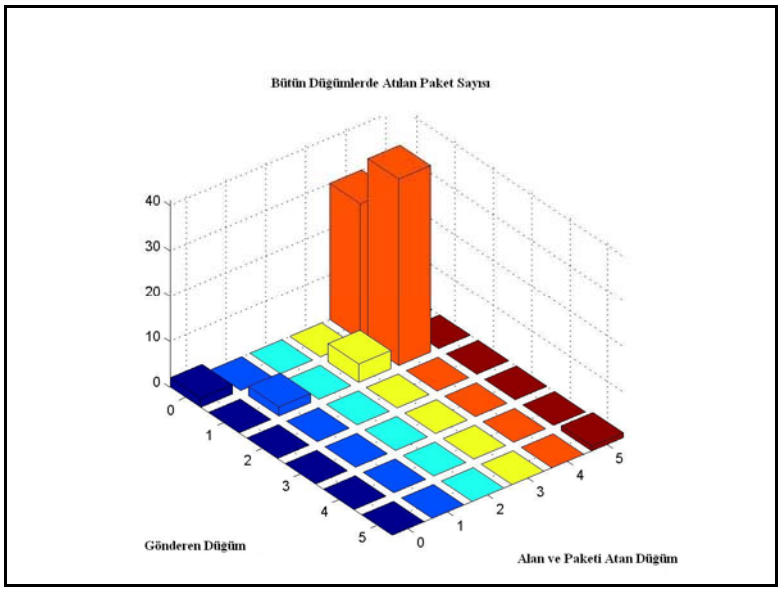
Geliştirilen protokol ile AODV protokolü karşılaştırılarak çeşitli sonuçlar elde edilmiştir. Bir protokolün tıkanıklık durumunda nasıl bir tepki verdiğini görmek için iki tane kıstas göz önüne alınmıştır. Bunlardan birincisi tüm ağda meydana gelen paket kayıpları sayısıdır. Kablolu ağlarda paket kayıplarının çok büyük bir kısmı tıkanıklık durumunda meydana gelir. Kablosuz ağlarda da paket kayıplarının büyük nedeni tıkanıklıktır. Bunun yanı sıra çevresel etkilerden ve aktarım ortamının açık olmasından dolayı paket Kayıpları meydana gelmektedir. İkinci kıstas ise, TCP bağlantılarına ait olan tıkanıklık penceresi boyutudur. Daha önce de anlatıldığı gibi, tıkanıklık meydana gelmediği durumlarda tıkanıklık penceresinin büyüklüğü maksimum değerdedir veya gittikçe bu değere yaklaşmaktadır. Tıkanıklık meydana geldiğinde ise, tıkanıklık penceresinin büyüklüğü yarı yarıya düşürülür veya 1 olarak ayarlanır.

AODV protokolü ve ilk senaryo ile gerçekleştirilen benzetim sonucunda elde edilen atılan paketler grafiği Şekil 5.1'de gösterilmektedir. Özellikle 0 ve 1 numaralı düğümlerin oluşturduğu paketlerin 2 ve 4 numaralı düğümlerde atıldığı görülmektedir. Kaynağı 0 düğümü olan paketler veri paketleri, kaynağı 1 olan paketler de ACK paketleridir. 2 numaralı düğümde atılan paketlerinin büyük kısmının atılma nedeni tıkanıklıktır. 4 numaralı düğümde atılan paketlerin büyük kısmı ACK paketidir. Tıkanıklıktan dolayı geciken ACK paketleri nedeniyle kaynak düğüm yeniden iletim yordamını çalıştırır. Yeniden gönderilen veri paketleri 4 numaralı düğüme geldiği zaman, 4 numaralı düğümde beklemekte olan ACK paketleri ve henüz gönderilmemiş olan veri paketleri atılır. 4 numaralı düğümde meydana gelen paket atılmalarının büyük nedeni budur.



Şekil 5.1. AODV protokolü ile ilk senaryoda atılan paket sayıları

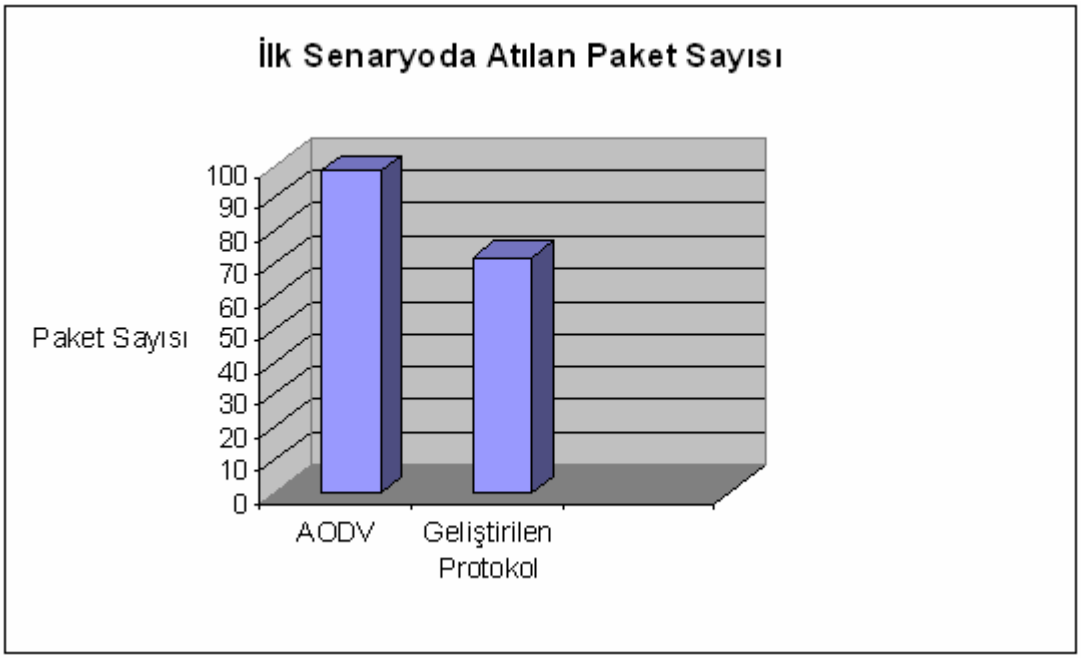
Geliştirilen yönlendirme protokolü ve ilk senaryo ile gerçekleştirilen benzetim sonucunda elde edilen atılan paketler grafiği Şekil 5.2'de gösterilmektedir.



Şekil 5.2. Geliştirilen protokol ile ilk senaryoda atılan paket sayıları

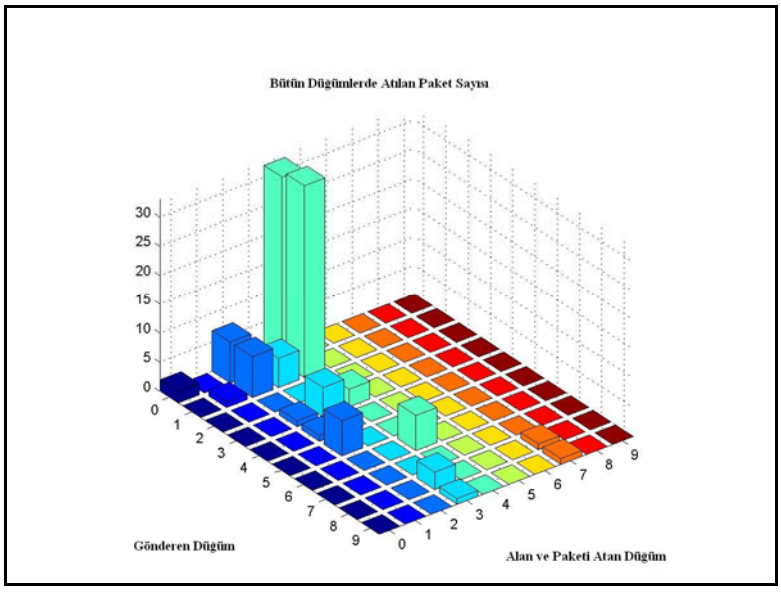
Geliştirilen ihtiyaç anında yük dağılımı yöntemi ile 2 numaralı düğümde hiç paket atılması meydana gelmediği görülmektedir. 3 numaralı düğümde küçük bir ACK paket atılması durumu yaşanmıştır. 4 numaralı düğümde meydana gelen paket atılmaları da AODV protokolüne göre daha düşüktür. AODV protokolü ile yapılan benzetimden elde edilen tıkanıklık penceresi grafiği aşağıdaki şekilde gösterilmektedir. Şekilden de görüleceği gibi yoğun trafiğin başladığı anda tıkanıklık penceresi boyutu artmaktadır ve maksimum değere ulaşmaktadır. Paket atılmaları da burada meydana gelir.

İki protokolün ilk senaryoya göre oluşturulmuş atılan toplam paket sayısı karşılaştırma grafiği Şekil 5.3'de gösterilmektedir. AODV protokolünde toplam olarak 98 paket atılması meydana gelirken, geliştirilen yöntemde 72 paket atılması meydana gelmiştir. Geliştirilen protokoldeki paket atılma oranının AODV protokolündeki paket atılma oranına göre yaklaşık olarak %18 daha az olduğu görülmektedir.



Şekil 5.3. İki protokolün ilk senaryoya göre oluşturulmuş atılan toplam paket sayısı karşılaştırma grafiği

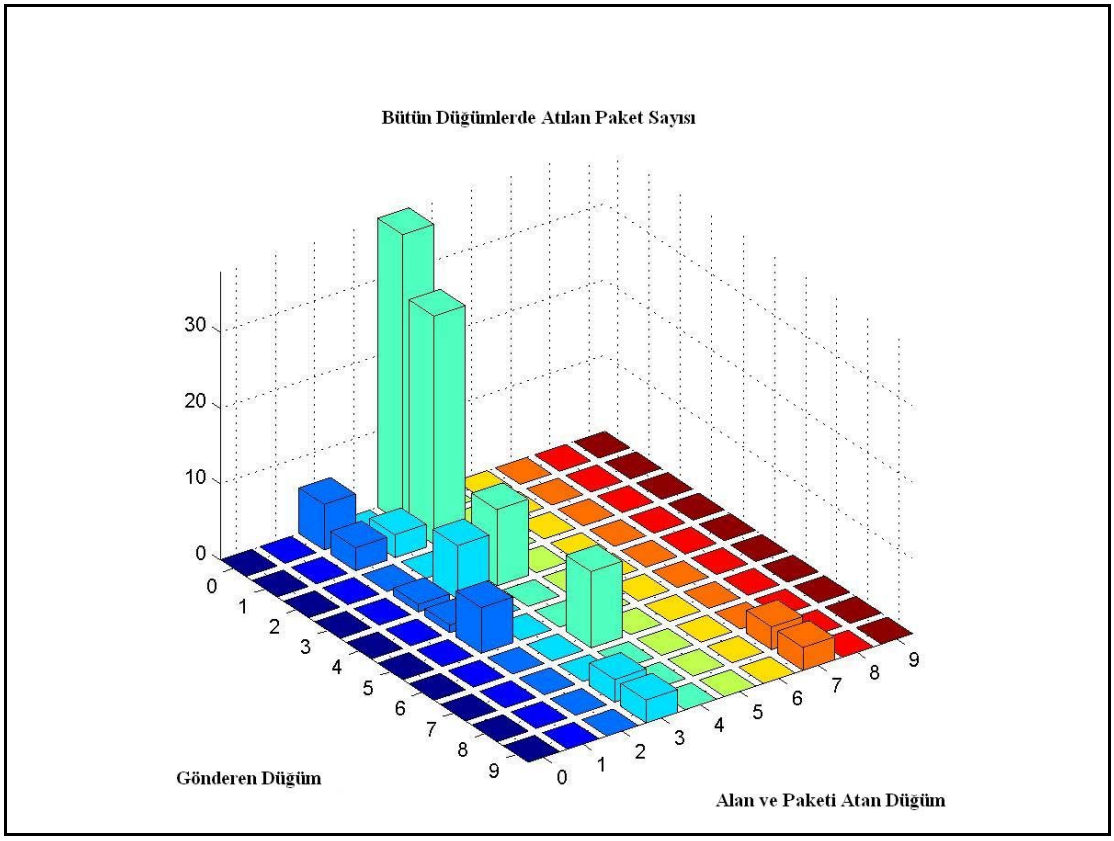
AODV protokolü ve ikinci senaryo ile gerçekleştirilen benzetim sonucunda elde edilen atılan paketler grafiği Şekil 5.4'de gösterilmektedir. İlk senaryoda olduğu gibi özellikle kaynağı 0 ve 1 numaralı düğümler olan paketlerin 2 ve 4 numaralı düğümlerde yüksek oranda atıldığı görülmektedir. Aynı şekilde, kaynağı 5 ve 6 numaralı düğümler olan paketlerin de 2 ve 4 numaralı düğümlerde atıldıkları görülmektedir. İkinci senaryoda daha fazla düğümün aynı anda veri göndermek istemesinden dolayı çakışmalar artmaktadır. Paket atılmalarının bir diğer nedeni de budur.



Şekil 5.4. AODV protokolü kullanılarak ikinci senaryo ile gerçekleştirilen benzetim sonucunda elde edilen atılan paketler grafiği

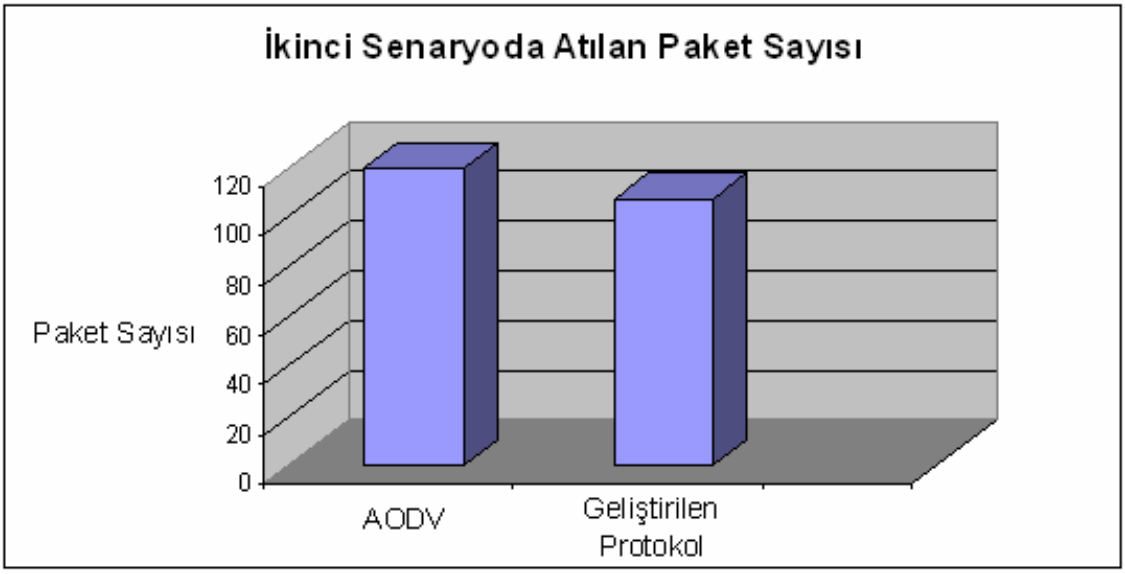
Geliştirilen yönlendirme protokolü ve ilk senaryo ile gerçekleştirilen benzetim sonucunda elde edilen atılan paketler grafiği Şekil 5.5'de gösterilmektedir. AODV protokolünde 2 numaralı düğüm tarafından atılan toplam paket sayısı 26 iken, geliştirilen yöntemde 2 numaralı düğüm tarafından atılan toplam paket sayısı 20 olarak tespit edilmiştir. AODV protokolünde 4 numaralı düğüm tarafından atılan toplam paket sayısı 78, geliştirilen protokolde 4 numaralı düğüm tarafından atılan toplam paket sayısı ise 77 olarak tespit edilmiştir. Geliştirilen protokolün 3 numaralı düğümü daha fazla kullanması ve bu düğüm üzerinden başka trafiklerin de geçiyor olmasına rağmen elde

edilen sonuçlar daha iyidir. Zira, AODV protokolünde 0–1 düğümleri arasındaki bağlantı için 3 numaralı düğüm kullanılmamaktadır. Dolayısı ile 3 numaralı düğümün daha sık sinyal yayması ve komşularının sinyallerini bozması, böylece de paketlerin atılmasına neden olması durumu söz konusu değildir.



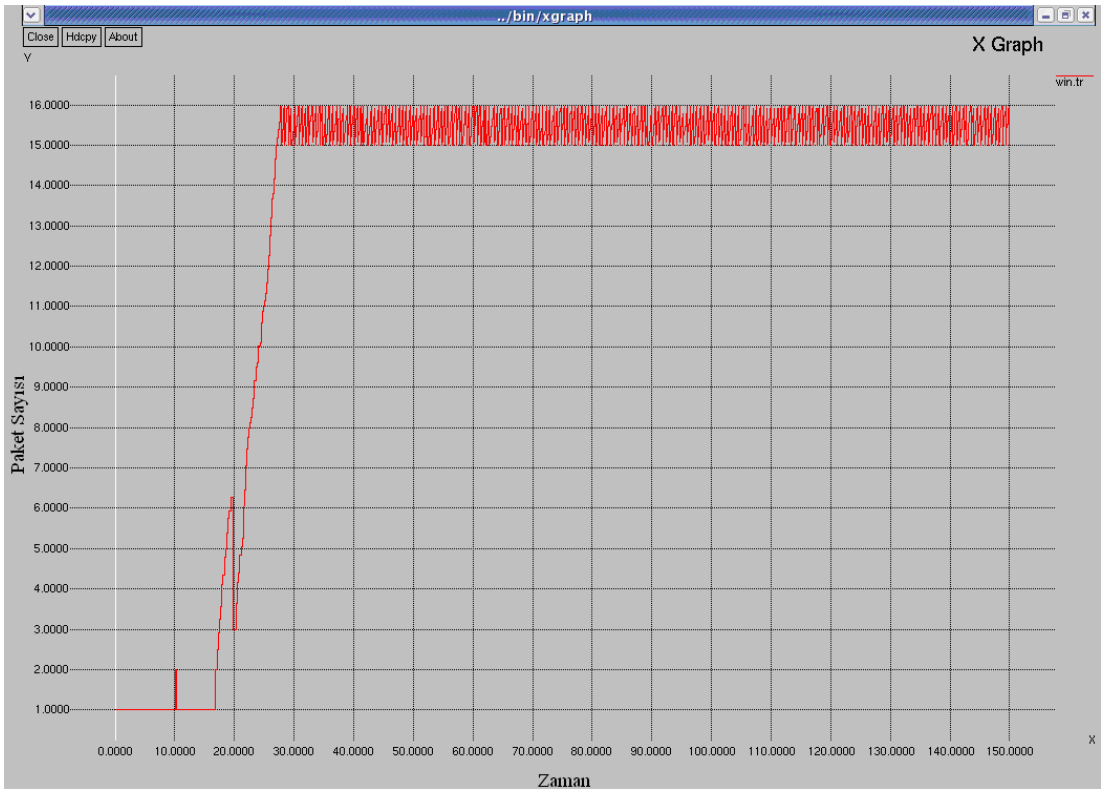
Şekil 5.5. Geliştirilen protokol kullanılarak ikinci senaryo ile gerçekleştirilen benzetim sonucunda elde edilen atılan paketler grafiği

İki protokolün ikinci senaryoya göre oluşturulmuş atılan toplam paket sayısı karşılaştırma grafiği Şekil 5.6'da gösterilmektedir. AODV protokolünde 119 paket atılması meydana gelmişken, geliştirilen protokolde 107 paket atılması meydana gelmiştir. Geliştirilen protokoldeki paket atılma oranının AODV protokolündeki paket atılma oranına göre yaklaşık olarak %10 daha az olduğu görülmektedir.



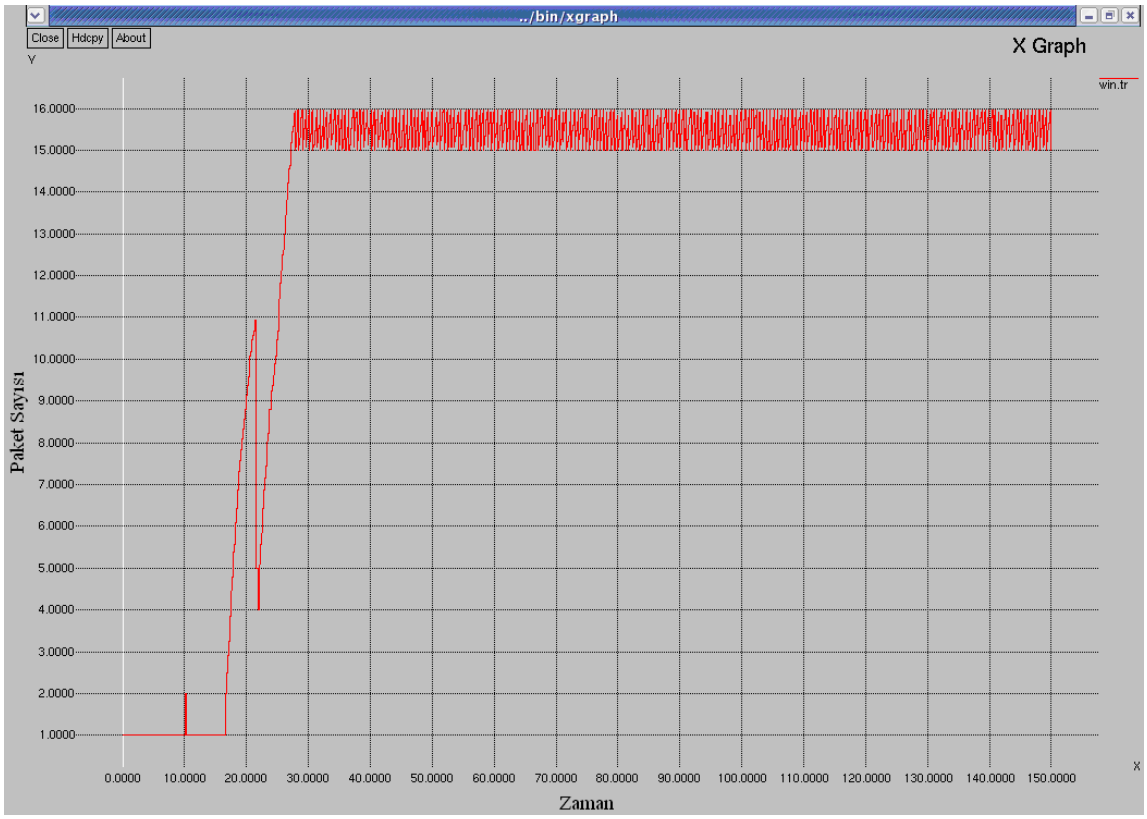
Şekil 5.6. İki protokolün ilk senaryoya göre oluşturulmuş atılan toplam paket sayısı karşılaştırma grafiği

İkinci kıstas olan tıkanıklık penceresi boyutuna göre her iki senaryoda iki protokol karşılaştırılmıştır. AODV protokolü ile ilk senaryoda gerçekleştirilen benzetimde 0-1 düğümleri arasındaki TCP bağlantısının tıkanıklık pencere boyutu Şekil 5.7'de gösterilmektedir. Benzetimin 15. saniyesinde trafiğin başlaması ile birlikte tıkanıklık penceresinin boyutu arttırılmaya başlanmıştır. 20. saniyede tıkanıklık meydana gelmiş ve pencere boyutu yarıya düşürülmüştür. İlerleyen sürede tıkanıklık penceresi arttırılmış ve 5–1 düğümleri arasındaki trafiğin sona ermesi ile tıkanıklık penceresi boyutu en üst değere ulaşmıştır. Elde edilen tıkanıklık penceresi grafiğindeki 25. saniyeden sonraki kararlı durum, yalnızca ilk oluşturulan TCP bağlantısının kalması sonucunda oluşmuştur. Diğer TCP bağlantısı sona erdiği zaman, paket atılmaları meydana gelmemiş ve tıkanıklık penceresi boyutu maksimum değere ulaşarak devam etmiştir. En üstteki salınımların nedeni ise, kuyruk uzunluğunun 15 olmasından kaynaklanan pencere artırımına izin verilmemesidir.



Şekil 5.7. AODV protokolü ile ilk senaryoda gerçekleştirilen benzetimde 0-1 düğümleri arasındaki TCP bağlantısının tıkanıklık penceresi büyüklüğü değişimi

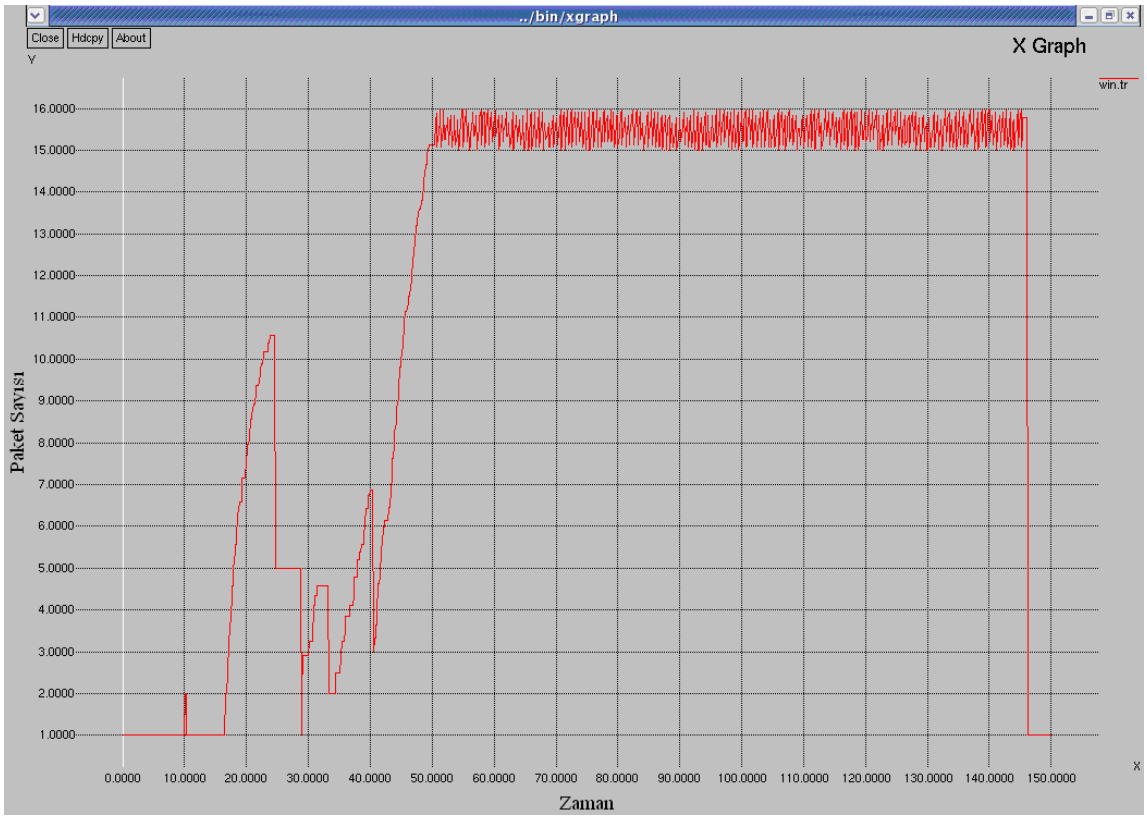
Geliştirilen protokol ile ilk senaryoda gerçekleştirilen benzetimde 0–1 düğümleri arasındaki TCP bağlantısının tıkanıklık pencere boyutu Şekil 5.8'de gösterilmektedir. Trafiğin başlaması ile birlikte tıkanıklık penceresi büyüklüğü arttırılmış, geliştirilen yöntemin sonucu olarak tıkanıklık penceresinin boyutu AODV protokolüne göre daha büyük değerlere ulaşmıştır. Genel olarak tıkanıklık penceresinin büyüklüğünün zamana göre seyri, geliştirilen protokolde daha yüksektir. Bunun anlamı, paket kayıplarının daha az olması ve ACK alınmadan gönderilen paket sayısının fazla olmasıdır. Bu da trafiğin rahatlığını göstermektedir. Grafikte en üstte meydana gelen salınımların nedeni yine kuyruk uzunluğunun 15 olmasıdır.



Şekil 5.8. Geliştirilen protokol ile ilk senaryoda gerçekleştirilen benzetimde 0-1 düğümleri arasındaki TCP bağlantısının tıkanıklık penceresi büyüklüğü değişimi

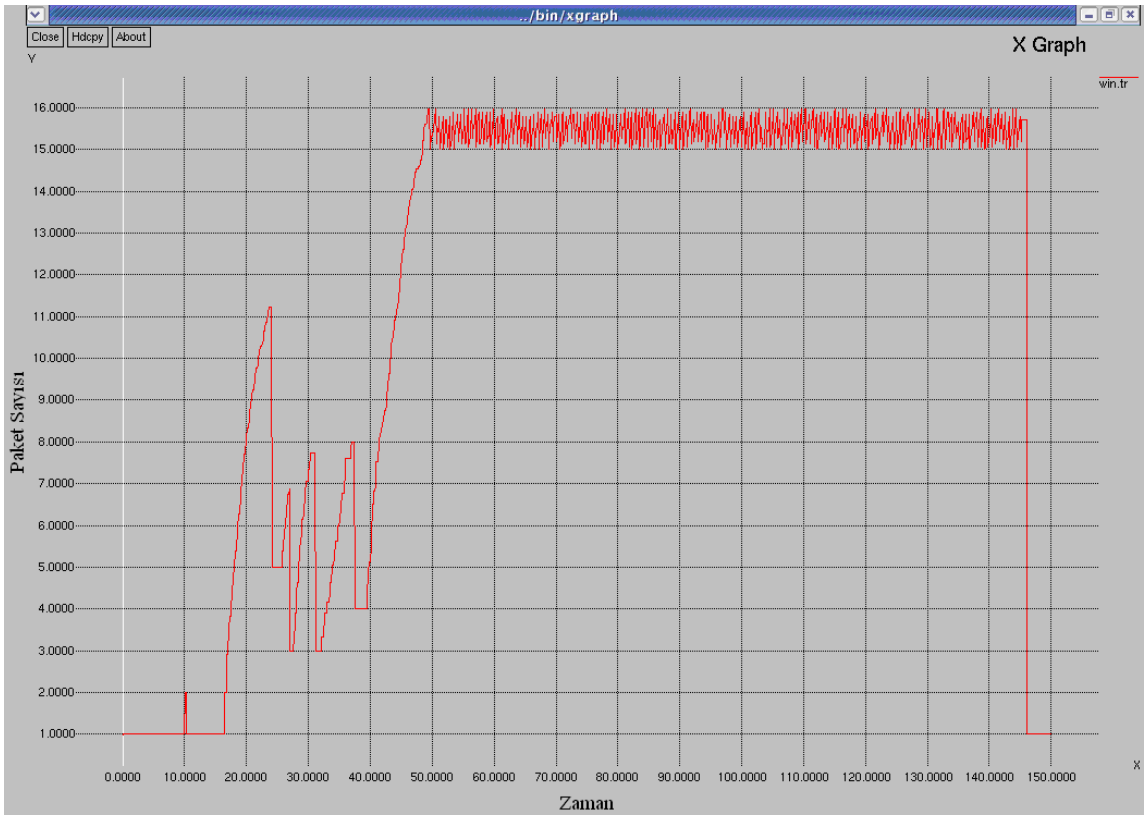
İkinci senaryo ile AODV protokolü kullanılarak elde edilmiş 0–1 düğümleri arasındaki TCP bağlantısının tıkanıklık penceresi büyüklüğü değişimi grafiği Şekil 5.9'da görülmektedir. İlk senaryoda olduğu gibi 15. saniyeden itibaren tıkanıklık penceresi büyüklüğü artmaya başlamış ve meydana gelen paket kayıpları nedeni ile genel olarak düşük seviyelerde seyretmiştir. Ağdaki diğer trafiklerin bitmesi ile birlikte maksimum değere ulaşmış ve bu şekilde trafik bitene kadar devam etmiştir.

Protokolleri karşılaştırmak için bir diğer kıstas da uçtan uca gecikme süreleri olabilir. Fakat tıkanıklık penceresinden ve atılan paket sayıları oranından uçtan uca gecikme süresinin nasıl olabileceği tahmin edilebilir. Bu yüzden bu benzetimde uçtan uca gecikme süresi dikkate alınmamıştır.



Şekil 5.9. AODV protokolü ile ikinci senaryoda gerçekleştirilen benzetimde 0-1 düğümleri arasındaki TCP bağlantısının tıkanıklık penceresi büyüklüğü değişimi

Geliştirilen protokol ile ikinci senaryoda gerçekleştirilen benzetimde 0–1 düğümleri arasındaki TCP bağlantısının tıkanıklık pencere boyutu Şekil 5.10'da gösterilmektedir. İlk senaryoda olduğu gibi bu senaryoda da geliştirilen yöntem kullanılarak gerçekleştirilen benzetim sonucunda elde edilen tıkanıklık penceresi büyüklüğünün AODV kullanılarak elde edilen tıkanıklık penceresi büyüklüğünden daha üst değerlerde seyrettiği görülmüştür.



Şekil 5.10. Geliştirilen protokol ile ikinci senaryoda gerçekleştirilen benzetimde 0–1 düğümleri arasındaki TCP bağlantısının tıkanıklık penceresi büyüklüğü değişimi

Gerçekleştirilen benzetimler sonucunda elde edilen değerler ve grafiklerden, geliştirilen protokolün kablosuz ağlarda yaygın olarak kullanılan bir protokol olan AODV protokolünden daha iyi sonuç verdiği gözlemlenmiştir. Oluşturulan koşullar, tıkanıklığın uç noktalarıdır. Topolojideki tüm düğümler iletişimde bulunmaktadır. Benzetimde birçok TCP bağlantısı vardır. Bu şartlar altında bir protokolün iyi sonuç vermesi, diğer şartlarda da iyi sonuç vereceğine bir delildir.

6. SONUÇLAR VE ÖNERİLER

Bu tezde kablosuz ağlarda tıkanıklık denetimi için bir protokol geliştirilmiştir. Geliştirilen protokol ns-2 benzetim aracı kullanılarak test edilmiş ve kablosuz ağlarda yaygın olarak kullanılan AODV protokolü ile karşılaştırılmıştır. Yaygın olarak kullanılan çoklu yol kurma ve sadece tek yol kurma yaklaşımıyla çalışan protokollerin önemli sakıncaları bulunmaktadır. Başlangıçta çoklu yol kurma yaklaşımlarında, bağlantılardaki kopukluklara uyum sağlanamamakta, yol kurma ve iletişim maliyeti yüksek olmaktadır. Sadece bir yol kurma ile iletişim yapan yaklaşımlarda ise tıkanıklık olması durumunda veya bağlantıların kopması durumunda iletişim tamamen kesilebilmektedir.

Geliştirilen protokol yükü birden fazla yola dağıtırken öncelikli olarak ilk kurulan yolu kullanmaktadır. Öncelikli olarak kurulan yolun tıkanmaya başlaması ile birlikte ikinci yolun uygun olması durumunda trafik buraya kaydırılmakta ve ilk belirlenen yoldaki kuyruk uzunluğu eşik değerin altına ininceye kadar ikinci yol kullanılmaktadır. Bunun yerine aktarımın sonuna kadar iki yol birlikte kullanılabilir. Ancak ilk belirlenen yol en kısa yol olduğundan ikinci yolun kullanılması iletişim maliyetini artıracaktır. Maliyet etkin en iyi performansı elde etmek için kuyruk eşik değerinin kurulacak yolun maliyetine göre iyi belirlenmesi ve uyarlanabilir olması gerekmektedir.

Deneysel çalışmaların sonucunda, geliştirilen protokolün en kısa yoldaki iletişim tamamen tıkanmadan, uzunluğu daha fazla olan başka bir yol kurup iletişimin bir kısmını bu yola aktarmasının toplam atılan paket sayısını önemli ölçüde azalttığı görülmüştür. Ayrıca atılan paketlerin kaynaktan tekrar gönderilmesinden dolayı ortalama paket gecikme süresi de atılan paket sayısı düştüğünden önemli ölçüde kısalmaktadır. Bunun yanı sıra daha az paket atıldığı için hedeften kaynağa ACK paketleri gönderimi sıklığı da düşmekte ve bunun sonucunda ağ trafiğinde de önemli ölçüde azalma olmaktadır.

KAYNAKLAR

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

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

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

4. Welzl M., "Network Congestion Control, managing internet traffic", ***Wiley***, New Jersey, 0-47-002528-X (2005).

5. C.E. Perkins and P. Bhagwat, "Highly Dynamic Destination - Sequenced Distance - Vector Routing (DSDV) for Mobile Computers", ***Comp. Comm. Rev.***, 234-244 (1994).

6. M. Allman, V. Paxson and W. Stevens, "TCP Congestion Control", ***IETF RFC 2581***, Virginia, 7-9 (1999).

7. Floyd S., "Congestion Control Principle", ***IETF RFC 2914***, Virginia, 6-7 (2000).

8. K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A Feedback-Based Scheme for Improving TCP Performans in Ad Hoc Wireless Networks", ***IEEE Personal Communications Magazine,*** 8(1): 34-39 (2001).

9. Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar, "The Zone Routing Protocol (ZRP) for Ad Hoc Networks", *Technical report*, ***Internet draft***, ***Virginia***, 5-6 (2002).

10. C. Siva Ram Murthy, B. S. Manoj, Ad-Hoc Wireless Networks- Architectures And Protocols, ***Prentice Hall***, New Jersey, 0-13-147023-X (2004).

11. Forouzan B., "Data Communications and Networking", ***Mc-Graw Hill***, New York, 71232419 (2006).

12. S. Oktuğ, "İTÜ Bilgisayar Mühendisliği Bölümü Bilgisayar Haberleşmesi Ders Notları", ***İTÜ***, 15-19 (2004).

13. Çölkesen R., Örencik B., "Bilgisayar Haberleşmesi ve Ağ Teknolojileri", ***Papatya Yayıncılık***, İstanbul, 9756797002 (2000).

14. Odom W., "CISCO CCNA #640-607 Sınavı Sertifikasyon Rehberi", ***CISCO Press***, İstanbul, 975322301-3 (2003).

15. Dongkyun Kim, C. K. Toh, Yanghee Choi, "TCP-BuS: Improving TCP Performans in Wireless Ad Hoc Networks", ***Journal of Communications and Networks***, 3(2): 12-14 (2001).

16. B. S. Manoj, R. Ananthapadmanabha, and C. Siva Ram Murthy, "Link Life-Based Routing Protocol for Ad Hoc Wireless Networks", ***Proceedings of IEEE ICCCN 2001***, 573-576 (2001).

17. J. C. Hoe, "Improving the Start-Up Behavior of a Congestion Control Scheme for TCP", ***Proceedings of the ACM SIGCOMM***, 270-280 (1996).

18. Tadeusz A. Wysocki, Arek Dadej, Beata J. Wysocki, "Advanced Wired and Wireless Networks", ***Springer*****,** New York, 0-387-22781-4 (2003).

19. D. B. Johnson and D. A. Maltz, "Dynamic Source Routing Protocol in Ad-Hoc Wireless Networks"*,* ***Kluwer Academic Publishers***, vol. 353, 153-181 (1996).

20. R. S. Sisodia, B. S. Manoj, C. Siva Ram Murthy, "A Preferred Link-Based Routing Protocol for Ad-Hoc Wireless Networks", ***Journal of Communications and Networks,*** 4(1):14-21 (2002).

21. Bür K., "Quality-of-Service-Aware Multicast Routing for Multimedia Applications in Mobil Ad Hoc Networks", *Doctor of Philosophy in Computer Engineering*, ***Boğaziçi Üniversitesi Fen Bilimleri Enstitüsü***, İstanbul, 32-37 (2006).

22. R. Sriram, G. Manimaram, and C. Siva Ram Murthy, "Preferred Link-Based Delay-Constrained Least-Cost Routing in Wide Area Networks", ***Computer Communications*****,** 21: 1655-1669 (1998).

23. D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, "Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms", ***In Proc. SIGCOMM 2001***, *ACM Computer Communication Review*, 31(4): 6-7 (2001).

24. L. S. Brakmo and L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet", ***IEEE Journal on Selected Areas in Communications***, 13(8): 1-22 (1995).

25. Santi P., "Topology Control in Wireless Ad Hoc and Sensor Networks", ***Wiley*****,** New Jersey, 0470094532 (2005).

26. J. Postel, "Transmission Control Protocol", ***IETF RFC 793***, ***Virginia,*** 1-19 (1981).

27. Charles E. Perkins, Elizabeth M. Belding-Royer, and Samir Das, "Ad Hoc On Demand Distance Vector (AODV) Routing", ***IETF RFC 3561***, ***Virginia,*** 1-10 (2003).

28. B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet", ***IETF RFC 2309***, ***Virginia,*** 3-7 (1998).

29. Stallings W., "Wireless Communications and Networks", ***Prentice Hall***, New Jersey, 0130408646 (2004).

30. Kurose J. F., "Computer Networking: A Top-Down Approach Featuring the Internet", ***Addison Wesley***, Boston, 0321227352 (2004).

31. Basagni S., Conti M., Giordano S., Stojmenovi I., "Mobile Ad Hoc Networking", ***Wiley-IEEE Press***, New Jersey, 0471373133 (2004).

32. Aggelou G., "Mobile Ad Hoc Networks: From Wireless LANs to 4G Networks", ***McGraw Hill***, New York, 0071413057 (2004).

33. G. Holland and N. Vaidya, "Analysis of TCP Performance over Mobil Ad Hoc Networks", ***Proceedings of ACM MOBICOM, Texas,*** 219-230 (1999).

ÖZGEÇMİŞ

Kişisel Bilgiler

Soyadı, adı	: ŞİMŞEK, Mehmet
Uyruğu	: T.C.
Doğum tarihi ve yeri	: 22.09.1981 Şavşat
Medeni hali	: Bekâr
Telefon	: 0 (312) 369 14 22
Faks	: 0 (312) 230 84 34
e-mail	: mehmet.simsek@gazi.edu.tr

Eğitim

Derece	**Eğitim Birimi**	**Mezuniyet tarihi**
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. Akcayol M.A., Şimşek, M., Bay, İ., Toklu, S., Doğru, İ.A., "Genetic Algorithm Based Location Optimization of Emergency Service Units", 4th FAE International Symposium, European University of Lefke, 2006.
2. Akcayol M.A., Şimşek M., Bay İ., "Türkiye'de E-Kütüphane Çalışmalarının Durum Analizi ve Öneriler", Akademik Bilişim 2005, Gaziantep Üniversitesi, Gaziantep, 02-04 Şubat, 2005.