İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TELSİZ ÖRGÜ AĞLAR ÜZERİNDE DOSYA PAYLAŞIMI VE AĞ KODLAMA UYGULAMALARI

YÜKSEK LİSANS TEZİ

Reşat Andaç ÖZER

Anabilim Dalı : Elektronik ve Haberleşme Mühendisliği

Programı : Telekomünikasyon Mühendisliği

EKİM 2010

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TELSİZ ÖRGÜ AĞLAR ÜZERİNDE DOSYA PAYLAŞIMI VE AĞ KODLAMA UYGULAMALARI

YÜKSEK LİSANS TEZİ

Reşat Andaç ÖZER
(504051329)

Tezin Enstitüye Verildiği Tarih : 21 Eylül 2010
Tezin Savunulduğu Tarih : 1 Ekim 2010

Tez Danışmanı : Prof. Dr. M. Ertuğrul ÇELEBİ (İTÜ)
Diğer Jüri Üyeleri : Prof. Dr. Ümit AYGÖLÜ (İTÜ)
Prof. Dr. Serhat ŞEKER (İTÜ)

EKİM 2010

ÖNSÖZ

Bu tezin hazırlanmasında değerli katkılarından dolayı Prof. Dr. Mehmet Ertuğrul ÇELEBİ'ye, her türlü konu hakkında bilgisini esirgemeyen bütün haberleşme anabilim dalı araştırma görevlilerine, tez çalışmam boyunca manevi ve maddi desteklerini benden esirgemeyen, bana her zaman moral veren babam Remzi ÖZER, annem Güler ÖZER, ablam Özlem ÖZER'e teşekkür ederim.

Ekim 2010

Reşat Andaç ÖZER

Elektronik ve Haberleşme Mühendisi

İÇİNDEKİLER

Sayfa

KISALTMALAR

P2P : Eşler Arası İletişim
GF : Galois Alanı
LAN : Yerel Alan Ağ
WLAN : Telsiz Yerel Alan Ağ
IEEE : Elektrik Elektronik Mühendisleri Enstitüsü
MAN : Metropol Alan Ağ
AP : Erişim Noktası
GHz : Gigahertz
DHCP : Dinamik İstemci Ayarlama Protokolü
IP : Internet Protokolü
USB : Evrensel Seri Veriyolu
GPS : Küresel Konumlama Sistemi
MANET : Gezgin Tasarsız Ağ
VANET : Araçlar için Gezgin Tasarsız Ağ
DHCP : Dinamik İstemci Ayarlama Protokolü
FHSS : Frekans Atlamalı Yayılmış Spektrum
DSSS : Doğrudan Dizi Yayılmış Spektrum

ÇİZELGE LİSTESİ

Sayfa

ŞEKİL LİSTESİ

Sayfa

DENKLEM LİSTESİ

Sayfa

TELSİZ ÖRGÜ AĞLAR ÜZERİNDE DOSYA PAYLAŞIMI VE AĞ KODLAMA UYGULAMALARI

ÖZET

Günümüzde kullanılan telli ve telsiz ağlarda, bilgi iletimi, yönlendirme yöntemleri kullanılarak gerçekleştirilmektedir. Ağ kodlama yöntemi, mevcut yönlendirme yöntemlerine yardımcı olarak kullanılmakta ve ağ üzerindeki bir yönlendiriciye (router) gelen verilerin çeşitli yöntemlerle birleştirilerek, kodlanması esasına dayanmaktadır. Ağ kodlama matematiksel olarak lineer birleşim işlemi esaslarını temel almakta ve buna uygun algoritmalar kullanmaktadır. Bu şekilde verilerin ağ üzerinde yayılması hızlanmaktadır.

Telsiz yerel alan ağları üzerinde veri alış-verişinin yoğunlaşması, telsiz örgü ağlar üzerinde hızlı dosya paylaşımı sağlayan uygulamaların önemini, gitgide arttırmaktadır. Dosya paylaşımında telli ağların, hizmet süresi açısından, daha etkin kullanılmasını sağlayan uygulamalar, telsiz örgü ağlar üzerinde de uygulanabilmektedir. Dosya paylaşım tekniklerinin en popülerinden biri Gnutella'dır. Paylaşılan dosya parçalarını ağ üzerinde yönlendirmeyi sağlayan, Gnutella tabanlı dosya paylaşım uygulamaları, ağ üzerinde paket yönlendirme işlemini gerçekleştirmek için dedikodu (gossip) yönlendirme yöntemini kullanmaktadır.

Bu çalışmada, telsiz örgü ağ topolojisi üzerinde, gossip yönlendirme yöntemi kullanan bir dosya paylaşım uygulaması, hem ağ kodlamalı, hem de ağ kodlamasız durum için incelenmektedir. Bu uygulama gerçek hayatta eşler arası ağ düzeninde çalışan telsiz yerel ağ üzerinde, yama yazılımların, tüm ağ elemanlarına ulaştırılması işlemine karşılık gelmektedir. Ayrıca ağ elemanlarının kapsama alanlarındaki artışın ağ kodlama yöntemine olumlu etkisi de yine bu çalışmada incelenmektedir. Sonuç olarak, bu çalışmada, ağ kodlama yönteminin telsiz örgü ağlardaki dosya paylaşımına, hizmet süresi açısından olumlu katkı sağladığı ortaya konmaktadır.

FILE SHARING OVER WIRELESS MESH NETWORKS AND NETWORK CODING APPLICATIONS

SUMMARY

In today's wired and wireless networks, end-to-end information delivery is performed by routing methods. Network coding technique, is used as an auxiliary method for routing methods. A network node, can combine incoming data and can obtain new output data with using network coding techniques. Network coding is based on the principles of linear combination operations and it uses algorithms accordingly. According to this approach, file distribution over a network, accelerates.

The importance of wireless mesh network topology usage over wireless local area networks, is increasing day by day. Also file sharing applications are getting very useful and popular, and increase of data traffic over LAN's requires fast file sharing applications. The file sharing applications, which reduce file sharing service time on wired networks, can also be used for wireless networks. One of the most popular file sharing protocol is Gnutella. Gnutella protocol, uses gossip routing for file sharing applications.

In this thesis, firstly, some simulations are executed for gossip-based file sharing application over wireless mesh networks. The file transfer service time results are acquired. Then, network coding is applied to gossip-based file sharing application over wireless mesh network. Network coded and non-network coded service time results are compared. In real world, distribution process of software patches over a wireless mesh network, is similar with our simulation environment. On the other hand, it is shown that the increase in network elements' wireless coverage area, improves performance of network coding. Finally, the benefit of network coding over wireless mesh network's data distibution service time, is shown.

1.GİRİŞ

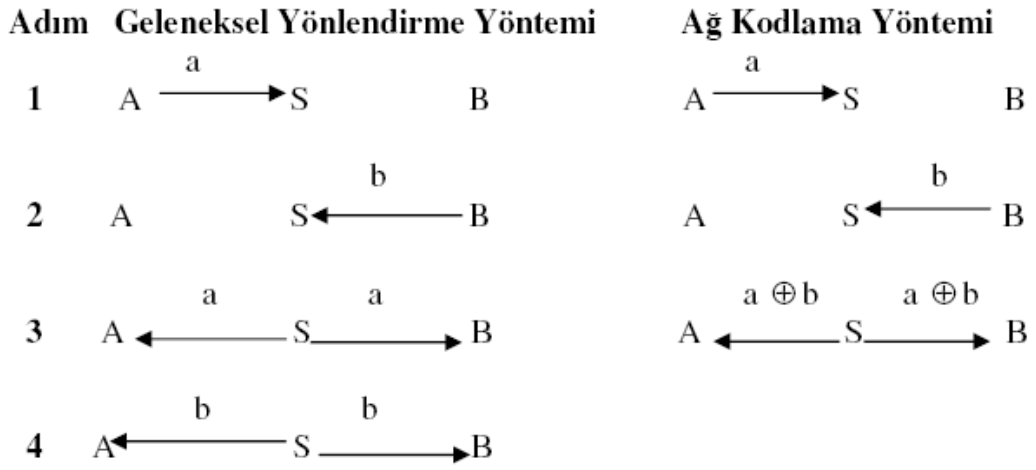
Haberleşme ağlarının temel çalışma prensibi genel olarak benzerdir. Haberleşme ağları veri iletimi için yönlendirme prensiplerinden faydalanmaktadır. Günümüzde çoğunlukla kullanılan geleneksel yönlendirme metodu, sakla ve ilet şeklinde çalışmaktadır. Bu yönteme göre, ağa giriş yapan bilginin kodlama ve sıkıştırma işlemleri, dosya iletimini başlatan ağ elemanında yapılmaktadır. Ayrıca kanal kodlama işlemi iki ağ düğümü arasındaki fiziksel bağlantıda da yapılmaktadır.

Geleneksel yönlendirme yöntemlerinde, ağ düğümleri üzerinde yoğun işlemci kapasitesi gerektirecek işlemlerden kaçınılmıştır. Çünkü herhangi bir birleştirme işlemini gerçekleştirmek için , yönlendiriciler içerisinde mikroişlemci kullanımını gerektirmektedir. Yakın geçmişte, işlemci maliyetleri yüksek olmasından ve işlemci teknolojisinin uygun fiyatlarla yeterince hızlı işlemciler geliştirmeye musait olmamasından dolayı, ağ elemanlarında hızlı işlemcilerin kullanımından kaçınılmaktaydı. Internet üzerindeki bilgi paketlerinin iletimi göz önüne alındığında, birbirinden bağımsız bilgi akışları, aynı ağ yönlendiricilerinin (router) veya tekrarlayıcılarının (repeater) üzerinden akabilir, ancak bu ağ üzerindeki yönlendiriciler veya tekrarlayıcılar, paketleri yalnızca saklayıp iletmektedir. Fakat günümüzde yeterince hızlı işlemciler oldukça uygun maliyetlerde üretilmektedir ve dolayısıyla ağ elemanları üzerinde, iletilmeye çalışılan paketleri birleştirme işlemini gerçekleştirmek de, düşük maliyetlerle mümkün olmaktadır.

Ağ kodlama, ağ üzerindeki bir yönlendiriciye veya tekrarlayıcıya gelen paketlerin, çeşitli matematiksel yöntemlerle birleştirilerek, yeni bir çıkış verisi elde edilmesi yöntemidir. Ağ kodlama, geleneksel yönlendirme yöntemlerine yardımcı olarak kullanılmaktadır. Ağ kodlama, yönlendirme yöntemlerine yeni bir yaklaşım getirmektedir.

Ağ kodlaması yöntemi sayesinde, herbir ağ yönlendiricisi, kendisine komşu yönlendiricilerden ard arda gelen bilgi paketleri üzerinde matematiksel işlemleri

kullanarak, bu paketleri tek bir bilgi paketiymiş gibi hedeflenen ağ elemanına iletebilmektedir. Şekil 1.1 ağ kodlaması teknikleri için basit bir örnek oluşturur.



Şekil 1.1 : Ağ kodlama metodu - geleneksel yönlendirme metodu karşılaştırması

Şekil 1.1' de üç noktalı ağ topolojisi görülmektedir. Bu basit örnek, ağ kodlama yönteminin hizmet süresinde kısalma sağladığını göstermesi açısından önemlidir. Bu çalışmada, hizmet süresi, veri iletişiminin başlamasından sonlanmasına kadar geçen süredir.

Şekil 1.1'de , A ve B ağ elemanları arasında, S ağ elemanı üzerinden bilgi alışverişi yapılmaktadır. Buradaki S ağ elemanını bir baz istasyonu gibi düşünebiliriz. A ve B elemanları ise ,bir cep telefonu veya herhangi bir mobil iletişim cihazı gibi düşünülebilir. A elemanı B' ye "a" paketini göndermektedir. B elemanı A düğümüne "b" paketini göndermektedir. Daha sonra S elemanı A ve B'ye, kendisine gelmiş olan, her iki paketi de göndermektedir. Bu işlem 4 birim zamanda gerçekleşmektedir.

Ağ kodlama yönteminde ise, S elemanı, A ve B' ye, paket göndermeden önce basit bir matematiksel işlem olan "ayrıcalıklı veya" ($\oplus$) işlemini kullanmaktadır. Bu işlem 3 birim zamanda gerçekleşmektedir. Sonuç olarak ağ üzerindeki, bilgi paketinin iletimi için gerçekleşen toplam adım sayısı 1 adet azalmıştır. Dolayısıyla hizmet süresi de, teorik olarak 1 birim süre azalmış olmaktadır.

Ağ kodlama için, matematiksel yöntem olarak, bir Galois sonlu alanında tanımlı lineer birleşim işlemi kullanıldığında, yöntem "lineer ağ kodlama" olarak adlandırılmaktadır. Bu çalışmadaki uygulamalarda, lineer ağ kodlama yöntemi kullanılmaktadır .

1.1. Tezin Amacı

Günümüzde, dosya paylaşım uygulamalarında, bir kullanıcıdaki bilgi, ağdaki diğer kullanıcılar tarafından paylaşılmaktadır. Telsiz yerel alan ağlar üzerinde, telsiz örgü ağ topolojisinin kullanımı, telsiz modemlerin kullanım oranının artması ve dar kapsama alanı içerisinde yüksek veri iletişim hızları sunan, küçük hücre (small cell) ürünlerinin yaygılaşması ile beraber daha da önem kazanmaktadır. Dolayısıyla, telsiz örgü ağlarda, hızlı veri alışverişi yapmak gitgide önemli hale gelmekte ve bu konudaki bilimsel çalışmalar da yoğunluk kazanmaktadır.

Dosya paylaşımında, telli ağı hizmet süresi açısından, daha performanslı kullanmamızı sağlayan uygulamalar, telsiz örgü ağlar üzerinde de uygulanabilmektedir. Bu çalışmada, dosya paylaşımı sırasında Gnutella tabanlı dosya paylaşım uygulamalarının kullandığı gossip (dedikodu) yönlendirme yöntemi üzerinde ağ kodlamalı ve kodlamasız durumlar incelenmektedir. Gnutella protokolü ve Gossip yönlendirme yöntemi, günümüzde yaygın olarak, Limewire, Morpheus ve iMesh uygulamalarında kullanılmaktadır. Gossip yönlendirme yöntemi ile beraber ağ kodlama yönteminin uygulanmasının, telsiz örgü ağlar üzerinde hizmet süresini daha da kısaltacağı öngörülmektedir. Ayrıca bu çalışmada, telsiz örgü ağ elemanlarının kapsama alanlarındaki artışın ağ kodlama yönteminin performansına etkiside incelenmektedir. Ağ kodlamalı ve ağ kodlamasız durumlar arasındaki farkı incelemek ve ortaya somut sonuçlar koymak, bu tezin ana amacıdır.

1.2 Literaür Özeti

Ağ kodlama yönteminin temelleri R.W.Yeung ve Z. Zhang tarafından , iletişim ağları için 1999 yılında atıldı [1]. 2000 yılında R. Ahlswede, N. Cai, S-Y.R. Li ve R.W. Yeung, ağ bilgi akışını inceledikleri makalelerinde, ağ kodlama ismini ilk kez kullandılar. Bu yayında, ağ kodlamanın avantajlarından ayrıntılı olarak bahsedildi ve ağ kodlamasının en temel örneği olarak kabul edilen kelebek ağ modeli üzerinde uygulamalar gerçekleştirildi [2]. 2003 yılında N.Cai, S. R. Li , R.W. Yeung, yayınladıkları makale ile lineer ağ kodlamanın temellerini açıkladılar[3].2003 yılında T.Ho, R. Koetter, M. Medard, D.R. Karger ve M Effros tarafından lineer ağ kodlama sırasında rasgele katsayılar kullanmanın avantaj sağladığı ortaya kondu.[4]. 2005 yılında Gkantisidis tarafından, telli ağlarda dosya paylaşımı uygulamaları için ağ kodlaması metodundan faydalanılabileceği belirtildi ve kaynak düğümden gönderilen

bilginin çoğa erişimli (multicast) eşler arası (P2P) bir ağda, ağ kodlama yöntemi kullanılarak, daha hızlı servis edilebileceği, kanıtlandı [5]. 2005 yılında S. Katti, D. Katabi, W. Hu, H. Rahul, ve M. Medard tarafından telsiz örgü ağlarda ağ kodlama uygulamaları gerçekleştirildi [6]. 2006 yılında D.M. Chiu , R.W Yeung, J. Huang, B. Fan, teke gönderimli (unicast), eşler arası bir ağ yapısında, ağ kodlaması kullanılan ve kullanılmayan durumlarda, hizmet süresinin birbirine yakın değerlerde olduğunu ortaya koydular, ağ kodlamanın, hizmet süresinde herhangi bir performans düşüşü yaşamadan, tüm ağ topolojilerinde kullanılabileceğini gösterdiler. [7]. 2006 yılında, A.A., Hamra, C. Barakat, T. Turletti, yayınladıkları makale ile telsiz örgü ağlar üzerinde ağ kodlaması tekniğinin nasıl uygulanması gerektiğine ilişkin uygulamalar gerçekleştirdiler[8]. 2009 yılında, P. Sattari, A. Markopoulou ve C. Fragouli tarafından, birden fazla veri kaynağından, ağ üzerine bilgi akışı başlatılması durumunda, ağ kodlama yöntemini kullanmanın getireceği faydalar ortaya kondu[9].

Bu çalışmadaki uygulamalar, Gnutella dosya paylaşım protokolünde kullanılmakta olan gossip yönlendirme yöntemi üzerinde gerçekleştirilmektedir. Gnutella dosya paylaşım protokolü, ilk başlarda, çeşitli internet kullanıcılarının ortaya attığı yöntemlerden ibaretti. Gnutella protokolünün işleyişi , 2002 yılında M. Ripenau, I. Foster, A. Iamnitchi tarafından, ilk kez bir akademik çalışmayla ayrıntılı şekilde açıklandı[10]. 2002 yılında, Haas Z. J., J.Y. Halpern, L. Li tarafında gossip yönlendirme yönteminin, eşler arası paylaşım programlarına uygulanmasına ilişkin bir çalışma gerçekleştirildi, gossip yönlendirme yönteminin diğer yönlendirme yöntemlerine göre daha başarılı bir yöntem olduğu ortaya kondu[11]. 2007 yılında, gossip protokolünün Gezgin Tasarsız Ağ (MANET) üzerinde operatör servislerini dağıtılmasında kullanılmasına ilişkin S. Clarke ve R. Cunningham tarafından bir makale yayınlandı[12]. MANET' in yaygınlaşmasıyla birlikte gossip yönlendirme tekniği yeniden popüler hale geldi. 2009 yılında, F. Davide, R. Guerraoui, A.M. Kermarrec ve M. Manod tarafından gossip yönlendirme tekniği, canlı video paylaşımı için denendi ve hizmet süresinde performans artışı sağlandığı ortaya kondu[13].

Bu çalışmada, literatür özetindeki çalışmalar ışığında, geçmişte gerçekleştirilen ağ kodlama , Gnutella paylaşımı dosya paylaşımı uygulamaları, gossip yönlendirme ve telsiz örgü ağlar konuları ışığında, günümüzde telekom operatörlerinin önemli bir problemi olan, yama yazılımların sahada yer alan ağ elemanlarına dağıtılması

problemine fayda sağlanmaya çalışılmaktadır. Bu yönüyle bu çalışmadaki uygulama yeni ve ilk defa gerçekleştirilmiş bir uygulamadır.

Bu tezin, ikinci bölümünde öncelikle graflar konusu anlatılmaktadır. Daha sonra ağ kodlama yönteminin matematiksel altyapısı, sonlu alanlar aritmetiği , ve ardından da lineer ağ kodlama yöntemi ve faydaları açıklanmaktadır. Üçüncü bölümde, telsiz yerel alan ağları, örgü ağlar ve telsiz örgü ağlar, konuları anlatılmaktadır. Üçüncü bölümde ayrıca eşler arası (P2P) dosya paylaşım protokolü ve Gnutella dosya paylaşım protokolü açıklanmaktadır. Ayrıca bu bölümde eşler arası dosya paylaşımı uygulamalarında kullanılan gossip (dedikodu) yönlendirme yöntemi anlatılmakta, bu yöntem ve ağ kodlama ilişkisi açıklanmaktadır. Dördüncü bölümde, bu çalışmada gerçekleştirilen uygulamalara ve benzetimlere yer verilmektedir. Beşinci ve son bölümde sonuçlar ve tartışma yer almaktadır.

1.3 Hipotez

Gossip yönlendirme yöntemi, günümüzde P2P dosya paylaşımı, P2P TV ve VoD (video kesitlerini yükleme), gezgin tasarsız ağlarda katma değerli servislerin sunulması gibi dosya paylaşımının yoğun olduğu uygulamalarda kullanılan yönlendirme yöntemlerindendir. Dosya paylaşımda, gerek veri iletimi aşamasında, gerekse de talep edilen dosyayı, diğer kullanıcılardan sorgulaması aşamasında gossip yönlendirme kullanılmaktadır.

Bu çalışmada, telsiz örgü ağ topolojisi üzerinde, gossip yönlendirme yöntemi kullanan bir dosya paylaşım uygulaması, hem ağ kodlamalı, hem de ağ kodlamasız durum için incelenmekte, ağ kodlama yönteminin hizmet süresinde kısalma sağladığı ortaya konmaya çalışılmaktadır.

Öncelikle, sadece gossip yönlendirme uygulanmış telsiz örgü ağda hizmet süresi hesaplanmakta, daha sonra aynı sistem üzerine ağ kodlama yöntemi uygulanmakta ve ardından sonuçlar karşılaştırılmaktadır. Benzetimlerin sonucunda, ağ kodlamalı durumda, dosya paylaşımının daha kısa sürdüğü sonucunun ortaya konulması amaçlanmaktadır.

Bu çalışma sonucunda, n adet telsiz örgü ağ düğümüne, m adet bilgi paketinin en hızlı şekilde paylaştırılabilmesi konusundaki çalışmalara katkı sağlanmaya çalışılmaktadır. Bu problemin, pratikteki karşılığı olarak, telsiz modemlerin üzerinde ve telsiz bağlatıya sahip aygıtların üzerinde donanım yazılımları güncellenmesi veya

telekom operatörlerinin yama yazılımlarını telsiz ağ elemanlarına dağıtması uygulamaları örnek gösterilebilir. Ayrıca telsiz örgü ağ elemanlarının kapsama alanlarındaki artışın, ağ kodlama yönteminin hizmet süresine etkisini olumlu yönde arttıracağının gözlenmesi öngörülmektedir. Sonuç olarak bu çalışma ile, ağ kodlama yöntemi kullanarak, telsiz örgü ağlarda dosya paylaşımı uygulamalarında, hizmet süresini kısaltmak amaçlanmaktadır.

2. GRAFLAR VE AĞ KODLAMA

Bilgisayar ağları, birbirlerine bağlı bilgisayarların haberleşmek, bilgi ve kaynaklarını paylaşabilmek için oluşturdukları yapılardır. En az iki bilgisayar birbirine bağlanarak, bir bilgisayar ağını oluşturulabilir. En bilinen bilgisayar ağı internettir. Ağ üzerinde istemci, sunucu, tekrarlayıcı vb. gibi birçok öğe mevcuttur. Bu tezde, bu aşamadan sonra geçen tüm ağ sözcükleri, bilgisayar ağları anlamında kullanılacaktır. Bilgisayar ağları ve bilgisayar ağlarına ilişkin problemler graflar üzerinde ifade edilmektedir.

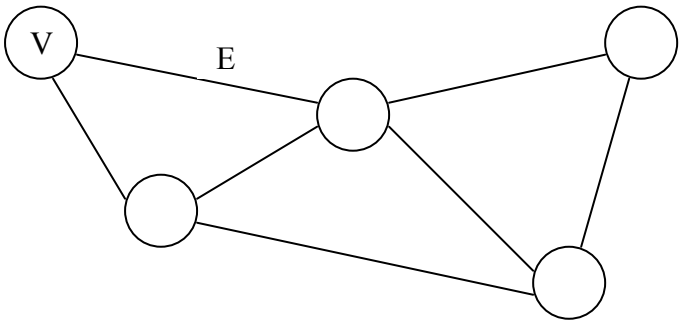
Ağ kodlaması uygulamalarını açıklamak için öncelikle bilgisayar ağ yapılarının graflar aracılığıyla nasıl gösterildiğini belirtmek gereklidir. Bu bölümde, birinci kısımda grafların tanımı ve graflara ilişkin temel kavramlar anlatılmaktadır. İkinci kısımda "ayrıcalıklı veya" işlemini kullanan ağ kodlama yöntemi anlatılmaktadır. Üçüncü ve dördüncü kısımda, sonlu uzaylar aritmetiği ve lineer ağ kodlama işlemi gösterilmektedir. Beşinci kısımda lineer ağ kodlama yönteminin sık kullanılan bir ağ topolojisi üzerinde, nasıl uygulanacağı anlatılmaktadır. Altıncı kısımda ise sonuç bulunmaktadır.

2.1 Graflar

Graf, düğüm olarak adlandırılan noktalar ve her biri bu noktaları veya sadece noktanın kendisini birleştiren ve ayrıt olarak adlandırılan çizgiler topluluğudur. Örnek olarak şehirleri düğüm ve onları bağlayan yolları ayırt olarak gösteren, yol haritaları verilebilir. Graflar, modern hayatın karmaşık ve geniş kapsamlı birçok probleminin çözümü için kullanılmaktadır. Bu uygulamalar; bilgi iletimi, taşıma planlaması, yönetim bilimi, satış planlaması gibi alanları kapsamaktadır. Graflardan, bu uygulamalardaki problemleri tanımlamada ve yapısal olarak ilişkileri belirlemede faydalanılmaktadır. Bir grafı tanımlamak için öncelikle düğümlerin ve ayrıtların kümesini tanımlamak gereklidir.

Köşelerin düğüm (V), kenarların ise ayrıt (E) olarak tanımlandığı grafın matematiksel olarak ifadesi G=<V,E> şeklindedir. Bir grafta düğümler nokta veya

daire şeklinde, ayrıtlar ise çizgi şeklinde gösterilir. Bir grafın büyüklüğü o graftaki düğüm sayısına eşittir. Bir düğümün derecesi de o düğüme giren ve çıkan ayrıtların toplamına eşittir. Şekil 2.1 örnek bir graf yapısıdır.

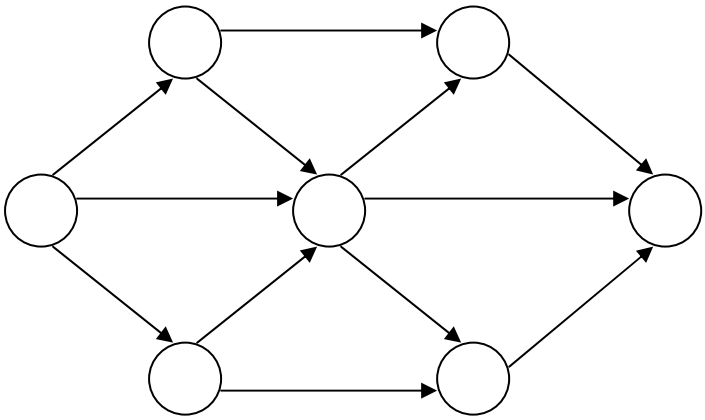


Şekil 2.1 : Örnek bir graf

2.1.1 Graf çeşitleri

Graflar özelliklerine göre çeşitlere ayrılmaktadır. Bunları; ağırlık ve yön bilgisinin olmadığı basit graf, yön bilgisinin olduğu yönlü graf, kenarların sayısal olarak ağırlıklandırıldığı maliyetli graf, her düğümün diğerine doğrudan bağlantısının olduğu tam graf, tüm düğümlerin derecelerinin eşit olduğu düzenli graf ve hiçbir kenarın birbirini kesmediği düzlemsel graf olarak sıralamak mümkündür.

Yönlü bir graf her bir ayrıtı sıra ile bir düğüm çiftiyle ilişkilendirilmiş ve ayrıtları yönlü olan graftır. Şekil 2.2'de yönlü graf yapısı gösterilmiştir.



Şekil 2.2 : 7 Düğümlü basit yönlü graf

Bu çalışmada, ağ üzerinde bilgi iletimine ilişkin problemleri tanımlamak için ve yapısal olarak ilişkileri belirlemek için graflar kullanılmaktadır. Aynı şekilde ağ

kodlamanın, ağ üzerinde uygulanması da graflar üzerinde ifade edilecek, yapısal ilişkiler graflar üzerinde gösterilecektir.

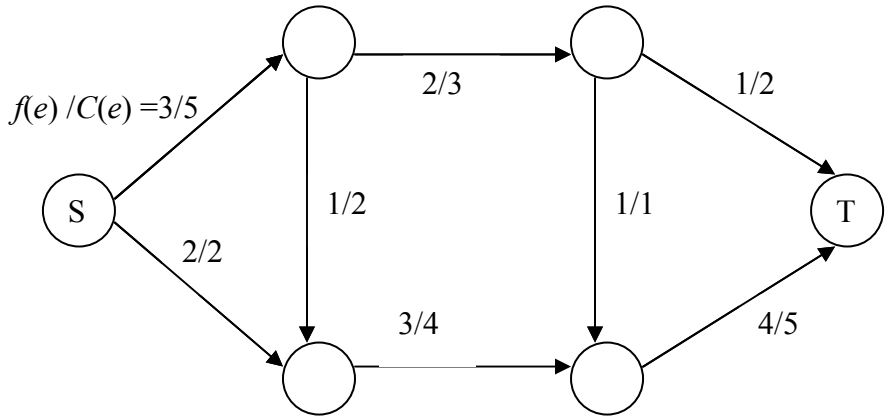
2.1.2 Ağ bilgi akışı ve graflar

Ağ yapısı , hiçbir gelen ayrıtı olmayan kaynak (source) düğümünü (*S*), hiçbir çıkan ayrıtı olmayan çıkış (sink) düğümünü (*T*) ve her ayrıt tayin edilmiş $e \subset E$ (*e:*ayrıt,*E*:ayrıtlar kümesi) pozitif bir kapasiteyi temsil eden kapasite fonksiyonunu *(C)* içeren yönlü bir graf yapısıdır[14].

$G = < V,E >$ (G:ağ (yönlü graf), *V*: düğümler, *E*: ayrıtlar).

Ağ üzerindeki herhangi düğümüne gelen tüm ayrıtlar (*e*) üzerindeki akış miktarının toplamı *f*(*e*) ile ifade edilmektedir.,

S düğümünden çıkan toplam akış miktarı *T* düğümüne gelen toplam akış miktarına eşittir. *f*(*e*)=*C*(*e*) eşitliği sağlandığı durumda, *e* ayrıtı doymuş hale gelmiştir. Şekil 2.3'te ağ bilgi akışı ve ayrıtlar üzerindeki kapasite bilgisine ilişkin bir örnek bulunmaktadır.



Şekil 2.3 : Ağ bilgi akışı ve kapasite

2.3 Lineer ağ kodlama ve matematiksel altyapı

Lineer ağ kodlama yöntemi, ağ kodlama işlemi sırasında, kaynak düğümden gelen, orijinal paketlerin lineer birleşimlerini hedef düğümlere ileten bir tekniktir. Bir bilgisayar ağındaki yönlendirici ele alınsın. Bu yönlendirici, graflar üzerinde bir düğüme karşılık gelmektedir. Geleneksel paket yönlendirme yöntemlerinde, yönlendiriciler bir bilgi paketini hedefe yönlendirirken basitçe tekrarlama yapmaktadır. Ağ kodlaması tekniği sayesinde bir yönlendirici birden fazla paketi

birleştirip tek bir paket halinde komşu yönlendiricilere gönderebilmektedir. Her bir paket, L bit uzunluklu olsun. Aynı uzunluklu olmayan paketler arasında da en uzun paketin boyu L kabul edilerek, paketlerin birleştirilmesi sırasında kısa olan paketlerin sonuna "0" lar eklenerek kısa paketlerin boyu L bit yapılsın. Lineer ağ kodlama yönteminde, bir yönlendirici düğümden çıkan paketler, bu yönlendiricinin kendisine komşu düğümlerden gelmiş olan paketlerin lineer birleşimi olmaktadır.

Lineer birleştirme, birbiri ardına ekleme işlemi değildir. Eğer L uzunluklu paketleri lineer bir şekilde birleştirirsek kodlanmış paketin uzunluğu da L olacaktır. Birbiri ardına eklemenin aksine her bir kodlanmış paket, kaynak düğümden gelen paketlerin içerdiği bilginin bir bölümünü içinde barındırır. Birbiri ardına eklemede ise gelen bilgiler aynen çıkış paketinde de bulunmalıdır ve çıkış paketinin boyunu L ile sınırlamak mümkün olmamaktadır[3,15].

Paket boylarını sınırlı tutabilmek için sonlu alanlar aritmetiğini kullanmak gereklidir. Ağ kodlama işlemi gerçekleştirilirken, uygulanan lineer birleşim işlemleri Galois Alanı üzerinde tanımlı işlemler olacaktır.

Lineer ağ kodlama işleminin matematiksel altyapısını ifade edebilmek için, öncelikle, sonlu alanlar aritmetiği ve Galois Alanı konuları anlatılmalıdır.

2.3.1 Sonlu alanlar aritmetiği

Aritmetik işlemlerde, işlem sonrasında sonuç uzunluğu değişmektedir. Çarpma veya toplama yapıldığında sonucun gösterileceği bit sayısı da artacaktır. Sonlu alanlar aritmetiği, bu problemi ortadan kaldırmaktadır. Sonlu alanlar aritmetiğinde tanımlı tüm işlemler, yine aynı uzayda sonuçlar üretmekte, sonucun uzunluğu değişmemektedir. "Galois Alanı", sonlu uzayın bir alt kümesidir.

2.3.2 Galois alanı

$GF(p^n)$ gösteriminde p Galois Alanı'nın karakteristiğidir. Galois Alanı'nın kaç farklı eleman içereceğini gösterir ve asal bir sayı olmalıdır. Ağ kodlaması uygulamalarında p değeri 2 olarak seçilir. n ise toplam kaç $GF(p^n)$ elemanı bulunduğunu gösterir. Örneğin, $GF(2^8)$ alanında 256 adet farklı eleman bulunur. Bu sayıların her biri 1 bayt ile ifade edilip yapılan işlemlerde n. dereceden indirgeme polinomu ile indirgeme yapılır. İndirgeme işlemi primitif polinoma göre alınana modulo işlemidir. Yapılan bu indirgeme ile sonucun yine aynı sonlu uzayda oluşması sağlanmış olur.

2.3.2.1 Galois alanında elemanların gösterilimi

Galois Alanı'nda elemanlar ikilik (binary), onaltılık (hexadecimal) veya polinom şeklinde gösterilebilirler. Galois Alanı'nda çarpma ve toplama işlemleri, polinomlar ile gösterilirler.

$b \in GF(2^n)$ 'nin polinom gösterilimi aşağıdaki gibidir.

$$b(x) = \sum_{i=0}^{n-1} b_i \cdot x^i = b_{n-1} \cdot x^{n-1} + b_{n-2} \cdot x^{n-2} + \ldots + b_1 \cdot x + b_0 \qquad \textbf{(2.1)}$$

Örnek olarak 8 bitlik bir eleman olan {10001011} elemanı için 2.1 denklemini uygularsak,

$$b(x) = x^7 + x^3 + x + 1$$

şeklinde gösterilir.

2.3.3 Galois alanında aritmetik işlemler

Galois Alanı'nda toplama ve çarpma işlemleri, karakteristiğinin 2 olması nedeniyle, özel olarak tanımlanmışlardır.

2.3.3.1 Galois alanında toplama işlemi

Polinom toplama işleminde, aynı üstel değere sahip sayılar toplanırlar ve oluşan ara sonuca, modulo 2 işlemi uygulanır. Bu işlemle sonucun sonlu alanda kalması sağlanır. Aşağıdaki örnek Galois Alanı'nda toplama işlemini göstermektedir.

$$a = \{00001001\} \Rightarrow a(x) = x^3 + 1$$
$$b = \{10001011\} \Rightarrow b(x) = x^7 + x^3 + x + 1$$
$$a(x) + b(x) = x^7 + 2x^3 + x + 2$$

Modulo 2 uygulandıktan sonra,

$$c(x) = x^7 + x$$

haline gelir.

$c = \{10000010\}$ olmaktadır.

Başka bir yol da a ve b sayılarının her birini bit düzeyinde "ayrıcalıklı veya" işlemine sokmaktır. $GF(2^p)$ sonlu alanında her bir bit 2 farklı değer alabilir, bu değerler lojik 1 veya 0' dır. Bu işlem 2.2 denklemi ile gösterilebilir.

$$c(x) = a(x) + b(x) = \sum_{i=1}^{n-1} (a_i \oplus b_i) x^i \qquad \textbf{(2.2)}$$

Denklem uygulanırsa,

$$c = \{00010001\} \oplus \{10010011\} = \{10000010\}$$

$c = \{10000010\}$ değeri kolayca bulunur.

Bit seviyesindeki işlemlerde çıkarma işlemi, toplama işleminin aynısı olmaktadır.

2.3.3.2 Galois alanında çarpma işlemi

GF(2^8) alanında, çarpma işlemi, iki polinomun 8. dereceden indirgenemez polinom ile modulo alınması işleminden oluşur. Şöyle söylenebilir ki, 8. dereceden indirgenemez polinomun böleni tektir ve sadece kendisidir. Örnek olarak aşağıdaki polinom incelenecektir.

$m(x) = (x^8 + x^4 + x^3 + x + 1)$ indirgenemez (primitif) polinom.

$$(x^6 + x^4 + x^2 + x + 1)(x^7 + x + 1) = x^{13} + x^{11} + x^9 + x^8 + x^7 +$$

$$x^7 + x^5 + x^3 + x^2 + x +$$

$$x^6 + x^4 + x^2 + x + 1$$

$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$$

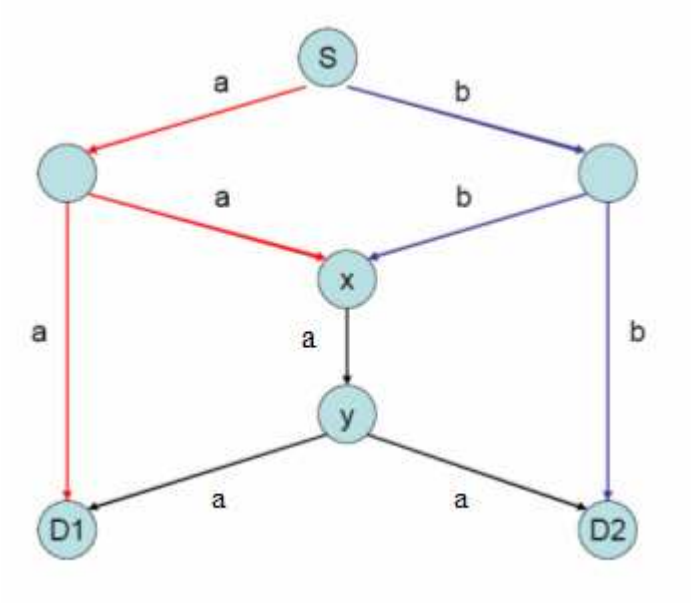
$$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1) \bmod (x^8 + x^4 + x^3 + x + 1) = (x^7 + x^6 + 1)$$

Modulo işlemi sonucunda, sonucun derecesi 8'den küçük olacaktır.

2.4 Ağ Kodlama Uygulamaları

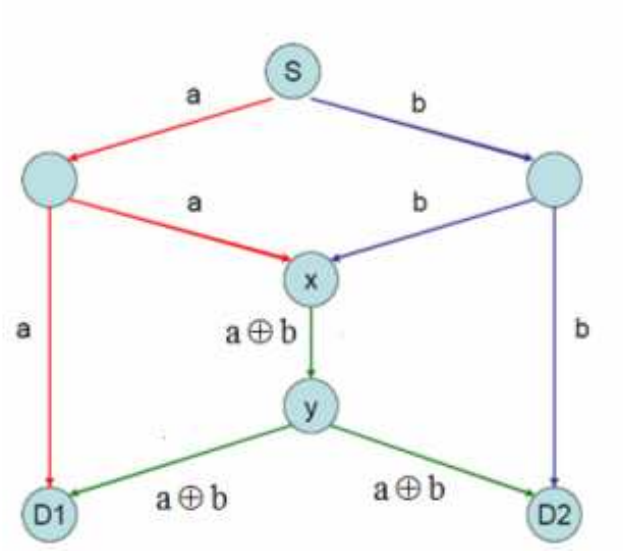
2.4.1 Kelebek ağ

Lineer ağ kodlama yöntemi, ağ kodlama işlemini gerçekleştirirken lineer birleşimlerden faydalanmaktadır. "Ayrıcalıklı veya" işlemi kullanılarak elde edilen lineer birleşim ve bu birleşimlerin getirdiği faydalar bu bölümde kelebek ağ topolojisi üzerinde anlatılacaktır. Şekil 2.4 üzerinde kelebek ağ topolojisi görülmektedir. Şekil 2.4'teki ağ geleneksel yönlendirme yöntemleri kullanılan, ağ kodlamasız bir ağdır.



Şekil 2.4 Kelebek ağ ve geleneksel yönlendirme yöntemi

Ağda amaç, S düğümünden çıkan tüm paketlerin, D1 ve D2 kaynak düğümlerinde aynı anda ulaştırılmasıdır. Şekil 2.4'te görüldüğü üzere, S düğümünden gönderilen "a" ve "b" aynı uzunluklu paketleri, X düğümüne geldiğinde, ya "a" ya da "b" paketi tercih edilmelidir. Burada X düğümü tarafından öncelikle "a" paketi tercih edilmiştir. Bu seçim sonuncunda D2 üzerinde "a" ve "b" paketlerinin her ikisinin elde edildiği anda, D1 üzerinde yalnızca D1 paketi elde edilmiş olacaktır. Bu işlem 4 birim zaman sürmüş, istenilen amaca ulaşılamamıştır.Şekil 2.5'te ise, Şekil 2.4'te gösterilen durum için ağ kodlama uygulanmıştır.



Şekil 2.5 Kelebek ağ ve ağ kodlama yöntemi

Şekil 2.5'te görüldüğü üzere, X düğümü üzerinde "ayrıcalıklı veya" işlemi yapılmaktadır. Böylece tekbir paket içerisinde,hem "a" hem de "b" verisi birleştirilmiş olarak taşınmaktadır. Bu işlem sonucunda D1 ve D2 düğümleri aynı anda her "a" ve "b" paketlerinin birleşimini almış olmaktadır [2]. D1 ve D2, aldıkları bu birleşimleri, lojik işlemler ile kolayca çözüp, hem "a" hem de "b" paketlerini elde edebilmektedirler.

2.4.2 Lineer ağ kodlama

$M_1,...,M_n$ belirli sayıda bir ya da birkaç kaynak tarafından oluşturulmuş orijinal paketler olsun. Lineer ağ kodlama metodunda, her bir paket, sıra ile bir katsayı ile ilişkilendirilmektedir. Bu ilişkilendirme $X=\sum_{i=1}^{n} g_j M_i$ şeklinde bir lineer birleşimdir. Bu $\{g_1,....,g_n\}$ katsayıları kümesi ikili Galois alanı GF(2^n) içindedir yani $\{0,1\}$ sembol kümesinin elemanıdır.

Şekil 1.1' deki örnek tekrar ele alınırsa, $M_1=a$ ve $M_2=b$ olarak kabul edilirse, S düğümü tarafından oluşturulan lineer birleşim M_1+M_2 şeklinde gösterilebilir.

Tüm katsayıları içeren pakete kodlayan vektör denir ve $g=(g_1,...,g_n)$ elemanlarından oluşur.

$$X_j=\sum_{i=1}^{n} g_{ij} M_i \qquad \textbf{(2.3)}$$

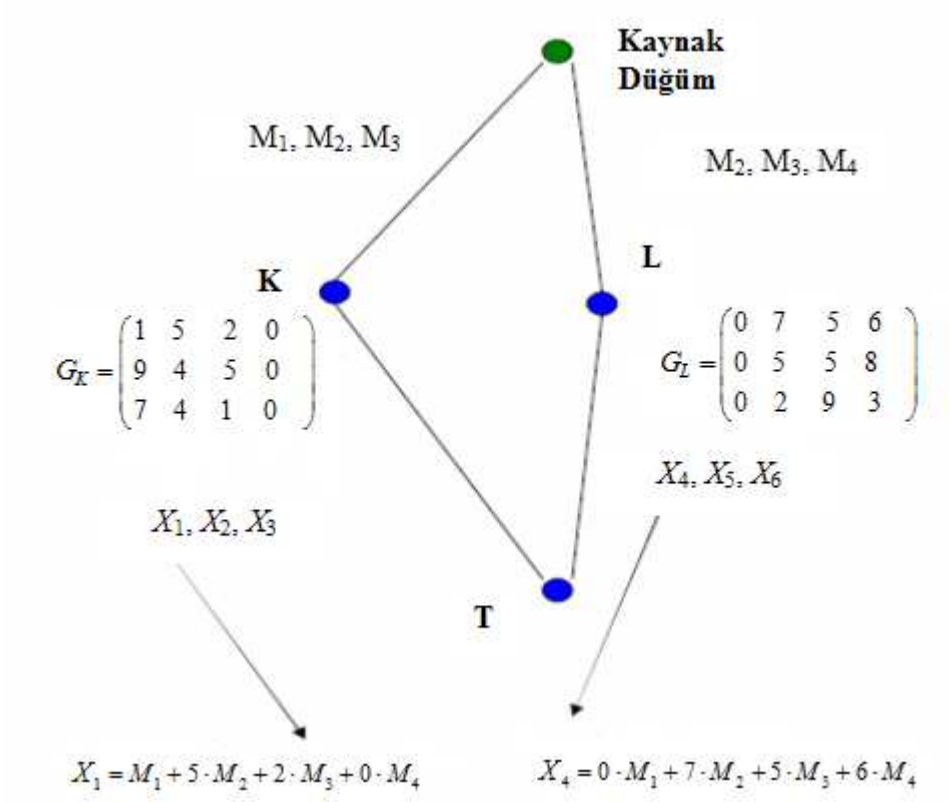
vektörü, bir düğüme ait herhangi j. çıkış paketine ait i bilinmeyenli bir denklem olmaktadır. Şekil 2.6'da ağ kodlama yönteminin uygulandığı bir ağ görülmektedir.

Şekil 2.6'dan da anlaşılabileceği gibi K düğümünün j. çıkış denklemi $X^j=\sum_{i=1}^{n} g_i^j M^i$ formüle edilebilmektedir. K düğümüne ait lineer kodlama bilgi vektörü

$$X=\sum_{j=1}^{m}\sum_{i=1}^{n} g_{ji} M_i \qquad \textbf{(2.4)}$$

olmaktadır[15].

Şekil 2.6'da sembolik katsayılar matrisinin, sayısal katsayılarla gösterildiği bir örnek görülmektedir.



Şekil 2.6 : Lineer ağ kodlama ve katsayılar

Şekil 2.6' da görüldüğü üzere, kaynak düğümden ağa gönderilen paketler (M_1, M_2, M_3, M_4) 'dir. M_1, M_2, M_3 paketleri K düğümüne , M_2, M_3, M_4 paketleri ise L düğümüne 1 birim zamanda iletilmektedir. K düğümü üzerinde lineer ağ kodlaması işlemi gerçekleştirilmektedir. K düğümünde X_1, X_2, X_3 çıkışları, L düğümünde ise X_4, X_5, X_6 çıkışları elde edilmektedir[16].

K düğümüne ilişkin lineer kodlama bilgi vektörü ;

$$X = \sum_{j=1}^{3} \sum_{i=1}^{4} g_{ji} M_i = \begin{pmatrix} 1 & 5 & 2 & 0 \\ 9 & 4 & 5 & 0 \\ 7 & 4 & 1 & 0 \end{pmatrix} . \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{pmatrix} = \begin{pmatrix} M_1 + & 5M_2 + & 2M_3 \\ 9M_1 + & 4M_2 + & 5M_3 \\ 7M_1 + & 4M_2 + & M_3 \end{pmatrix}$$

olmaktadır. Bu bilgi matrisinin içerisinde lineer birleşimde kullanılan katsayılara ilişkin tüm bilgiler bulunmaktadır. Bu matris çözülebilir bir denklem takımıdır.

Sisteme bir F düğümü daha eklenirse, denklem takımlarını genişletmek gerekir.

Fakat bunu gerçekleştirirken önceki adımlarda, atanmış katsayılar kümesini de korumak gereklidir. Şekil 2.7'de bu durum gösterilmektedir. Yeni düğüm ile beraber sisteme eklene katsayılar Şekil 2.7'de, H vektörü ile ifade edilmiştir. Eşitlikler, yeni gelen düğümün getirdiği katsayılarla birlikte, eski katsayılar kümesini de içermelidir. Böylece herhangi bir düğümde, orijinal giriş paketleri, denklem çözüm yöntemleriyle çözülebilir. Yeni gelen katsayılara birlikte, genişletilmiş denklemler aşağıdaki gibi olacaktır.

$X'_{j'} = \sum_{j=1}^{3} h_{j'j} X_j$; T düğümündeki j. çıkışa ait yeni denklem **(2.5)**

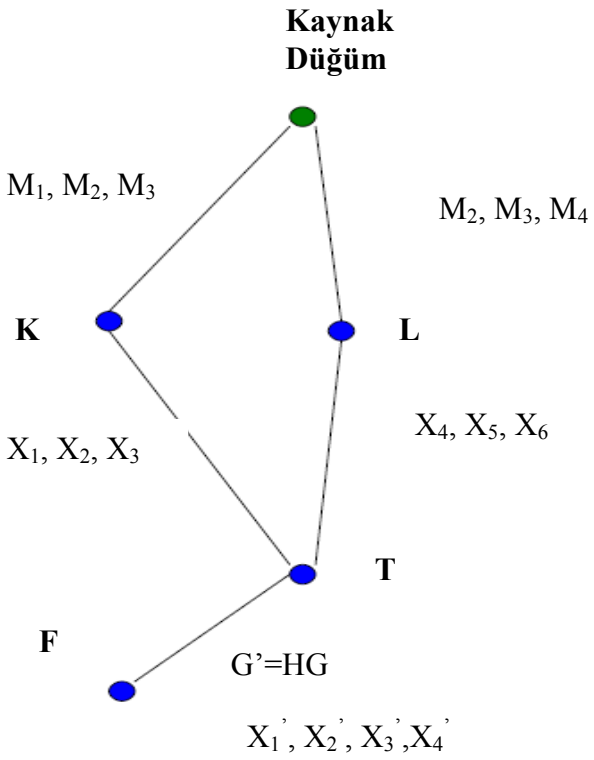
$g'_{j'i} = \left\langle h_{j'j} g_{ji} \right\rangle_{m \times n}$; T düğümündeki j. çıkışa ve S düğümündeki katsayılara bağlı yeni katsayı vektörü **(2.6)**

$X' = \sum_{j=1}^{4} h_{j'j} \sum_{i=1}^{4} g_{ij} M_i$; T düğümüne ait genişletilmiş lineer kodlama bilgi vektörü **(2.7)**

Şekil 2.7' de, yukarıdaki eşitlikler, sayısal örneklerle açıklanmıştır.

Şekil 2.7' de görüldüğü üzere, kaynak düğümden ağa gönderilen paketler M_1, M_2, M_3, M_4 paketleridir. K düğümünde X_1, X_2, X_3 çıkışları, L düğümünde ise X_4, X_5, X_6 çıkışları elde edilmektedir. T düğümü üzerinde, X_1, X_2, X_3 ve X_4, X_5, X_6 lineer birleşim işlemine alınmaktadır. Yeni lineer kodlama bilgi vektörü aşağıdaki gibi olmaktadır. $G' = H \cdot G$ şeklinde elde edilir.

$$X' = \begin{pmatrix} g'_{11} & g'_{12} & g'_{13} & g'_{14} \\ g'_{21} & g'_{22} & g'_{23} & g'_{24} \\ g'_{31} & g'_{32} & g'_{33} & g'_{34} \\ g'_{41} & g'_{42} & g'_{43} & g'_{44} \end{pmatrix} \cdot \begin{pmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{pmatrix} = \begin{pmatrix} g'_{11}M_1 + & g'_{12}M_2 + & g'_{13}M_3 + & g'_{14}M_4 \\ g'_{21}M_1 + & g'_{22}M_2 + & g'_{23}M_3 + & g'_{24}M_4 \\ g'_{31}M_1 + & g'_{32}M_2 + & g'_{33}M_3 + & g'_{34}M_4 \\ g'_{41}M_1 + & g'_{42}M_2 + & g'_{43}M_3 + & g'_{44}M_4 \end{pmatrix}$$



Şekil 2.7 : Eklenen düğüme ilişkin genişletilmiş denklemler

Bu matristen görüldüğü üzere, T düğümünde gerçekleşen lineer birleşim işlemi sonucunda, yalnızca 4 adet çıkış (X_1', X_2', X_3', X_4') ile orijinal bilgi paketlerini çözülebilir şekilde taşıyabilmektedir. Lineer ağ kodlama işlemi yapılmasaydı 6 adet çıkış paketi (X_1', X_2', X_3', X_4', X_5', X_6') iletilmek zorundaydı. Eğer ağ üzerindeki ayrıtların kapasitesi, 1 birim zamanda en fazla 4 paket iletecek şekilde, sınırlı olsaydı, kalan 2 paketin iletilmesi için, geleneksel yönlendirme metodunda, T düğümünde bu 2 paketi saklamak ve daha sonra iletmek gerekliydi. Bu durum ek hizmet süresi demektir. Burada hizmet süresi, orijinal paketlerin tamamının hedef düğüme ulaştırılması için gerekli süredir. Ağ kodlama yöntemi kullanılırsa hizmet süresindeki bu ek yük ortadan kalkacaktır. Ağ kodlamanın hizmet süresine faydası, böylece görülmektedir.

2.4.3 Lineer birleşimlerin Galois alanlarında gösterilimi

Ağ kodlama işlemi sonucunda elde edilen lineer denklemler, gerçek hayattaki bit düzeyinde işlemleri ifade etmektedir. Denklemlerin sayısal olarak, GF(2^8) alanında nasıl gösterildiği, aşağıda ifade edilmiştir.

$X_1 = 1 \cdot M_1 + 5 \cdot M_2$, $M_1 = 6$ ve $M_2 = 192$ olsun.

$1 \cdot M_1 = (00000001) \cdot (00000110) = (1) \cdot (Z^1 + Z^2) = 00000110 = 6$

$(Z^1 + Z^2) \bmod (1 + Z^2 + Z^3 + Z^4 + Z^8) = Z^1 + Z^2 = 6$

$5 \cdot M_2 = (000000101) \cdot (11000000) = (1 + Z^2) \cdot (Z^6 + Z^7) = Z^6 + Z^7 + Z^8 + Z^9$

$(Z^6 + Z^7 + Z^8 + Z^9) \bmod (1 + Z^2 + Z^3 + Z^4 + Z^8) = (1 + Z + Z^2 + Z^5 + Z^6 + Z^7) = 11100111 = 231$

$1 \cdot M_1 + 5 \cdot M_2 = 00000110 + 11100111 = 11100001 = 225$

X_1 değeri GF(2^8) alanında bit düzeyinde (11100001) olarak ifade edilebilmektedir.

2.4.4 Ağ kodlama ve rasgele katsayılar

Ağ kodlamanın önemli bir problemi, her bir düğüm için katsayıların nasıl seçileceğidir. Öyle denklemler oluşturulmalıdır ki , denklemlerin hiçbiri lineer bağımlı olmamalıdır. Ağdaki her düğümde kodlama aşamasında rasgele katsayılar seçilirse buna, buna rasgele ağ kodlaması denmektedir. Katsayı kümesini yeterince geniş tutarsak örneğin 2^8 elemanlı bir katsayı kümesi gibi, lineer bağımsız birleşimler elde etme olasılığımız oldukça yüksek olacaktır. Birbiriyle aynı birleşimler, yani aynı denklemler, elde etme olasılığımız böylece oldukça küçük olacaktır[15]. Aynı katsayının 2. kez seçilme olasılığı $1/2^8$' dir. Bu olasılık dikkate alındığında, sistemde lineer bağımlı denklemlerin oluşma olasılığı ihmal edilebilir düzeydedir.

Birbirinden farklı lineer birleşimlerin her biri, hedef düğümde oluşan denklem sisteminin çözümüne katkı sağlayacak, orijinal paketler hedef düğümde, hiçbir karışıklık yaşanmadan çözülebilecektir.

2.4.5 Lineer birleşimleri çözümleme

Ağ üzerindeki kodlama ve iletim işlemleri sonucunda hedef düğümün $(g_1, X_1) \ldots (g_m, X_m)$ elemanlarını aldığı varsayılsın. Orijinal paketi çözebilmek için $X_j = \sum_{i=1}^{n} g_{ij} M_i$ lineer denklem sistemini çözmemiz gerekmektedir. Çıkış düğümüne gelindiğinde, elde önceki katsayılar kümesi $(h_{1,\ldots,} h_j)$'yi de kapsayan ve $X' = \sum_{j=1}^{m} h_j \sum_{i=1}^{n} g_{ij} M_i$ şeklinde formüle edilebilecek m x n boyutlu ve çözülebilir bir denklem takımı olmalıdır.

Bu denklem sistemi n bilinmeyenli , m adet denklemden oluşmaktadır. Denklem sisteminin çözülebilmesi için $m \geq n$ olmalıdır. Bu da hedef düğümden alınan paket

sayısının, en az orjinal giriş paketleri sayısı kadar olması gerektiği anlamına gelmektedir.

Şekil 2.7'de de görüldüğü üzere, ağ üzerinde $M_1,...,M_n$ orijinal veri paketlerinin kendilerini taşımak yerine, bunların lineer birleşimini taşımak daha hızlı olacaktır. Ayrıca her bir düğüme ilişkin rasgele $g=(g_1,...,g_n)$ katsayılarını taşımak da, taşınan orijinal bilgi paketlerinin boyutları yanında ihmal edilebilir düzeydedir. Sonuç olarak günümüz ağ işlemleri için en önemli şey olan, hizmet süresinin kısa sürmesidir ve ağ kodlama bunu sağlamaktadır. Şekil 2.7'deki çıkış düğümündeki lineer denklem kümemiz $X^{'}=\sum_{j=1}^{m} h_{j'j} \sum_{i=1}^{n} g_{ij} M_i$, $m \geq n$ koşulu için aşağıdaki gibi 4 bilinmeyenli 4 denklemden oluşacak ve çözülebilir olacaktır.

$$X_1^{'} = g_{11}^{'} M_1 + g_{12}^{'} M_2 + g_{13}^{'} M_3 + g_{14}^{'} M_4$$
$$X_2^{'} = g_{21}^{'} M_1 + g_{22}^{'} M_2 + g_{23}^{'} M_3 + g_{24}^{'} M_4$$
$$X_3^{'} = g_{31}^{'} M_1 + g_{32}^{'} M_2 + g_{33}^{'} M_3 + g_{34}^{'} M_4$$
$$X_4^{'} = g_{41}^{'} M_1 + g_{42}^{'} M_2 + g_{43}^{'} M_3 + g_{44}^{'} M_4$$

Yukarıdaki denklem takımını çözmek için Gauss eliminasyon yöntemini çıkış düğümde uygulamak yeterli olacaktır.

2.4.6 Gauss eliminasyon yöntemi

Lineer birleşimler sonucu oluşan denklem takımları çözülürken, Gauss eliminasyon yönteminden faydalanılabilir. Ağ kodlama sonucu oluşmuş bir lineer denklem takımının çözümüne ilişkin, Gauss eliminasyon yöntemine ilişkin sayısal bir örnek, aşağıda belirtilmiştir.

Üç Bilinmeyenli üç denklemden oluşan sistem çözümü:

$$x_1 - x_2 + x_3 = 3$$
$$x_1 - 2x_2 = 0$$
$$x_2 - 3x_3 = -1$$

sisteminin asal ve genişletilmiş matrisleri

$$\mathrm{A} = \begin{pmatrix} 1 & -1 & 1 \\ 1 & -2 & 0 \\ 0 & 1 & -3 \end{pmatrix} \quad G = \begin{pmatrix} 1 & -1 & 1 & 3 \\ 1 & -2 & 0 & 0 \\ 0 & 1 & -3 & -1 \end{pmatrix}$$

1. ve 2. satır etalon matrisleri aşağıda gösterilmektedir.

$$S1 = \begin{pmatrix} 1 & -1 & 1 & 3 \\ 0 & -1 & -1 & -3 \\ 0 & 1 & -3 & -1 \end{pmatrix} \qquad S2 = \begin{pmatrix} 1 & -1 & 1 & 3 \\ 0 & -1 & -1 & -3 \\ 0 & 0 & -4 & -4 \end{pmatrix}$$

$$x_1 - x_2 + x_3 = 3$$
$$-x_2 - x_3 = -3$$
$$-4x_3 = -4$$
$$x_3 = 1 \ , x_2 = 2, \ x_1 = 4$$

Böylece 3 bilinmeyenli, 3 denklemden oluşan denklem takımı çözülmüş olacaktır.

Sonlu alanlar artimetiğinde çıkarma ve toplama işlemi , ayrıcalıklı veya işlemine karşılık gelmektedir. Ayrıca negatif sayılar bit düzeyindeki işlemlerde söz konusu değildir. Benzetim yazılımı hazırlanırken bu durum göz önüne alınmıştır.

2.5 Sonuç

Ağ kodlama işlemi "ayrıcalıklı veya" operatörü kullanılarak ya da lineer ağ kodlama yöntemi kullanılarak gerçekleştirilebilir. Bu bölümde görüldüğü üzere, düğüm sayısının az olduğu kelebek ağ modelinde "ayrıcalıklı veya" operatörünü kullanmak daha kolay, pratik olarak görülebilir. Fakat ağ içerisindeki düğüm sayısı, kaynak düğüm sayısı ve ayrıt sayısı arttıkça, Şekil 2.7' de görüldüğü üzere, lineer ağ kodlama yöntemi, katsayıları kullanmanın getirdiği avantajla, orijinal giriş paketlerini, birbirinden farklı denklem takımları şeklinde, hedef düğümlere, çözülebilir olarak taşıyabilmektedir. Katsayılar kümesi, orijinal paketlerin işaretleyicisi gibi kullanılmaktadır. Böylece hedef düğümlerde, ağ kodlaması uygulanmış paketler basit denklem çözme metodlarıyla çözülebilmektedir. Yine Şekil 2.7' de görüldüğü üzere hizmet süresinde de önemli ölçüde kısalma meydana gelmektedir.

3. TELSİZ ÖRGÜ AĞLAR, P2P DOSYA PAYLAŞIMI VE TELSİZ ÖRGÜ AĞLARDA AĞ KODLAMA

Bu çalışmada, telsiz yerel alan ağ üzerinde, örgü ağ topolojisi ve dosya paylaşımına ilişkin uygulamalar gerçekleştirilmektedir. Dolayısıyla bu bölümde, telsiz yerel alan ağları ve telsiz örgü ağlara ilişkin açıklamalar bulunmaktadır. Ayrıca, tezin değindiği diğer bir konu olan, P2P dosya paylaşımı uygulamaları da yine bu bölümde anlatılacaktır.

3.1 Yerel Alan Ağları (LAN)

Yerel alan ağları (LAN) bir bina, okul, hastane, kampus gibi sınırlı bir coğrafi alanda kurulan ve çok sayıda kişisel bilgisayarın yer aldığı ağlardır. LAN, ilk başlarda bir sunucunun birkaç terminale telli olarak bağlandığı küçük sistemlerden ibaretti. Günümüzde ise LAN telli ve telsiz olarak birçok sunucu ve istemciyi birbirine bağlayabilmektedir. LAN, yüksek hızları destekleyen yüksek verimlilikte ağlar haline dönüşmüş ve geleneksel veri işlemenin yansıra ses ve video-konferans gibi işlevleri destekleyen ağlar haline gelmeye başlamıştır. Yerel alan ağlarında iletişim, genellikle bir bina veya binalar grubu, hastane, fabrika, üniversite kampüsü ve benzeri alanlar ile sınırlıdır.

3.1.1 Telsiz yerel alan ağları (WLAN)

LAN'larda bilgisayarlar ve ağ içerisindeki diğer cihazlar arasında iletişimi sağlamak üzere kablo yerine RF veya kızılötesi teknolojisi kullanılması durumunda, telsiz yerel alan ağları (WLAN) olarak adlandırılmaktadır. Telsiz yerel alan ağları, kablolu LAN'ların tüm özelliklerine sahiptir. WLAN sistemleri; kullanıcılarına kablosuz geniş bant internet erişimi, sunucu üzerindeki uygulamalara ulaşım, aynı ağa bağlı kullanıcılar arasında elektronik posta hizmeti ve dosya paylaşımı gibi çeşitli imkanlar sağlamaktadır. Ayrıca kablosuz bir sistem olması nedeniyle cadde, sokak, park, bahçe ve benzeri açık alanlarda WLAN sistemleri başarılı bir şekilde kullanılmaktadır. Ancak yerel kullanım amacıyla geliştirildiğinden, WLAN

sistemlerinin mesafesi 25-100 metre civarındadır. Yerel alan ağlarında IEEE 802.11x standardı yaygın olarak kullanılmaktadır.

3.1.1.1 IEEE 802.11x

WLAN uygulamalarında en çok kullanılan ve bugünkü popülerliğini kazandıran IEEE tarafından yayınlanan bir dizi standarttır. IEEE 802 LAN/MAN standart komitesi ilk olarak Haziran 1997'de IEEE 802.11 standardını yayınlamıştır. Bu temel standarda göre 2.4 GHz frekans bandında FHSS veya DSSS teknikleri kullanılarak 2 Mbps'e kadar data iletişimi sağlanabilmektedir. 802.11 standardın esas amacı mevcut kablolu LAN'ların, kablosuz olarak genişlemesine olanak tanımak ve sabit sistemlerle mobil sistemleri bir çatı altında toplamaktır. Elde edilen başarı sonrasında IEEE tarafından WLAN uygulamaları için 802.11x adı altında bir dizi standard daha yayımlanmıştır. Günümüzde, veri iletişimini 54 Mbps'e kadar çıkaran 802.11g standardı cihazlar rağbet görmektedir.

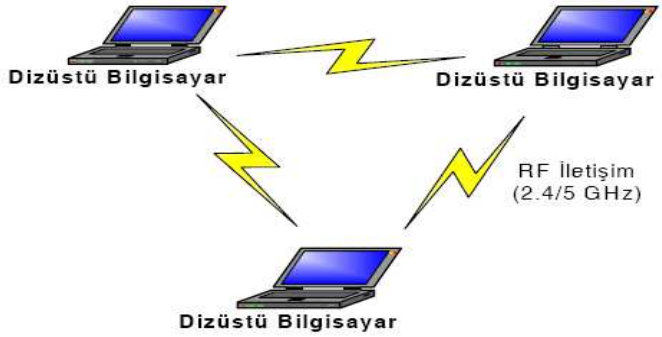
3.1.2 Yerel alan ağı işletim türleri

Telsiz yerel alan ağları, ağ üzerinde bilgisayarların nasıl yapılandırıldığına ve bu bilgisayarların bilgilere nasıl erişitiğine göre ikiye ayrılır:

-Eşler arası ağ (peer-to-peer)

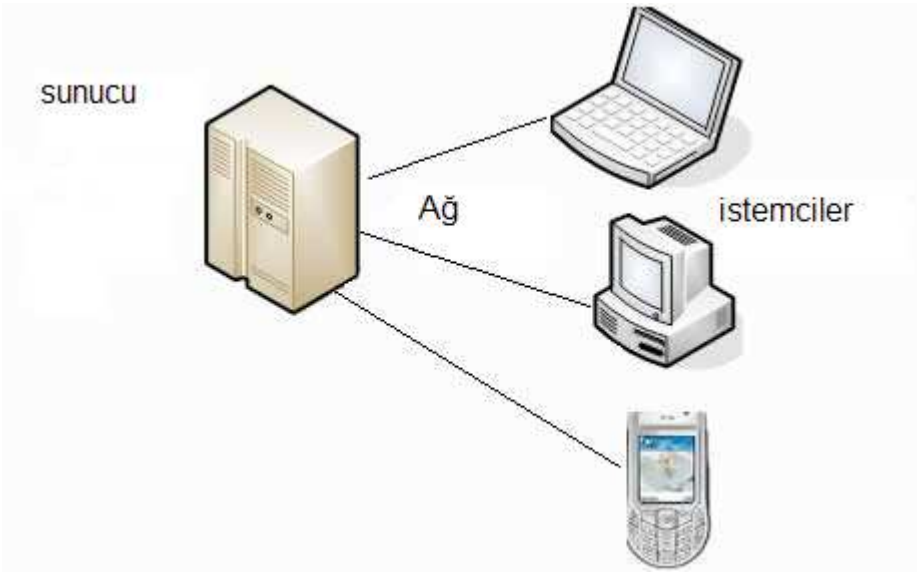
-Sunucu tabanlı ağ

Eşler-arası telsiz yerel ağlarda, sınırlı sayıda kullanıcı birbirine bağlıdır. Bu kullanıcılar düzey olarak aynıdır. Yani içlerinden birisinin merkezi sunucu olarak kullanılması söz konusu değildir. Bir bağlantı aracılığıyla isteyen kullanıcılar birbirleriyle iletişim kurar ya da dosya alışverişi yapabilirler. Bu tür ağlarda bulunan bilgisayarların sahip oldukları program, veri, dosya gibi tüm kaynaklar ağdaki diğer bilgisayarlar tarafından kullanılabilir. Eşler arası ağ , prensip olarak hızlı kurulabilen, telli veya telsiz erişim noktası (Access Point - AP) gibi herhangi bir altyapı gerektirmeyen basit kuruluma sahiptir. İstemci veya sunucu olmasına bakılmaksızın ağda yer alan tüm bilgisayarlarda sadece telsiz çalışma özelliğinin olması yeterlidir. Ayrıca telsiz yönlendiriciler de kendi aralarında eşler arası bağlantı kurabilmektedirler. Böylece kapsama alanlarını genişletebilmektedirler. Şekil 3.1'de eşler arası ağ örneği verilmiştir.



Şekil 3.1 : Eşler arası ağ

Sunucu tabanlı ağlarda bir ana bilgisayar vardır. Bu ana bilgisayara, sunucu denir. Sunucu üzerinde ağ yönetimi yapılır. Ayrıca, ağa kaydolacak ya da bağlanacak tüm istemciler, bu ana bilgisayar üzerinde yer alan kullanıcı hesaplarına göre kontrol edilir ve bağlantıları gerçekleştirilir. Bu ağlar telli veya telsiz erişime sahip olabilir. Şekil 3.2'te sunucu tabanlı ağ yapısı gösterilmiştir.

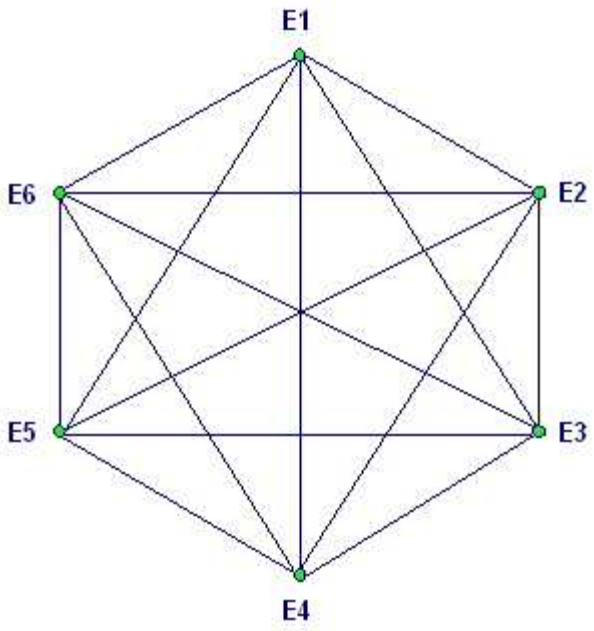


Şekil 3.2 : Sunucu tabanlı ağ

3.2 Örgü Ağlar

Ağ ögelerinin, birbirlerine birden fazla ayrıt (link) ile bağlı olduğu ağlardır. Telli veya telsiz oluşturulabilirler. İletilmek istenen bilgi yönlendiriciden yönlendiriciye iletilir. Yönlendiricilerin her biri, birbiriyle birebir veya dolaylı iletişim halindedir. Yönlendiriciler yerel alan ağlar içerisinde dinamik yönlendirme işlemini gerçekleştirir. Yönlendiriciler arası yol atama işlemi devamlı güncellenir. Bilginin

yönlendiriciden yönlendiriciye iletilmesi işine atlama (hop) adı verilir. Bu işlem bir düğümdeki bilgilerin diğer düğümlere aktarılmasını sağlamaktadır. Gezgin tasarsız ağlar dışındaki tüm örgü ağ tiplerinde, yönlendiriciler sabittir.Şekil 3.3'te tam bağlı örgü ağ topoloji görülmektedir Bu topolojide tüm ağ elemanları birbirleriyle, aktarma işlemi olmaksızın, birebir iletişim kurabilmektedir.



Şekil 3.3 : Tam bağlı örgü ağ

3.2.1 Telsiz örgü ağlar

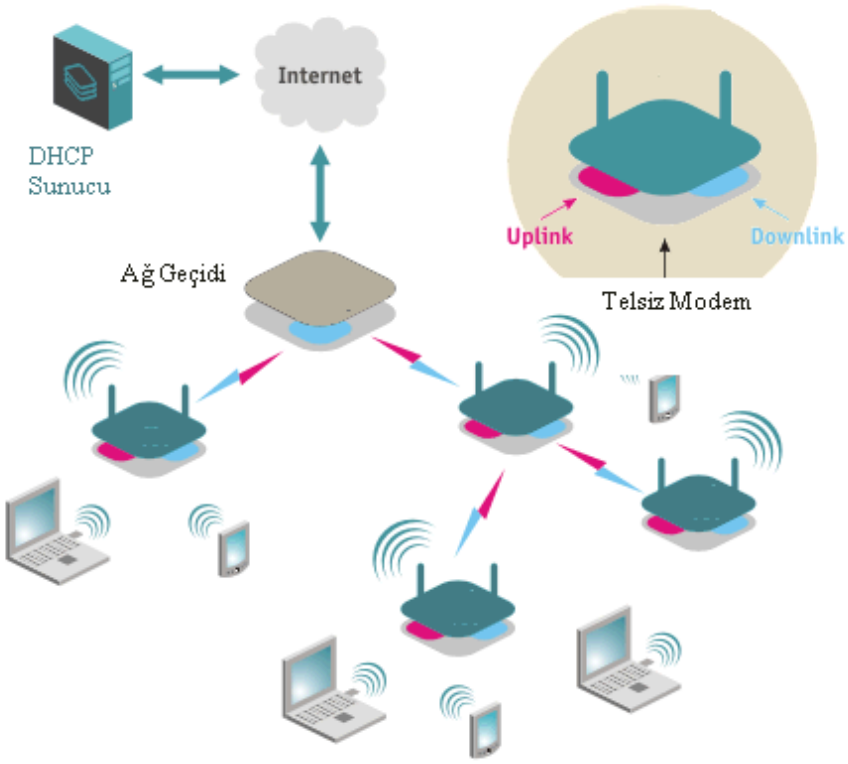
Telsiz ağların, örgü ağlar şeklinde organize olduğu topolojidir. Telsiz yerel alan ağlarda, sıkça kullanılan bir ağ çeşididir. İşletim türlerine göre, eşler arası ağ yapısındadırlar, çünkü baz istasyonu vb. benzeri bir merkezi telsiz sunucuları yoktur. Bu tip ağlar, örgü ağ yönlendirici ve örgü ağ istemcisinden oluşur. Örgü ağ yönlendiriciler, ethernet telsiz yönlendirici benzeri aygıtlardır. Günümüzde kullanılan telsiz modemler, örgü ağ yönlendiricilerine güzel bir örnektir. Wi-fi bağlantısı bulunan cep telefonları ve bilgisayarlar da örgü ağ istemcisine örnek teşkil etmektedir. Birçok örgü yönlendirici, birbirleriyle telsiz bağlantı kurup örgü topoloji oluştururlar. Örgü topolojide, birbiriyle bağlantı kurabilen yönlendiriciler bulunur. Bu da örgü ağın omurgasını oluşturur. Eşler arası bağ kurulur. Ağdaki tüm elenamlar birbirleriyle iletişim halindedir. Yönlendiriciler dağınık şekilde organize oldukları halde veri akışı sağlayabilmektedir.

Tasarsız örgü ağın özelliklerini detaylı olarak ele alırsak , bir yönlendirici , kendi kapsama alanında olmayan , yani direk iletişim kuramadığı uzak bir yönlendirici ile, aradaki yönlendiriciler vasıtasıyla iletişim kurabilmektedir. Yani bir düğüm, diğer düğüme aktarma yapmaktadır. Bu işlem, hedef örgü yönlendiriciye, bilgi ulaşana kadar devam etmektedir.

Telsiz örgü ağlar kendiliğinden yapılanma özelliğine de sahiptir. Ağın denetimi, yönetimi ve yol atama için merkezi bir otorite yoktur. Telsiz yönlendiricilerin her biri gerektiğinde yol atama fonksiyonunu yerine getirirler.

Telsiz örgü ağlarda, örgü yönlendiricilerin bazıları, ağ geçidi görevi görüp internete de bağlanabilmektedir. Örgü ağ yönlendiricisine bağlanan örgü istemcilerine örnek olarak, yazıcı, bilgisayarlar, cep telefonları verilebilir. Şekil 3.4'te bir telsiz örgü ağ örneği verilmiştir. Şekilde örgü yönlendiricilerin ve örgü istemcilerin, gerçek hayatta karşımıza çıkan örnekleri gösterilmiştir. Telsiz modemler, örgü yönlendiricilere örnek teşkil etmektedir. Bilgisayar ve wi-fi özellikli cep telefonları ise örgü yönlendiricilere örnektir. Telsiz yönlendiriciler de kendi aralarında eşler arası bağlantı kurabilmektedirler. Wi-fi özelliği bulunan bilgisayarlar da hem örgü isteci hem de örgü yönlendirici özelliği gösterilebilirler ve kendi aralarında telsiz örgü ağ kurabilirler. Böylece kapsama alanlarını genişletebilmektedirler. Bu durum örgü ağ topolojisi sayesinde gerçekleşmektedir[17].

Şekil 3.4'te görüldüğü üzere örgü istemciler, örgü istemcilerle veri iletişimi gerçekleştirmekte, örgü yönlendiriciler de bu iletişimin sonucunda başarılması istenen uygulamanın özelliğine göre, ya kendisine bağlı olan istemcileri internete bağlamak için bir ağ geçidine bağlanmaya çalışacak ya da diğer örgü yönlendiricilerle kendi aralarında yine örgü topolojiye uygun veri iletişimi, dosya paylaşımı yapmaya çalışacaktır. Ağ geçidi üzerinden gerçekleşen internet bağlantısı da, uzak bir merkezdeki sunucu tarafından kontrol edilmektedir. Buradaki DHCP sunucu, telsiz örgü ağın bir parçası olmayıp, genel internet omurgasının bir parçasıdır.



Şekil 3.4 : Telsiz örgü ağ

3.3 P2P Dosya Paylaşım Uygulamaları

P2P (peer-to-peer) bilgisayar kaynaklarının karşılıklı olarak paylaşım yapmasına verilen isimdir. Kaynak paylaşımı, bilgi teknolojilerinin temel kullanım amacıdır. P2P dosya paylaşımına hız getirmektedir. Çünkü paylaşım işlemi merkezi bir sunucudan değil, birbirinden bağımsız bilgisayar kaynaklarından yapılmaktadır.

P2P protokolü uygulamalarının temeli, ilk olarak 1999 yılında Shawn Fanning tarafından ortaya çıkarılan, Napster dosya paylaşım programı ile oluşturulmuştur. Bu ücretsiz paylaşım programı, ilk önce bir takım medya dosyalarının paylaşımı için ortaya konulmuştur. Kısa sürede milyonlarca kişi tarafından kullanılmaya başlanmıştır.

Uygulamanın çalışma mantığı şöyledir. Napster sunucusu, Napster yazılımını bilgisayarlarında çalıştıran tüm kullanıcıların paylaşıma açtığı dosyaların listelerini tutmaktadır. Napster yazılımı aracılığıyla arama yapan bir istemcinin talebi, Napster protokolüne ait kontrol paketleri kullanılarak Napster sunucusuna ulaşmaktaydı. Sunucu, aranan dosya ismini, tuttuğu listelerde aramaktaydı. Eğer dosyayı paylaşıma açmış bir kullanıcı varsa, listeden elde edilen sonuca göre bu kullanıcının IP adresi,

istemciye, yine kontrol paketleri aracılığıyla iletilmektedir. Bir sonraki aşamada, istemci IP adresini bildiği kullanıcıdan istediği dosyayı " parçalara ayrılmış paketler" halinde indirmeye başlamaktadır.

İlerleyen yıllarda, Napster dosya paylaşım programı birtakım yasal prosedürleri yerine getirmediği gerekçesiyle durdurulmuştur. Fakat P2P dosya paylaşımı uygulamaları gitgide yaygınlaşmış, hala da yaygın şekilde kullanılmaktadır.

Bir sonraki nesil olarak ortaya konulan Gnutella protokolü, Napster'dan farklı olarak tek sunuculu yapı yerine, her istemcinin aynı zamanda bir sunucu olduğu yapıyı ortaya koymuştur. Günümüzde LimeWare dosya paylaşım yazılımında, Gnutella ve P2P dosya paylaşım protokolü temelleri kullanılmaktadır. Gnutella protokolü, hızlı dosya paylaşım ortamı ve geniş bir veritabanı sağlaması açısından, bilimsel yayınlarda da kendisine yer bulmuştur. Geleceğin ağları ile ilgili bilimsel yayınlarda kendilerine yer bulan konulardan biri olmuştur.

3.3.1 GNUTELLA dosya paylaşım protokolü

Gnutella, günümüzde dosya paylaşım uygulamaları tarafında kullanılmakta olan bir dosya paylaştırma protokolüdür. Herhangi bir Gnutella uygulama yazılımını (örneğin LimeWare, Morpheus, iMesh vb.) bilgisayarında kullanan bir kullanıcı, yazılımı çalıştırdığında otomatik olarak bir sunucuya bağlanmaktadır. Bu sunucunun IP adresi, yazılımı programlayanlar tarafından başka bir sunucu bilgisi bilinmediğinde de paylaşımın sağlanabilmesi amacıyla yazılım içine gömülmüştür. Kullanıcı, istediği takdirde, bildiği başka Gnutella sunucu IP adreslerini yazılım veri tabanına ekleyebilmektedir. Yazılımı kullanarak dosya araması yapan istemcinin sorgusu, ilk olarak istemcinin bağlı olduğu sunuculara aktarılmaktadır. Sorguyu alan sunucu, kendisinin ve kendisine direk bağlı diğer kullanıcıların paylaşıma açtığı dosya listesinde arama yapılmaktadır. Bir sonuca ulaşıldığında, dosya sahibinin IP adresi istemciye gönderilmektedir. Eğer arama başarısız olursa, sorgu sunucu tarafından kendisine direk bağlı olan tüm kullanıcılara aktarılmaktadır. Daha önce yönlendiricilerde kullanılan ve "gossip" (dedikodu) olarak anılan bu yönlendirme yöntemi sayesinde sorgu ağ üzerinde kademe kademe ilerleyebilmektedir. Sorgunun, bir sunucudan, bu sunucuyla bağlantılı sunuculara aktarılma islemi sırasında istemci tarafından belirlenen, yönlendirme esnasında sorgu paketin adım sayısını belirten, TTL (Time-to-Live) değeri bir kademe düşürülmektedir. Bu değer sıfıra ulaştığında,

sorgu sona erdirilerek sonsuz döngünün önüne geçilmektedir. Dosya bulunanamış varsayılacak, dosya paylaşım işlemi sona erdirilecektir. Arama sırasında her kullanıcının hem istemci hem de sunucu seklinde davranıyor olması, P2P (eşler arası) iletişiminin, geleneksel sunucu istemci iletişiminden en önemli farkıdır [10]. Napster'da bu fark sadece "veri paketleri iletişiminde" ortaya çıksa da Gnutella ile birlikte kontrol paketleri iletişimi de tüm kullanıcılar arasında yapılmaya başlanmıştır.

Gnutella protokolü temelli dosya paylaşım uygulamaları, yönlendirme için gossip (dedikodu) metodunu kullanmaktadır. Gossip yönlendirme yöntemi, herbir düğümün, çevresindeki komşu düğümlerden rasgele olarak seçtiği bir düğüme, kendisine gelen paketi iletmesi esasına dayanır. Bu işlemi gerçekleştirmeden önce orijinal veri paketi, bir splitter(bölümleyici) aracılığıyla parçalara ayrılmaktadır. Bir düğüm kaynaktan gelen bir dosyanın bir parçasını alınca, onu tüm komşu rasgele seçtiği bir komşu düğüme iletmektedir. Her bir düğüm, bir dosya parçasını aldığı anda o parçayı, çevresindeki düğümlerden rasgele bir tanesini seçerek iletmektedir. Gossip yönlendirme protokolü günümüz uygulamalarında hem dosya paylaşımı hem de indirilmek istenen dosyaya sahip kaynak düğümün ağ üzerinde sorgulanması işlemi için kullanılabilmektedir[11].

Gnutella dosya paylaşım uygulamaları, gossip yönlendirme yönteminden iki işlemi gerçekleştirirken faydalanmaktadır. Bu işlemler sırasıyla, indirilmek istenen dosyaya sahip kaynak düğümün ağ üzerinde sorgulanması ve sorgulama sonrasında kaynak düğümü bulunmuş dosyanın ağ düğümleri üzerinde paylaşımı işlemleridir. Bu çalışmada, gossip yönlendirme yöntemi, sorgulama sonrasında indirilmek istenen dosyaya ait kaynak düğümün bulunmuş olduğu duruma için kullanılmaktadır.

3.3.2 Dosya paylaşım uygulamalarına ağ kodlamanın getirdiği faydalar

Geniş band internetin yaygınlaşmasıyla birlikte kullanım oranı artan eşler arası (P2P) dosya paylaşımı için, bilgisayar programları tasarlanmıştır. İlk zamanlarda yalnızca medya dosyalarını paylaşmak amacıyla oluşturulan bu programlar, internetin hızla yaygınlaşması nedeniyle her türden program ve dosyanın temin edilebildiği kaynaklar haline gelmiştir. Dosya paylaşımı gerçekleştiren programların günümüzde kullanılan en önemli örnekleri dc++, Limewire, Ares, Morpheus. Kazaa ve iMesh'dir.

Dosya paylaşımı yapan programlar, paylaşım işlerini çok farklı yöntemlerle yerine getirebilmektedir. Limeware gibi programlarda paylaşılmak istenen dosya çok küçük parçalara ayrılmaktadır ve farklı parçalar farklı kullanıcılardan indirilmektedir. Böylece parçalar halinde indirilen dosya hafıza birimlerinde tekrar birleştirilmekte ve dosya yeniden elde edilmiş olmaktadır.

Ağ kodlaması birçok dosya parçasını birleştirerek, ağ üzerinde taşımaktadır. Bu işlem sayesinde, ağ üzerinde nadir bulunan ve yeterince popüler olmayan dosya parçaları bile, bu birleşim içinde yer aldığından ilgili düğümlere hızlıca iletilmiş olacaktır. Böylece eğer dosyanın belli bir yüzdesi indirilmişse, diğer parçaların beklenmesine gerek kalmadan yeniden oluşturulabilmektedir. Bu da %99'u inmiş bir dosyayı tümüyle yüklemek için haftalarca bekleyen kullanıcıların sorununu ortadan kaldırmaktadır. Böylece paylaşım programlarında görülen, az sayıda kullanıcıda bulunan parçaların çok uzun süre beklenmesi gibi problemi ise ağ kodlama yöntemi ile çözülmektedir. [18].

Gkantisidis, çalışmasında eşler arası erişim sağlanan telli ağlarda geniş içerikli bilgini dağıtımını anlatırken, dağıtım süresinin ağ kodlaması kullanıldığı zaman daha hızlı olduğunu gösterdi [6]. Eşler arası erişilen ağlarda, kaynak dosyada en az olasılıkla bulunan olan bilgiyi barındıran dosya parçasının iletimi, dosya paylaşımı süresinin belirlenmesinde önemli etkendir.

3.3.3 Telsiz örgü ağlarda ağ kodlamanın getirdiği faydalar

Telsiz örgü ağlarda iki telsiz yönlendirici, telsiz kapsama alanları yeterli olduğu ölçüde birbirebir haberleşmektedir. Fakat kapsama alanları yetmiyorsa, kendilerine yakın diğer düğümler aracılığıyla veri transferlerini gerçekleştirmektedirler. Dosya paylaşım uygulamalarında ise ana fikir, bir kaynak veri dosyasının belli sayıda kullanıcı tarafından ağdan çekilmesidir. Ağ kodlaması uygulamaları da, kaynak düğümlerden gelen veriyi, lineer birleşim işlemi aracılığıyla, tüm düğümlere geleneksel yönlendirme metodlarına nazaran daha hızlı bir şekilde iletmek için tasarlanmıştır. Gerek dosya paylaşımı uygulamalarının kullanılma yoğunluğu gerekse de ağ kodlamasının hizmet süresine olumlu etkisi, telsiz örgü ağların, ağ kodlama yönteminin uygulanması için uygun bir araştırma alanı olduğunu göstermektedir [8]. Örgü ağlarda, ağ kodlamasının faydalı bir şekilde kullanılması dosya alış-verişi, veritabanı güncellenmesi, yeni yazılımların yamalarının internet

üzerinden kolayca ve hızlıca entegre edilmesinde faydalar sağlamaktadır. Ayrıca firmware olarak adlandırılan, telsiz iletişim cihazlarının donanımsal yükleri de ağ kodlama sayesinde hızlıca güncellenebilir.

Üzerinde bir dosya paylaşımı uygulaması koşan telsiz örgüsel ağda, ağ kodlamasının bize sağladığı faydalar oldukça fazladır. Ağ kodlama hizmet süresi açısından verimliliği arttıran bir yöntemdir. Ağ kodlamanın, telsiz örgü ağlar için de ağ performansını hizmet süresi açısından arttıracağı öngörülmektedir [5,8].

3.4 Sonuç

Telsiz örgü ağlar günümüzde yaygın bir kullanım alanına sahiptir ve daha da yaygın olma yolunda ilerlemektedirler. Aynı şekilde dosya paylaşım uygulamaları da, günümüzde hızlı dosya paylaşımı sağlamalarından dolayı önemli ve yaygındırlar. Özellikle Gnutella tabanlı uygulamalar, merkezi bir sunucu gerektirmeyen paylaşım sistemiyle, telsiz örgü ağlarda uygulanmak için oldukça uygundur. Günümüzde bir telsiz yönlendiriciye USB bellek birimi takılabilmekte, bu belleğin üzerindeki dosyaları, telsiz örgü ağ üzerinde paylaşabilmemizi sağlayan entegre cihazlar dahi çıkmış durumdadır. Bu ortamda dosya paylaşımı uygulamalarını kullanmak, veri iletişimini hızlandırmaktadır. Ağ kodlama yönteminin uygulanması, gelecekte P2P dosya paylaşım işlemini daha da hızlı hale getireceği öngörülmektedir.

Bu çalışmada, gossip yönlendirme yöntemi, kaynak düğümü bulunmuş olan dosyanın paylaştırılmasına ilişkin uygulamalar gerçekleştirilmektedir. Ağ kodlama yönteminin de uygulanmasıyla, hizmet süresinde kısalma olacağı gösterilmektedir.

4. BENZETİMLER

Bu bölümde telsiz örgü ağ ortamındaki dosya paylaşımı üzerinde ağ kodlama uygulamalarının benzetim yazılımı tarafından, bu çalışmada nasıl gerçekleştirildiği anlatılmaktadır. Dosya paylaşımında gossip yönlendirme yöntemi kullanılmaktadır. C++ programlama diliyle yazılan benzetim yazılımı ile elde edilen sonuçlar, MATLAB programı aracılığıyla çizdirilmiş, sonuçlar yorumlanmıştır. Ağ kodlamanın, hizmet süresinde getirdiği avantajlar ortaya konulmuştur. Bu çalışmada yazılımını gerçekleştirdiğim uygulama, gerçek hayatta telekomünikasyon operatörlerinin, güncel yama yazılımlarını, eşler arası ağ düzeninde işletilen telsiz örgü ağ elemanlarına, daha hızlı bir şekilde paylaştırmalarını sağlamakta yardımcı olmayı amaçlamaktadır.

4.1 Gossip Yönlendirme Yöntemi ve Telsiz Örgü Ağlarda Ağ Kodlama

Gossip yönlendirme protokolü dosya paylaşımında günümüzde çok sık kullanılan bir protokoldür. Gnutella ve Limeware yazılımları bu yönledirme protokolünü kullanmaktadır. Ayrıca MANET ve VANET uygulamalarına ilişkin son yıllardaki çalışmalarda, gezgin iletişim cihazları üzerinde araçlar üzerinde P2P dosya paylaşımı uygulanması deneylerinde, gossip yönlendirme protokolü sıklıkla kullanılmaktadır. Geleceğin tasarsız ağlarında dosya paylaşımına ilişkin yapılan çalışmalar da, bu yönde tüm hızıyla devam etmektedir[13,19]. Geçmişte telsiz örgü ağ üzerinde, taşmalar şeklinde (flooding) yönlendirme yöntemi kullanılarak dosya paylaşımına ilişkin çeşitli uygulamalar gerçekleştirilmiştir[8].

Bu çalışmada, gossip yönlendirme yöntemi kullanılmaktadır. Ağ kodlama yönteminin, telsiz örgü ağlarda gossip yöntemi kullanılan dosya paylaşımında, hizmet süresine etkisi incelenmektedir.

Paylaşılmaya başlanan dosya ilk önce eşit parçalara ayrılmalıdır. Bu işleme splitting denilmektedir. Eşit uzunluktaki dosya parçaları, bir kaynak düğüm üzerinden hedef düğümlere gönderilmek istenmektedir. Ağ kodlama yöntemi, bu çalışmada, bu noktada devreye girmektedir. Ağ üzerindeki her bir düğüm, kendisine gelen bu veri

paketlerini lineer birleşim ile denklem takımlarına dönüştürmekte, daha sonra Gossip yönlendirme yöntemine uygun olarak, kapsamı alanındaki düğümlerden birini rasgele seçerek, sıradaki düğüme iletmektedir. Her bir alıcı düğüm, bir lineer birleşimi alacak, aldığı bu bilgiyi yeni bir lineer birleşim haline getirerek, diğer bir düğüme iletecektir. Aynı zamanda, bu birleşimlerin oluşturduğu denklem sisteminin çözülebilmesi için ihmal edilebilir boyutlarda bir hafıza alanı kapsayacak olan rastgele oluşturulmuş katsayılar da, orijinal paketlerin yanında düğümden düğüme taşınmaktadır. Lineer birleşim şeklinde taşınmış orjinal paketler, hedef düğümlere çözülebilir denklem takımları şekline ulaştırılmaktadır.

Telsiz örgü ağlarda, her bir örgü yönlendirici, kendi kapsamı alanındaki diğer yönlendiricilerin çalışır durumda olup olmadığı hakkında bilgi sahibidir. Telsiz örgü yönlendiricilerin her biri, graf gösteriliminde bir düğüme tekabül etmektedir. Telsiz örgü ağlar, bilgi aktarma işlemi (hop) aracılığıyla, kendi komşularını kullanarak, kapsamı alanında bulunmayan hedef düğümlere veri iletimini gerçekleştirebilmektedir. Her bir düğüm GPS aracılığıyla kendi konum bilgisini öğrenip, bunu diğer düğümlere kontrol paketi şeklinde iletebilir. Böylece ağ üzerindeki düğümler diğer düğümlerin konumları hakkında bilgi sahibi olmaktadır [20].

Bu çalışmada, konum bilgisi hesaplamaları ve komşuluk ilişkileri, yazılım ile yaratılacak sanal bir tablo üzerinde belirtilecektir. Çünkü fiziksel bir deney düzeneği kurulmayacaktır, ağ kodlama işleminin faydası, bu çalışma için özel olarak hazırlanmış benzetim yazılımı aracılığı ile gösterilecektir. Ağ üzerindeki bilgi akışı sırasında her bir düğüm bu sanal tabloyu kontrol etmekte, bu doğrultuda, kendisine gelen bilgi paketlerini diğer düğümlere iletmektedir. Burada aynı kapsam alanı içerisinde veri paketi alan düğüm, aynı anda veri paketi iletimi yapamamaktadır. Bu durum, bu çalışmada benzetim yazılımı tarafından sanal tablolar aracılığı ile kontrol edilmiştir.

İlk benzetimler, sadece telsiz örgü ağlarda gossip yönlendirme protokolü ile dosya paylaşımı yapıldığı durum için gerçekleştirilmektedir. Ardından aynı duruma, bir de ağ kodlama yöntemi eklenerek, telsiz örgü ağlarda dosya paylaşımına ilişkin benzetimler gerçekleştirilmektedir. Böylece karşılaştırılabilir sonuçlar elde edilmektedir.

Bu çalışmadaki benzetimin, gerçek hayattaki karşılığının, bir mobil operatörün yama yazılımlarını, sahada bulunan, eşler arası işletim özelliğine sahip bir telsiz örgü ağ üzerindeki ağ elemanlarına dağıtması durumudur. Ağ üzerinde elemanların hepsi hem istemci hem de sunucu özelliği göstermektedir. Tüm düğümler, tüm yama yazılımları yüklemeyi tamamlayana kadar benzetim yazılımı çalışmaya devam etmektedir. Bu sırada her bir düğüm, kendi yüklediği yama yazılım parçalarını kaydetmekte ve sanki birer kaynak düğüm gibi davranarak, kendisine komşu olan düğümlere, sahip olduğu dosya parçalarını gossip yönlendirme yöntemi ile iletmektedir.

4.2 Benzetim Ortamı

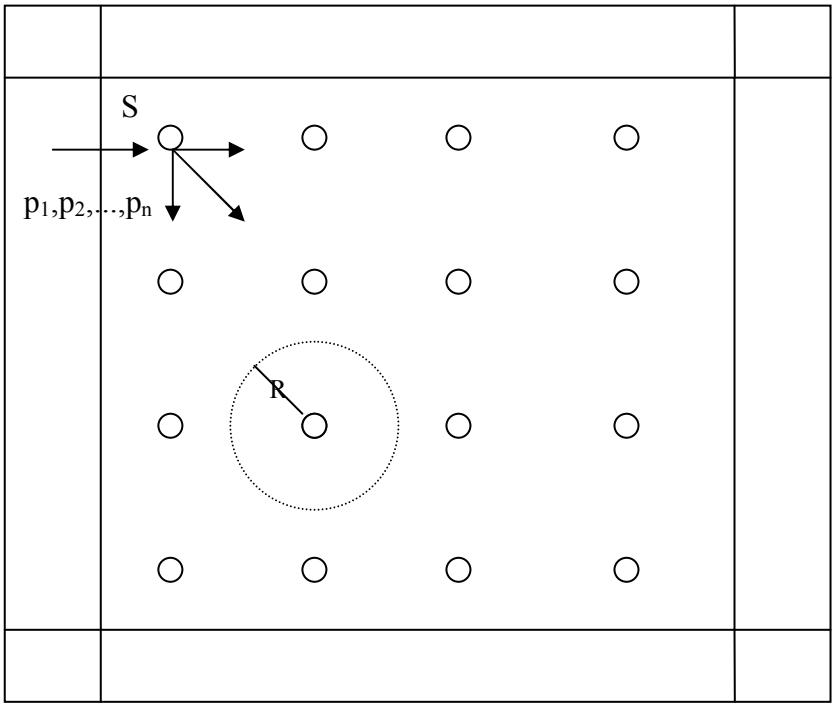
Telsiz ağları, 2 boyutlu bir alanı rasgele dağılmış olarak kapsamaya çalışan çemberler kümesi olarak düşünebiliriz. S alanına sahip bir kare düzlemde, nokta-alan yoğunluğu D olan ağ düğümlerinin, 2 boyutlu düzlemde Poisson sürecine uygun olarak birbirleriyle iletişim kurduğunu söyleyebiliriz. Bu düzlem üzerindeki ağ düğümleri Poisson noktaları olarak adlandırılırsa, pratik olarak, çember yarıçapı R'yi, o düğümün veri iletişimi yapabildiği düğüm sayısı , kapsama alanı, olarak adlandırabiliriz. [21]. Şekil 4.1'de bu duruma örnek olabilecek bir telsiz örgü ağ örneği verilmiştir. Buradaki Poisson noktaları gerçek hayatta baz istasyonları ve telsiz modemlere, yani telsiz örgü yönlendiricilere veya tekrarlayıcılara karşılık gelmektedirler. Şekil 4.1'de yer alan telsiz örgü ağ, S alanlı kare düzlemde dağılım göstermektedir. Bu kare üzerinde telsiz örgü ağ düğümleri rasgele dağılabilir. Bu çalışmada, R parametresi bir düğümün kendi kapsama alanı içerisinde sahip olduğu komşu düğüm sayısını ifade etmektedir. Bu parametre, bu çalışmada oluşturulan benzetim yazlımında, en az 2 en çok ise 3 değerini almaktadır.

Ağ üzerinde kapsanmayan düğüm olması problemiyle karşılaşılmamak için , N ağ üzerindeki düğüm sayısı olmak üzere , N, R yarıçapı ve S alanı arasında bir ilişki bulunmaktadır. Bu ilişki

$$N \geq \frac{10 \cdot S}{\pi \cdot R^2} \qquad \textbf{(5.1)}$$

şeklindedir [21]. Bu ilişki sağlanırsa tüm düğümler kapsanmaktadır. Bu çalışmadaki benzetimlerde, kolaylık olması açısından $N = \frac{10 \cdot S}{\pi \cdot R^2}$ eşitliğini kullanarak ağ üzerindeki düğümlerin sayısı *N*, verilen alan bilgisi S göz önüne alınarak, simulatör

ile otomatik olarak hesaplanmaktadır. Şekil 4.1'de $p_1,p_2,..,p_n$ orijinal giriş paketleridir.



Şekil 4.1 : Telsiz örgü ağ modeli

4.3 Benzetim Yazılımına İlişkin Açıklamalar

Benzetim yazılımı, bu çalışma için C++ programlama dilinde yazıldı. Benzetim yazılımı Linux işletim sistemi üzerinde komut dosyası aracılığıyla çalıştırılmaktadır. Benzetimlerde telsiz örgü ağda, ağ kodlamalı ve kodlamasız durumlar için, düğüm sayısı ve hizmet süresi ilişkisi incelenmektedir. Benzetim yazılımı, ideal 802.11x telsiz yerel alan ağı gibi hareket etmektedir. Linkler üzerinde kayıplar olmadığı varsayılmaktadır. Telsiz örgü ağ ve Gnutella dosya paylaşım uygulamalarının çalışma prensiplerine, Gossip yönlendirme yöntemi esasına uygun bir işleyiş mevcuttur. Kaynak dosya, öncelikle küçük parçalara ayrılmaktadır. Bu işleme bölümleme (splitting) denir. Bu bölümleme işlemi, ana dosyayı 8 Kbayt' lık paketlere bölmektedir. Ayrıca katsayılar 1 bayt büyüklüğündedir. Bu çalışmada kullanılan indirgenemez polinom :

$m(x) = x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^3 + 1$' dur.

Bu çalışmada paylaşılan dosya 50 adet 8 Kbayt'lık parçaya ayrılmaktadır. İndirgenemez polinom ve sonlu alanlar aritmetiği sayesinde, 8 kbayt'lık paket

büyüklüğü dosya paylaşımı süresi boyunca sabitlenmiş olacaktır.

Rasgele lineer ağ kodlama işlemi sırasında atanan, rasgele katsayılar, ağ üzerinde ihmal edilebilir bir ek veri yükü oluşturacaktır. Bu ihmal edilebilir ek veri yükü, aşağıdaki şekilde hesaplanacaktır[22].

Bölüm 2.4' te anlatıldığı üzere, katsayılar matrisinin sütun sayısı, herbir çıkış denklemi için, atanan katsayı sayısını vermektedir. Toplamda 50 parça orijinal giriş paketi olduğuna göre, bu katsayılar, ağ üzerinde, paylaşılan herbir paket için 50×1 bayt = 50 bayt 'lık bir ek yük bindirmektedir. Yani 8 Kbayt' lık herbir paketin yanında 50 bayt büyüklüğünde katsayı verisi taşınmaktadır. X, ağ üzerinde herhangi bir çıkış paketi olmak üzere, aşağıdaki gibi olmaktadır:

$$X = (1\text{bayt})_1 \cdot (8\ \text{Kbayt}) + (1\text{bayt})_2 \cdot (8\ \text{Kbayt}) + ... + (1\text{bayt})_{50} \cdot (8\ \text{Kbayt})$$

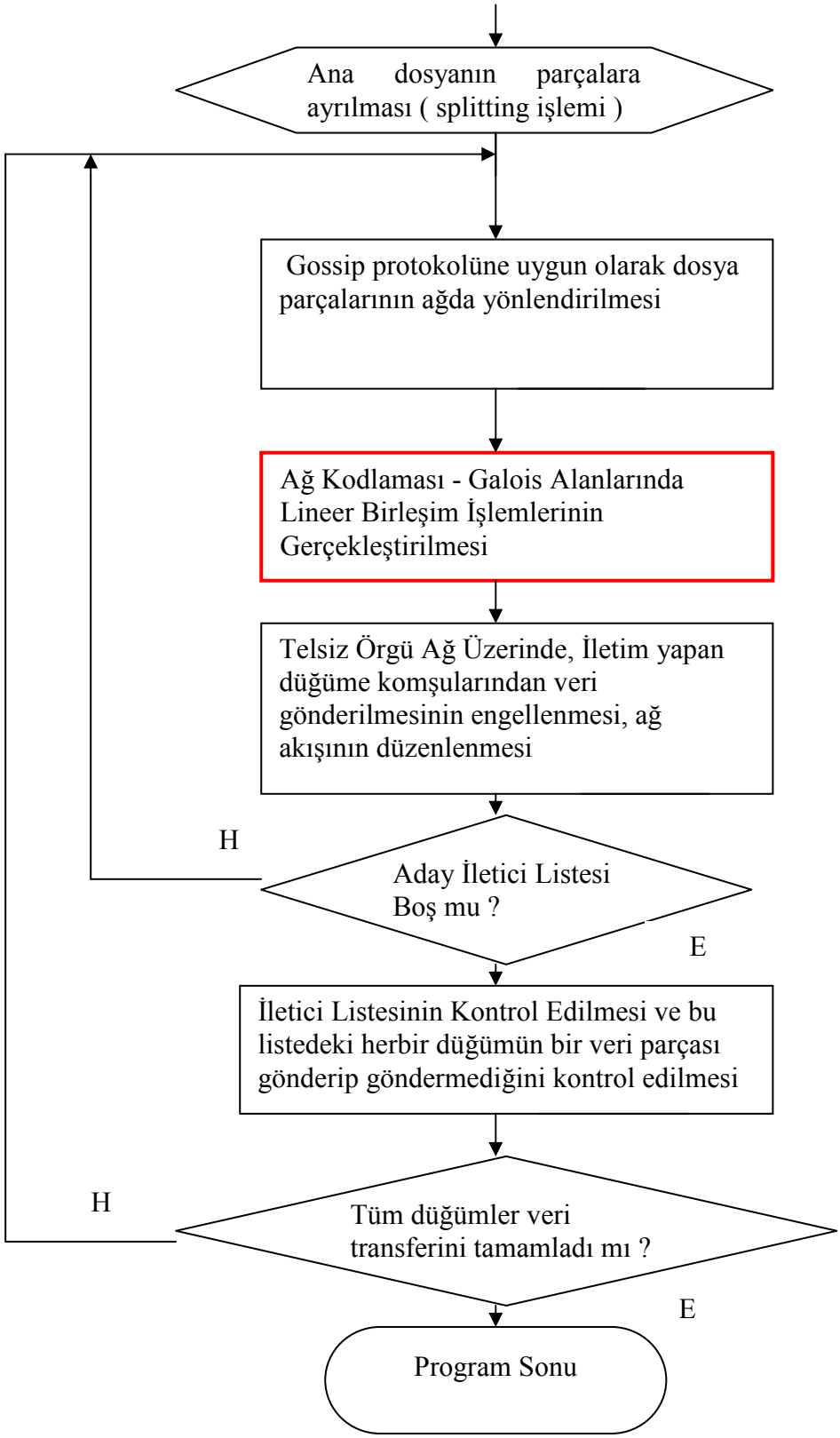
Dolayısıyla, katsayıların çıkış paketilerine getirdiği ek yük % 0.6 olmaktadır ve ağ kodlama sayesinde hizmet süresinde elde edilecek kısalma, ek yükü karşılayacak düzeyde yüksek olmaktadır. Hizmet süresindeki kısalmaya ilişkin benzetim sonuçları Şekil 5.18 - 5.19' da gösterilmektedir.

Bu parametre benzetim yazılımını çalıştırırken, komut satırında kullanıcı tarafından belirlenebilmektedir. Ayrıca bölüm 5.1' de belirtilmiş olan S alanı da kullanıcı tarafından belirlenebilmektedir. S alanının büyüklüğüne göre, N düğüm sayısı uygun şekilde belirlenmektedir. Böylece her bir kapsama alanı, 3 adet düğümü kapsayacak şekilde olmaktadır. Bu durumda R'nin değeri 2 olmaktadır. Düğümün kendisi de hsaba katılında aynı kapsama alanında 3 düğüm bulumaktadır.

Ağ üzerindeki hop işlemi , aynı adım içerisinde başlayıp , yine aynı adım içerisinde sona ermektedir. Telsiz örgü ağ üzerindeki her bir iletim işlemi, C orjinal paket parça sayısı olmak üzere, $1/C$ zaman dilimi içerisinde başlayıp, o zaman dilimi içerisinde sona ermektedir. 1 birim zamanda C adet adım gerçekleşmektedir. Dolayısıyla benzetim yazılımı, hizmet süresini birim zaman cinsinden hesaplamaktadır.

4.4 Benzetim Yazılımı Akış Diyagramı

Ağ kodlamalı durum için, akış diyagramı Şekil 4.2' de görülmektedir. Şekil 4.2'de öncelikle, kaynak dosya, dosya paylaşımı işleminin gerçekleşmesi için parçalara ayrılmaktadır. Daha sonra ilk dosya parçası, kaynak düğüm tarafından kendisine komşu olan düğümlerden birisi rasgele seçilerek iletilmektedir.



Şekil 4.2 : Telsiz örgü ağlarda gossip yöntemi kullanarak dosya paylaşımı ve ağ kodlama akış diyagramı

Ağ kodlama işlemi sonunda paket uzunlukları aynı kalacaktır. Bu çalışmada,akış diyagramından da görüldüğü üzere, yönlendirme yöntemi olarak gossip yöntemi kullanılmaktadır. Bu yönlendirme işlemi sırasında telsiz örgü ağ, dosya yönlendirme şartlarına uygun olarak, veriyi ileten düğüme komşu olan düğümler, bu düğüme veri iletimini yapmamaktadır. Çünkü telsiz örgü ağlarda, bir birim zaman içerisinde bir yönlendirici, veri alma veya iletme işlemlerinden yalnızca birisini gerçekleştirebilmektedir. Tasarlanan ağ, gecikmesiz bir ağdır, linklerde gecikme olmadığı varsayılmaktadır. Bir düğüme ilk gelen veri, hemen iletilmektedir.

Bu çalışmada, herbir düğüm için en az 2 komşu , en çok 3 adet komşu düğüm olması durumları incelenmektedir. Benzetimler, bu koşul altında gerçekleştirildi. Her bir durum için ayrı sonuçlar elde edilmekte ve sonuçlar karşılaştırılmaktadır.

4.5 Ağ Kodlamalı ve Ağ Kodlamasız Durumlar İçin Benzetim Sonuçları

Benzetimler, paylaşılan verinin 8 Kbayt' lik 50 parçadan oluştuğu göz önüne alınarak yapılmaktadır. Aşağıdaki şekilde telsiz örgü ağlarda, gossip yönlendirme yönteminin kullanıldığı senaryolar göz önüne alınarak, hizmet süresi ve düğüm sayısı ilişkisini belirten benzetim sonuçları elde edilmiş ve bu sonuçlar 2 boyutlu düzlemde, MATLAB programı aracılığıyla çizdirilmiştir. Benzetim yazılımında, eğrilerin çizdirilmesi öncesinde, Monte-Carlo yönteminden faydalanılarak sonuçlar analiz edilmiştir.

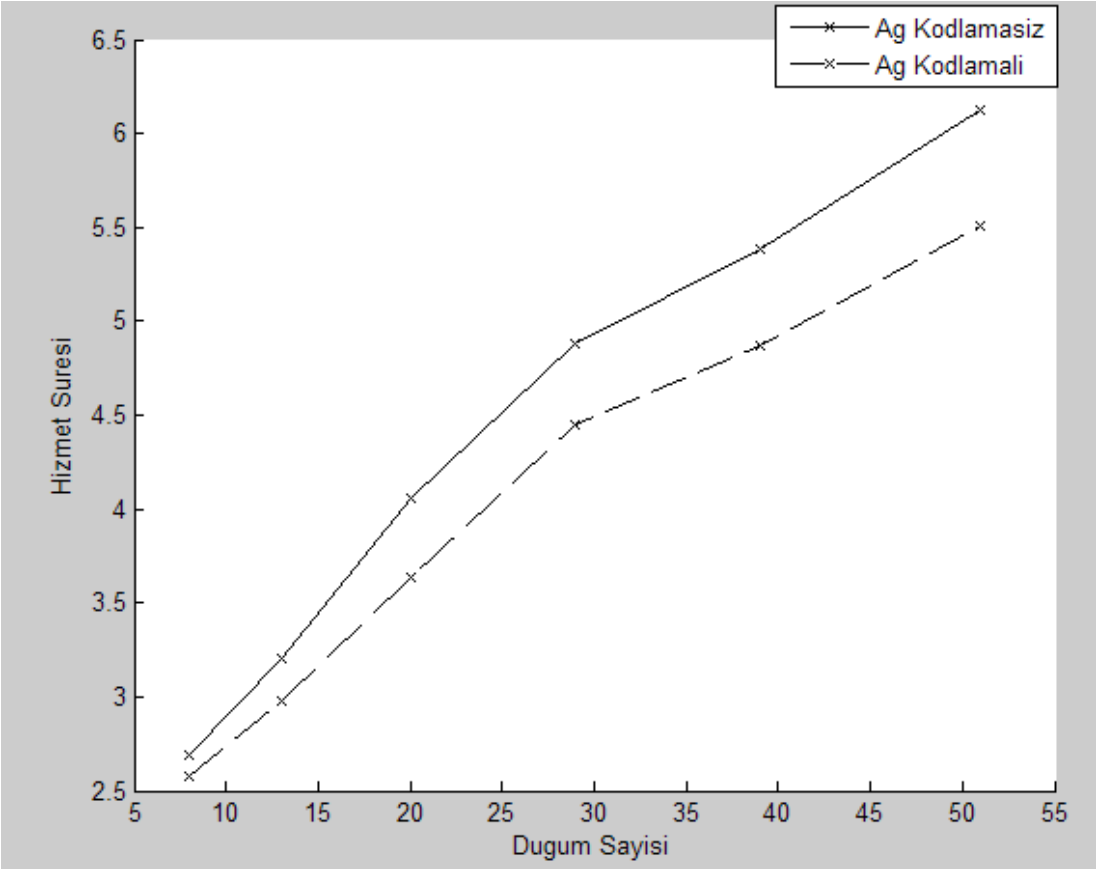
Çizelge 4.1' de telsiz örgü ağlarda düğüm sayısı ve hizmet süresine ilişkin ortalama benzetim sonuçları gösterilmektedir. Bu çizelgede komşu düğüm sayısını ifade eden R parametresinin değeri 2' dir.

Çizelge 4.1 : 2 komşu düğümlü durum için telsiz örgü ağlarda düğüm sayısı - hizmet süresi ilişkisi (R= 2 değeri için).

Telsiz Örgü Ağlarda Düğüm Sayısı - Hizmet Süresi İlişkisi (R=2)						
Düğüm Sayısı	8	13	20	29	39	51
Ağ Kodlamasız, Gossip Yönlendirme Yönteminde Hizmet Süresi (birim zaman)	2.74	3.12	4.03	4.88	5.38	6.12
Ağ Kodlama Uygulanmış, Gossip Yönlendirme Yönteminde Hizmet Süresi (birim zaman)	2.64	2.92	3.64	4.45	4.87	5.51

Benzetimlerde, en fazla 51 düğüm sayısına sahip olan telsiz örgü ağ yapıları incelenmiştir. Sonuçlar, benzetim yazılımının 1000 defa koşturulması sonucu oluşan sonuç değerlerin ortalamasıdır. Telsiz örgü ağlardaki iletim süresinin değişken olmasından dolayı, ortalama değerlere dayalı bir grafik çizme yaklaşımı tercih edilmiştir.

Çizelge 4.1' e ilişkin grafikte, yatay eksende düğüm sayısı, düşey eksende ise hizmet süresi bulunmaktadır. Çizelge 4.1' de gösterilen sonuçlar, MATLAB ortamında, Şekil 4.3' de görüldüğü gibi, çizdirilmiştir.



Şekil 4.3: Telsiz örgü ağlarda dosya paylaşımında, ağ kodlamalı ve ağ kodlamasız durumlar için, hizmet süresi ve düğüm sayısı ilişkisi (R=2)

Şekil 4.3'de elde edilmiş benzetim sonuçları, telsiz örgü ağlarda ağ dosya paylaşımında ağ kodlamalı durum ve ağ kodlamasız durumlar için karşılaştırmalı olarak çizdirilmiştir. Buradan ulaşılan sonuca göre, ağ kodlama yönteminin hizmet süresinde, hizmet süresinin ortalama olarak % 8 oranında kısalma sağladığı görülmektedir. Dosya paylaşım programları hizmet süresini kısaltmaktadır, bunun

üzerinde ağ kodlama yönteminin getirdiği % 8 oranında, hizmet süresinin kısalması günümüz iletişim teknolojileri için önemli bir değerdir.

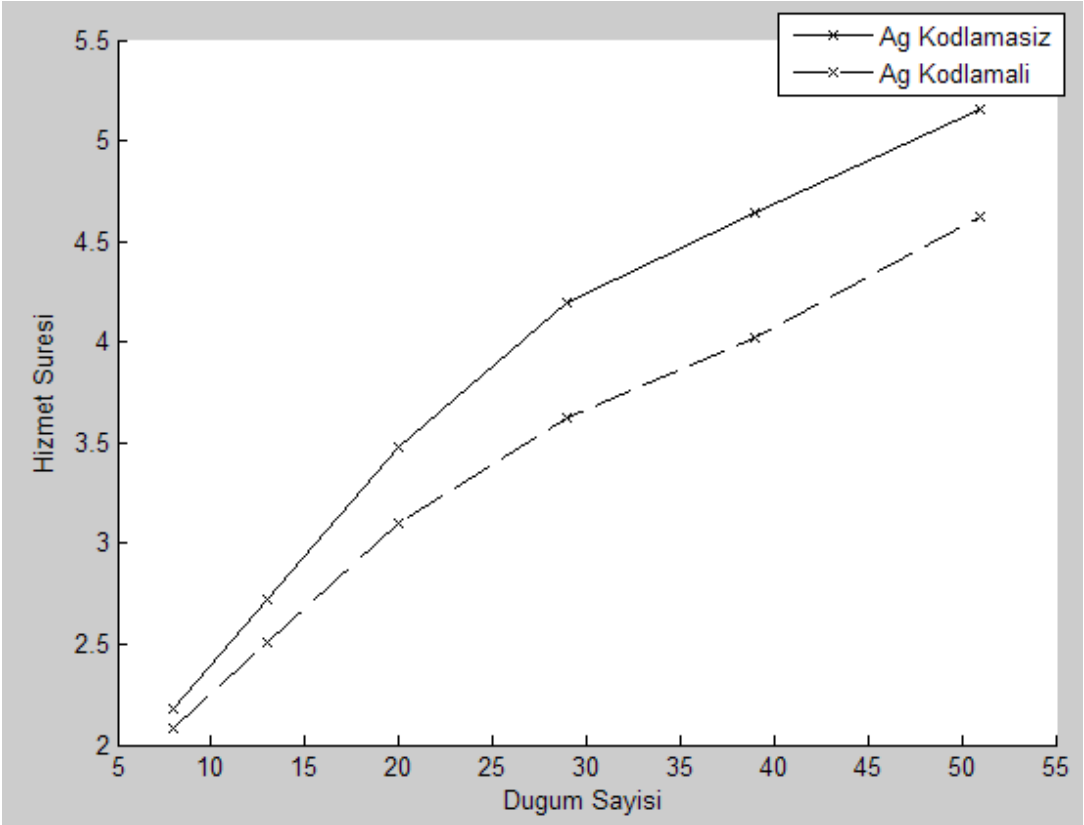
Çizelge 4.2' de telsiz örgü ağlarda düğüm sayısı ve hizmet süresine ilişkin ortalama benzetim sonuçları gösterilmektedir. Burada komşu düğüm sayısını ifade eden parametre olan R'nin değeri, 3 seçilmiştir.

Çizelge 4.2: 3 komşu düğümlü durum için telsiz örgü ağlarda düğüm sayısı - hizmet süresi ilişkisi (R= 3 değeri için)

Telsiz Örgü Ağlarda Düğüm Sayısı - Hizmet Süresi İlişkisi (R=3)						
Düğüm Sayısı	8	13	20	29	39	51
Ağ Kodlamasız, Gossip Yönlendirme Yönteminde Hizmet Süresi (birim zaman)	2.18	2.72	3.48	4.2	4.64	5.16
Ağ Kodlama Uygulanmış, Gossip Yönlendirme Yönteminde Hizmet Süresi (birim zaman)	2.08	2.51	3.1	3.62	4.02	4.62

Çizelge 4.2' ye ilişkin grafiğimizde, yatay eksende düğüm sayısı, düşey eksende ise hizmet süresi bulunmaktadır. Çizelge 4.2' de gösterilen sonuçlar, MATLAB ortamında, Şekil 4.4' te görüldüğü gibi, çizdirilmiştir.

Şekil 4.4 'te elde edilmiş benzetim sonuçları, telsiz örgü ağlarda ağ dosya paylaşımında ağ kodlamalı durum ve ağ kodlamasız durumlar için karşılaştırmalı olarak çizdirilmiştir. Buradan ulaşılan sonuca göre, ağ kodlama yönteminin hizmet süresinde, % 11 oranında kısalma sağladığı görülmektedir. Düğümlerin kapsama alanlarının 2 düğümden 3 düğüme çıkarılması ağ kodlamasının hizmet süresine getirdiği performansı da arttırmıştır. Dosya paylaşım programları hizmet süresini kısaltmaktadır, bunun üzerinde ağ kodlama yönteminin getirdiği % 11 oranında, hizmet süresindeki kısalma, günümüz iletişim teknolojileri için önemli bir değerdir. Telsiz örgü ağlardaki ağ elemanlarının kapsama alanlarının artması, ağ kodlama işlemiminin, hizmet süresi bakımından, performasını arttırmaktadır.

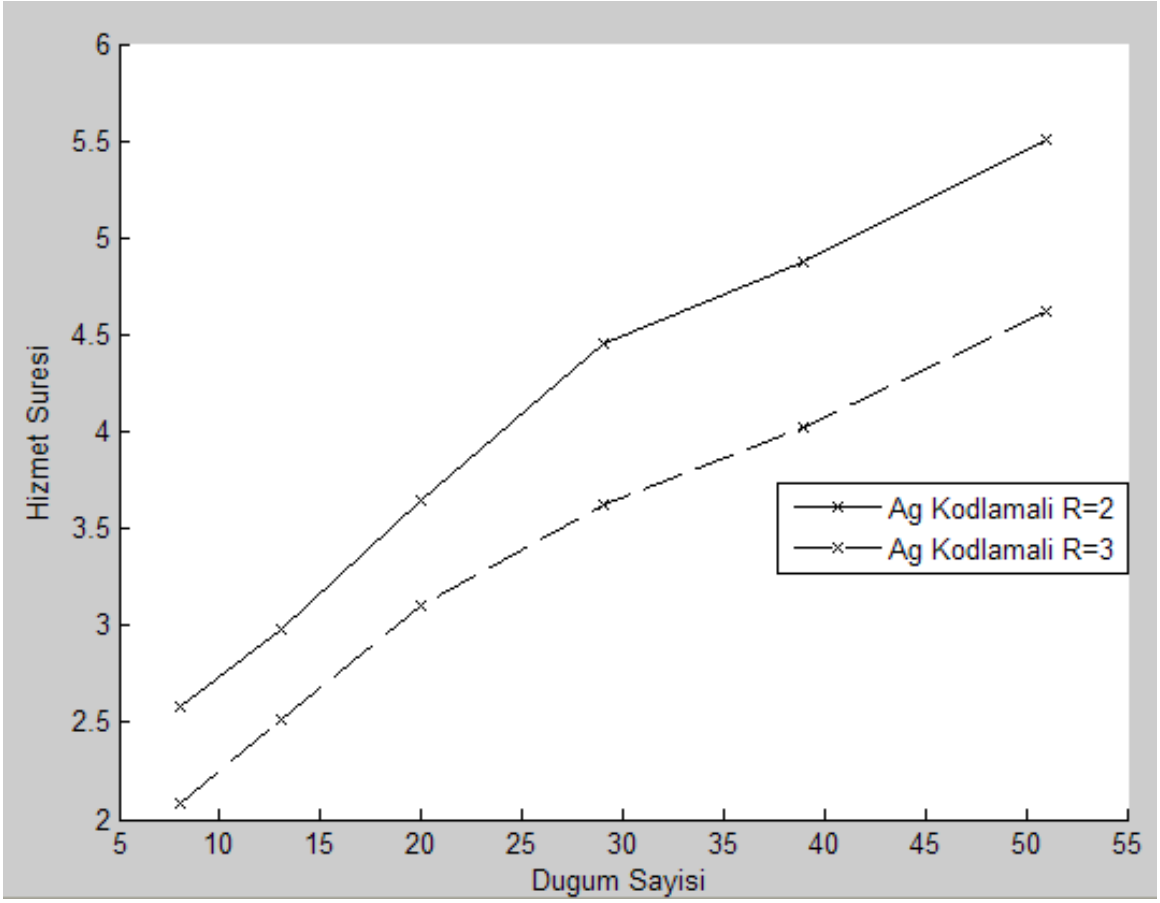


Şekil 4.4: Telsiz örgü ağlarda dosya paylaşımında, ağ kodlamalı ve ağ kodlamasız durumlar için, hizmet süresi ve düğüm sayısı ilişkisi (R=3)

Şekil 4.5'de görüldüğü üzere, düğümlerin kapsama alanlarındaki artış ve ağ kodlama yönteminin hizmet süresini kısaltmada gösterdiği performans incelenmektedir. Ağ kodlamalı durumda, R parametresinin 2 ve 3 değerleri için gerçekleştirilen benzetimlerin sonuçları, Şekil 4.5 üzerinde gösterilmiş ve her iki durum karşılaştırılmıştır.

Şekil 4.5'te her bir ağ düğümünün kapsama alanında, 2 düğüm ve 3 düğüm bulunması durumları, ağ kodlaması uygulanmış bir telsiz örgü ağda karşılaştırmalı olarak incelendi.

Sonuç olarak, Şekil 4.5'de görüldüğü üzere, düğümlerin kapsama alanlarındaki artış, ağ kodlama yönteminin hizmet süresini kısaltmada gösterdiği performansı arttırtmaktadır.



Şekil 4.5: R=2 ve R=3 için telsiz örgü ağlarda dosya paylaşımında, ağ kodlamalı ve ağ kodlamasız durumlar için, hizmet süresi ve düğüm sayısı ilişkisi karşılaştırması

5. SONUÇ VE ÖNERİLER

Bu çalışmada, ağ kodlama yönteminin, telsiz örgü ağlar üzerindeki dosya paylaşım uygulamalarında kullanılmasının, dosya paylaşımının tamamlanma süresine, yani hizmet süresine, olan etkisi incelenmektedir.

Benzetim yazılımının çalıştırılması sonucunda, ağ kodlamalı ve ağ kodlamasız durumlar için, telsiz örgü ağlarda, gossip yönlendirme yöntemini kullanan dosya paylaşımı uygulamalarının, hizmet süreleri hesaplanmaktadır. Bu sonuçlar karşılaştırılarak ağ kodlama yöntemi uygulanmasının, telsiz örgü ağlarda üzerinde gossip yönlendirme yöntemi kullanılarak gerçekleştirilen dosya paylaşımı uygulamalarında, hizmet süresini kısalttığı gösterilmektedir. Sonuç olarak, bu çalışma ile, n adet telsiz örgü ağ düğümüne, m adet bilgi paketinin en hızlı şekilde paylaştırılabilmesi problemi konusundaki çalışmalara, katkı sağlanmaya çalışılmaktadır. Bu çalışmada gerçekleştirilen uygulamanın gerçek hayattaki karşılığı, mobil şebeke operatörlerinin, sahada eşler arası ağ düzeninde çalışan telsiz yerel ağ üzerinde, yama yazılımlarını tüm ağ elemanlarına dağıtması işlemidir. Bu çalışmanın sonucunda, mobil şebeke operatörlerinin , yama yazılımlarını , ağ kodlama yöntemi kullanarak, sahadaki telsiz aygıtlara hızlıca dağıtabilmesi probleminin çözümüne katkı sağlanmaktadır.

Ağ elemanlarının kapsama alanlarındaki artışın ağ kodlama yöntemine olan etkisi de yine bu çalışmada incelenmektedir. Bunun sonucunda, ağ elemanlarının kapsama alanındaki artışın ağ kodlama işleminin performansını arttırdığı, kapsama alanı parametresi R'nin 2 ve 3 değerlerini aldığı durumlar için gösterilmektedir. Hizmet süresinde, R parametresinin değeri 2 olduğunda ortalama % 8, R parametresinin değeri 3 olduğunda ortalama %11, oranında kısalma olduğu gösterilmektedir.

Sonuç olarak, bu çalışma ile, rasgele lineer ağ kodlama yöntemi kullanılarak, telsiz örgü ağlar üzerinde, gossip yönlendirme yöntemini kullanan dosya paylaşımı uygulamalarının, dosya paylaşımını tamamlama süresini, yani hizmet süresini, kısalttığı gösterilmektedir.

Ağ kodlama yönteminin, hizmet süresini kısaltmanın yanı sıra, ağ güvenliği konusuna da katkısı vardır. Çünkü ağdaki paketler, düğümler üzerinde kodlanarak iletilmektedir, dolayısıyla ağ kodlama, ağ güvenliğinde önemli bir konu olan hattın dinlenmesi denemelerine (sniffing) karşı, ağ güvenliğine katkı sağlamaktadır[15]. Diğer bir araştırma konusu, telsiz örgü ağlar üzerindeki dosya paylaşımında ağ kodlama yöntemi ve ağ güvenliği olabilir. Ayrıca bu çalışmadaki uygulama sonuçları, benzetim yazılımı ile elde edilmiştir. Gerçek telsiz aygıtların olduğu ve gerçek bir IEEE 802.11x protokolü üzerinde haberleşme yapılan bilgisayar ağı üzerinde, ağ kodlama yöntemi uygulanması, hizmet süresine ilişkin sonuçlar elde edilmesi de, bir başka araştırma konusu olabilir. Bunun yanında, lineer ağ kodlama yönteminde, her bir ağ düğümünün çıkışında elde edilen çıkış denklemi, tüm orijinal giriş bilgi paketlerini değişken olarak içerisinde barındırmaktadır. Bu durum göz önüne alınarak, ağ üzerindeki paket kayıplarına karşı daha toleranslı bir ağ tasarlanabilir. Telsiz örgü ağlarda dosya paylaşımında ağ kodlama ve paket kayıplarına karşı tolerans, ayrı bir araştırma alanı olarak incelenebilir.

KAYNAKLAR

[1]**Yeung, R.W., Zhangi, Z.,**1999. "*Distributed source coding for satellite communications",* IEEE Transactions on Information Theory, Vol 45, No.4, pp. **1111 - 1120**

[2] **Ahlswede,R., Cai,N., Li, S.R., and Yeung. R.W.,**2000. **"***Network information flow",* IEEE Transactions on Information Theory.Vol 46, No. 4, pp.**1204-1216**

[3] **Cai, N., Li, S. R. , Yeung, R.W.**,2003. "*Linear Network Coding*", IEEE Transactions on Information Theory, Vol 49, No. 2 , pp. **371-381**

[4] **Ho, T., Koetter, R.,Medard, M., Karger, R., Effros, M.,**2003."*The benefits of coding over routing in a randomized setting",* International Symposium on Information Theory (ISIT), pp. **442.**

[5] **Katti, S., Katabi, D., Hu, W.,Rahul, H.,Medard M.,**2005,"*The importance of being opportunistic: Practical network coding for wireless environments",* In Proc. of 43rd Allerton Conference on Communication, Control, and Computing, Monticello, IL.

[6] **Gkantsidis, C., and Rodriguez, P.,**2005. "*Network Coding for large scale content distribution",* In Proc. Of INFOCOM, Miami, USA. Vol 4, pp: **2235-2245**

[7] **Chiu, D.M., Yeung, R.W, Huang, J., Fan, B.,** 2006. "*Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks"*,4th International Symposium on Information Theory , pp. **1- 5.**

[8] **Hamra, A.A., Barakat, C., Turletti, T.,**2006. "*Network Coding for Wireless Mesh Networks: A Case Study",* International Workshop on Wireless Mobile Multimedia Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks, pp. **103-114.**

[9] **Sattari, P., Markopoulou, A., Fragouli, C.,**2009, "*Multiple source multiple destination topology inference using network coding"*,in Proc. of Netcod Workshop, Lausanne, Switzerland. pp. **36-41**

[10] **Ripenau, M., Foster, I., Iamnitchi, A.,** 2002*, "A mapping the Gnutella network: Properties of large scale peer-to-peer systems and implications of system design",*IEEE Internet Computing Journal , Vol. **6 (1)**

[11] **Haas, Z.J., Halpem, J.Y, Li, L.,** 2002, "*Gossip based ad-hoc routing",* Proceedings of IEEE INFOCOM, Vol.14, Issue: 3 pp. **1707-1716.**

[12] **Nedos, A., Singh,K., Cunningham, R., Clarke, S.,**2007, "*A gossip protocol to support service discovery with heteregenous ontologies in MANETs",*

Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications(WiMob), pp. **53.**

[13] **Frey, D., Guerraoui,R., Kermarrec, A.M.,** 2009, "*Streching gossip with live streaming*", Dependable Systems & Networks 2009,IEEE/IFIP International Conference, pp.**259-264.**

[14] **Kırmızıgül, S.,** 2005. "*Ağ Akışı"*. Gebze Yüksek Teknoloji Enstitüsü Teknik Rapor

[15] **Fragouli,C. , Boudec,J.L. , Widmer, J.**, "*Network Coding: An Instant Primer*" ACM SIGCOMM Computer Communication Review, Vol.**36**, Issue.1 pp. **63-68**

[16]<*http://www.ece.rochester.edu/research/wcng/meetings/PerilloNetworkCoding.pdf*>, alındığı tarih 16.08.2010

[17] **Hossain, E., Leung, K.K.,**2008, *Wireless Mesh Networks, Architectures and Protocols* ,Springer Science+Bussiness Media LCC.

[18] <*http://www.networkcoding.com*>, alındığı tarih 27.03.2010

[19] **Lee , K.C., Yap, I.Y.,** 2006, "*CarTorrent : A Bit-Torrent System for Vehicular Networks*", IEEE INFOCOM.

[20] **Katti, S., Rahul, H., Hu, W., Katabi, D., Medard, M., Crowcroft, J.,** 2008, "*XORs in the air : practical wireless network coding",* IEEE/ACM Transactions on Networking,Vol 16, Issue 3 , pp.**497-510.**

[21] **T.K.Philips, S.S. Panwwar , and A.N. Tantawi.,**1989. "*Connectivity Properties of packet radio network model*" In Proc, IEEE Transactions on Information Theory, Vol.35 pp.**1044-1047**

[22] **Zhang, X., Li, B.,** 2008, "*Optimized multipath network coding in lossy wireless Networks"* Proceedings of the 2008 The 28th International Conference on Distributed Computing Systems, pp. **243-250**

[23] <*http://www.partow.net/projects/galois/*> , alındığı tarih 28.03.2010

EKLER

EK A.1: Galois Alanı ve C++ Programlama Dili Aracılığı ile Galois Alanlarındaki İşlemlerin Gerçekleştirilmesi

EK A.2 : Graf Elemanlarının Benzetim Ortamında Yaratılması

EK A.3 : Telsiz Örgü Ağlarda Gossip Yönlendirme ve Ağ Kodlama Benzetim Yazılımı Kodları

EK A.1

Galois Alanı ve C++ Programlama Dili Aracılığı ile Galois Alanlarındaki İşlemlerin Gerçekleştirilmesi

Her p asal sayısı ve her m≥1 tam sayısı için tek türlü belirli olarak var olan, p^m elemanlı sonlu cisme p^m elemanlı galois alanı denir ve GF(p^m) ile gösterilir. Burada 2 elemanlı Galois alanları GF(2^m) ile gösterilir, genelde bit düzeyindeki işlemler için kullanılmaktadır. Galois alanlarını programlamada C++ Partow Galois kütüphanesi kullanıcaktır [23].Primitif polinom genişletilerek ihtiyacımız olan diğer polinomlar kolayca elde edilebilmektedir. Bir Primitif polinomun C++ koduyla nasıl ifade edilebildiği aşağıda gösterilmiştir.Bu tezde, 8 Kbayt'lık veri paketleri dosya paylaşımında kullanılacağından, bu paketleri sonlu alanlar artimetiğine uygun şekilde taşımak için GF(2^{13}) uzayında tanımlı bir indirgenemez polinom elde etmemiz gereklidir. İndirgemenez polinom MATLAB üzerinde aşağıdaki komutla elde edilmiştir.

> m= 13;

> allprimpolys = primpoly (m) ;

Elde edilen polinomlarda biri seçilmiştir, ve bu polinom

$GF(2^{13}) = 2^{13} + 2^{12} + 2^{11} + 2^{10} + 2^9 + 2^8 + 2^7 + 2^3 + 1$ 'dur.

Bu polinom C++ Partow Galois kütüphanesinde aşağıdaki şekilde gösterilmektedir.

```
/*
p(x)=1x^13+1x^12+1x^11+1x^10+1x^9+1x^8+1x^7+0x^6+0x^5+0x^4+1x^3+0x^2
+0x^1+1x^0
*/
unsigned int prim_poly[14] = {1,0,0,1,0,0,0,1,1,1,1,1,1,1};
/*
 A Galois Field of type GF(2^13)
*/
 galois::GaloisField gf(13,prim_poly);
```

Primitif (indirgenemez) polinomun genişletilmesiyle yeni polinomların elde edilmesi de aşağıda gösterilmiştir.

```
galois::GaloisField gf(8,prim_poly);
galois::GaloisFieldElement element1(&gf, 1);
galois::GaloisFieldElement element2(&gf, 2);
```

Elde ettiğimiz genişletilmiş polinomlara katsayıların nasıl ekleneceği ise aşağıda belirtilmiştir . Elde etmek istediğimiz polinom

p(x) = 10x^9 + 9x^8 + 8x^7 + 7x^6 +6x^5 + 5x^4 + 4x^3 + 3x^2 + 2x^1 + 1x^0

olacaktır.

```
galois::GaloisField gf(8,prim_poly);
galois::GaloisFieldElement gfe[10] = {
                                        galois::GaloisFieldElement(&gf, 1),
                                        galois::GaloisFieldElement(&gf, 2),
                                        galois::GaloisFieldElement(&gf, 3),
                                        galois::GaloisFieldElement(&gf, 4),
                                        galois::GaloisFieldElement(&gf, 5),
                                        galois::GaloisFieldElement(&gf, 6),
                                        galois::GaloisFieldElement(&gf, 7),
                                        galois::GaloisFieldElement(&gf, 8),
                                        galois::GaloisFieldElement(&gf, 9),
                                        galois::GaloisFieldElement(&gf,10)
                                    };
galois::GaloisFieldPolynomial polynomial(&gf,9,gfe);
```

Galois Alanında bir lineer birleşim işleminin nasıl yapılacağı aşağıda belirtilmiştir.

```
galois::GaloisField gf(8,prim_poly);
galois::GaloisFieldElement element1(&gf, 1);
```

galois::GaloisFieldElement element2(&gf, 2);

galois::GaloisFieldElement element3; element3 = element1 + element2;

// lineer birleşim toplama

Katsayılar kümesini de hesaba katarak, ağ kodlaması için uygun olacak bir lineer birleşim örneği aşağıda gösterilmiştir. Benzetimlerde esas alınan işlem tipi aşağıdaki gibidir.

galois::GaloisField gf(8,prim_poly);

galois::GaloisFieldElement gfe1[10] = {

galois::GaloisFieldElement(&gf, 1),

galois::GaloisFieldElement(&gf, 2),

galois::GaloisFieldElement(&gf, 3),

galois::GaloisFieldElement(&gf, 4),

galois::GaloisFieldElement(&gf, 5),

galois::GaloisFieldElement(&gf, 6),

galois::GaloisFieldElement(&gf, 7),

galois::GaloisFieldElement(&gf, 8),

galois::GaloisFieldElement(&gf, 9),

galois::GaloisFieldElement(&gf,10)

};

galois::GaloisFieldElement gfe2[10] = {

galois::GaloisFieldElement(&gf,10),

galois::GaloisFieldElement(&gf, 9),

galois::GaloisFieldElement(&gf, 8),

galois::GaloisFieldElement(&gf, 7),

galois::GaloisFieldElement(&gf, 6),

galois::GaloisFieldElement(&gf, 5),

galois::GaloisFieldElement(&gf, 4),

```
                galois::GaloisFieldElement(&gf, 3),
                galois::GaloisFieldElement(&gf, 2),
                galois::GaloisFieldElement(&gf, 1)
            };
galois::GaloisFieldPolynomial poly1(&gf, 9,gfe1);
galois::GaloisFieldPolynomial poly2(&gf, 9,gfe2);
galois::GaloisFieldPolynomial poly3;
poly3 = poly1 + poly2; // lineer toplama
```

EK A.2

Graf Elemanlarının Benzetim Yazılımı ile Yaratılması

Grafların C++ 'da tanımlanması için genellikle yineleyicileri kullanmamız kolaylık sağlayacaktır.

Yineleyici : Kitlesel bir nesnenin altında bulunan nesnelere, nesnelerin nasıl temsil edildiklerine bakmaksızın, sırasıyla ulaşılmasını sağlamaktadır. Bu sayede farklı şekilde temsil edilen nesnelere tek bir arayüz üzerinden ulaşılabilecektir.

Aşağıda bir for döngüsü ve yineleyici karşılaştırması yapılmıştır. For döngüsü örneği aşağıdadır.

```
İnt i ;
For (i = 0 ; i < liste.size( ) ; i ++ ) {
    System.out.println ( i.get(i));
}
```

Yineleyici örneği örneği aşağıdadır.

```
iteratör i = liste.Iteratör( );
while ( i.hasNext( ) ){
    System.out.println( i.next( ) );
}
```

Benzetim yazılımında , yineleyici aracılığıyla düğüm yapıları oluşturulmaktadır.

EK A.3

Telsiz Örgü Ağlarda Gossip Yönlendirme ve Ağ Kodlama Benzetim Yazılımı Kodları

```
vector<int> gossip_all_chunks_;
class gossip_our_node_;
vector<gossip_our_node_ *> gossip_list_nodes_;
struct gossip_our_node_ {
public:
  gossip_our_node_();
  ~gossip_our_node_();
  int id_, aday;
  double x, y;
  vector<int> list_neighbors_, interested;
  vector<galois::GaloisFieldElement> full_comb;
  vector< vector<galois::GaloisFieldElement> > list_chunks_, just_received_;
  int independence_();
  int independent_nodes();
};
gossip_our_node_::gossip_our_node_() {
  this->id_ = 0;
  this->aday = 0;
  this->x = 0;
  this->y = 0;
  this->full_comb.clear();
```

```
  this->interested.clear();
  this->list_chunks_.clear();
  this->list_neighbors_.clear();
  this->just_received_.clear();
}
gossip_our_node_::~gossip_our_node_() {
  this->id_ = 0;
  this->aday = 0;
  this->x = 0;
  this->y = 0;
  this->full_comb.clear();
  this->interested.clear();
  this->list_chunks_.clear();
  this->list_neighbors_.clear();
  this->just_received_.clear();
}

int gossip_our_node_::independence_() {
  if (this->list_chunks_.size() == 0) {
    return 1;
  }
/* Galois Alanalarında Ağ Kodlama İşlemi*/
  int i, j, p, q;
  const int m=this->list_chunks_.size()+1, n=nb_chunks_;
  vector< vector<galois::GaloisFieldElement> > check_list_;
  for (i=0; i<m-1; i++) {
```

```
  check_list_.push_back(this->list_chunks_[i]);
 }
 check_list_.push_back(this->just_received_[0]);
/* Ağ Kodlama ve Galois Alanlarında uygun olarak lineer birleşimlerin oluşturulması*/
 i=0;
 j=0;
 while ((i < m) && (j < n) ) {
  galois::GaloisFieldElement max_val = check_list_[i][j];
  int max_ind = i;
  int k;
  if (max_val == 0) {
   for (k=i+1; k<m; k++) {
        galois::GaloisFieldElement val = check_list_[k][j];
        if (val > max_val) {
         max_val = val;
         max_ind = k;
         break;
        }
   }
  }
  if (max_val != 0) {
   if (max_ind != i) {
        vector< vector<galois::GaloisFieldElement> > temp_;
        temp_.push_back(check_list_[i]);
        temp_.push_back(check_list_[max_ind]);
```

```
        for (p=0; p<n; p++) {
          check_list_[max_ind][p] = temp_[0][p];
          check_list_[i][p] = temp_[1][p];
        }
    }
    for (q=i+1; q<m; q++) {
        galois::GaloisFieldElement toto_(&gf, 0);
        toto_ = check_list_[q][j];
        for (p=j; p<n; p++) {
          galois::GaloisFieldElement tata1(&gf, 0);
          tata1 = check_list_[i][p]/check_list_[i][j];
          galois::GaloisFieldElement tata2(&gf, 0);
          tata2 = toto_*tata1;
          check_list_[q][p] = check_list_[q][p] - tata2;
        }
    }
    i = i + 1;
  }
  j = j + 1;
}
int rank_ = 0;
for (i=0; i<m; i++) {
  for (j=0; j<n; j++) {
    if (check_list_[i][j] != 0) {
        rank_++;
        break;
```

```
      }
    }
  }
  if (rank_ == m) {return 1;}
  else {return 0;}
}
/* Gossip Yönlendirme Yöntemine Uygun Yönlendirme İşlemi */
int gossip_our_node_::independent_nodes() {
  bool keep_node_ = false;
  for (int i=0; i<this->list_neighbors_.size(); i++) {
    gossip_our_node_ * neighbor_;
    neighbor_ = gossip_list_nodes_[this->list_neighbors_[i]];
    if (neighbor_->list_chunks_.size() < nb_chunks_) {
      neighbor_->just_received_.clear();
      neighbor_->just_received_.push_back(this->full_comb);

      if (neighbor_->independence_() == 1) {
          keep_node_ = true;
          this->interested.push_back(neighbor_->id_);
      }
      neighbor_->just_received_.clear();
    }
  }
  if (keep_node_ == false) {return 0;}
  else {return 1;}
}
```

```
void gossip_create_nodes_() {
  for (int i=0; i<nb_nodes_; i++) {
    gossip_our_node_ *new_node_ = new gossip_our_node_();
    new_node_->id_ = i;
    gossip_list_nodes_.push_back(new_node_);
    if (
          ((kaynak_d == 1) && i!=source1->id_) ||
          ((kaynak_d == 2) && i!=source1->id_ && i!=source2->id_) ||
          ((kaynak_d == 3) && i!=source1->id_ && i!=source2->id_ && i!=source3-
>id_)


          ) {
      missing_nodes_.push_back(i);
      for (int j=0; j<nb_chunks_; j++) {
          galois::GaloisFieldElement x(&gf, 0);
          new_node_->full_comb.push_back(x);
      }
    }
    else {
      adays_queue_.push_back(i);
      new_node_->aday = 1;
      vector<galois::GaloisFieldElement> list_coef_;
      for (int j=0; j<nb_chunks_; j++) {
          galois::GaloisFieldElement x(&gf, 0);
          list_coef_.push_back(x);
```

```
      }
      for (int j=0; j<nb_chunks_; j++) {
           new_node_->list_chunks_.push_back(list_coef_);
      }
    }
  }
}
void gossip_generate_xy() {
  gossip_our_node_ * node1;
  node_xy * node2;
  for (int i=0; i<nb_nodes_; i++) {
    node1 = gossip_list_nodes_[i];
    node2 = list_xy[i];
    node1->x = node2->x;
    node1->y = node2->y;
    for (int j=0; j<node2->list_neighbors_.size(); j++) {
      node1->list_neighbors_.push_back(node2->list_neighbors_[j]);
    }
  }
}
void  gossip_select_transmitters() {
  vector<int>::iterator iter_int_;
  vector<int> potential_transmitters_;
    potential_transmitters_ = adaylar_queue_;
/* Rasgele katsayılar seçme ve  bu katsayıların lineer birleşimlerde kullanma*/
  while (potential_transmitters_.size() > 0) {
```

```
int j = int((double(rand())/RAND_MAX)*potential_transmitters_.size());
int k = potential_transmitters_[j];
gossip_our_node_ * selected_node_;
selected_node_ = gossip_list_nodes_[k];
selected_node_->full_comb.clear();
if (
    ((kaynak_d == 1) && selected_node_->id_==source1->id_) ||
    ((kaynak_d == 2) && (selected_node_->id_==source1->id_ || selected_node_->id_==source2->id_)) ||
    ((kaynak_d == 3) && (selected_node_->id_==source1->id_ || selected_node_->id_==source2->id_ || selected_node_->id_==source3->id_))
    ) {
  for (int x=0; x<nb_chunks_; x++){
    double tompo = (double(rand())/RAND_MAX)*CODING_SPACE + 1;
    int temp1 = int(tompo);
    galois::GaloisFieldElement temp_(&gf, temp1);
    selected_node_->full_comb.push_back(temp_);
  }
}
else {
  vector<galois::GaloisFieldElement> rand_coef_;
  rand_coef_.clear();
  int p,q;
  for (q=0; q<selected_node_->list_chunks_.size(); q++) {
    double tompo = (double(rand())/RAND_MAX)*CODING_SPACE + 1;
    int temp1 = int(tompo);
    galois::GaloisFieldElement temp_(&gf, temp1);
```

```
        rand_coef_.push_back(temp_);
    }
    for (q=0; q<nb_chunks_; q++) {
        galois::GaloisFieldElement tata_(&gf, 0);
        for (p=0; p<selected_node_->list_chunks_.size(); p++) {
          tata_ = tata_ + rand_coef_[p]*selected_node_->list_chunks_[p][q];
        }
        selected_node_->full_comb.push_back(tata_);
    }
  }
  if (selected_node_->independent_nodes() == 0) {
    vector<int>::iterator start = adaylar_queue_.begin();
    vector<int>::iterator end = adaylar_queue_.end();
    iter_int_ = find(start, end, selected_node_->id_);
    adaylar_queue_.erase(iter_int_);
    selected_node_->aday = 0;
    start = potential_transmitters_.begin();
    end = potential_transmitters_.end();
    iter_int_ = find(start, end, selected_node_->id_);
    potential_transmitters_.erase(iter_int_);
  }
  else {
    ready_queue_.push_back(selected_node_->id_);
    vector<int>::iterator start;
    vector<int>::iterator end;
```

```
vector<int> to_remove;
to_remove.clear();
to_remove.push_back(selected_node_->id_);
for (j=0;j<selected_node_->list_neighbors_.size(); j++) {
    int neighbor_id = selected_node_->list_neighbors_[j];
    gossip_our_node_ * neighbor_ = gossip_list_nodes_[neighbor_id];
    start = to_remove.begin();
    end = to_remove.end();
    iter_int_ = find(start, end, neighbor_id);
    if (iter_int_ == to_remove.end()) {to_remove.push_back(neighbor_id);}
    for (k=0;k<neighbor_->list_neighbors_.size(); k++) {
      int new_id = neighbor_->list_neighbors_[k];
      start = to_remove.begin();
      end = to_remove.end();
      iter_int_ = find(start, end, new_id);
      if (iter_int_ == to_remove.end()) {to_remove.push_back(new_id);}
    }
}
while(to_remove.size() > 0) {
    int removed_node = to_remove[0];
    iter_int_ = to_remove.begin();
    to_remove.erase(iter_int_);
    start = potential_transmitters_.begin();
    end = potential_transmitters_.end();
    iter_int_ = find(start, end, removed_node);
    if(iter_int_!=potential_transmitters_.end())
```

```
{potential_transmitters_.erase(iter_int_);}
      }
    }
  }
  potential_transmitters_.clear();
}
void gossip_perform_transmission() {
  vector<int>::iterator iter_int_;
  while (ready_queue_.size() > 0) {
    int j, k, q;
    gossip_our_node_ * transmitter_;
    j = ready_queue_[0];
    iter_int_ = ready_queue_.begin();
    ready_queue_.erase(iter_int_);
    transmitter_ = gossip_list_nodes_[j];
    packet_emitted++;
    packet_received = packet_received + transmitter_->list_neighbors_.size();
    vector<galois::GaloisFieldElement> transmitted_chunk_, rand_coef_;
    if (transmitter_->list_chunks_.size() == 1) {
      for (q=0; q<nb_chunks_; q++){
          transmitted_chunk_.push_back(transmitter_->list_chunks_[0][q]);
      }
    }
    else {
      for (q=0; q<nb_chunks_; q++){
          transmitted_chunk_.push_back(transmitter_->full_comb[q]);
```

```
    }
    transmitter_->full_comb.clear();
   }
   for (j=0; j<transmitter_->interested.size(); j++) {
    int proba = int((double(rand())/RAND_MAX)*100.0 + 1);
    if (proba > losses) {
        int neighbor_id_ = transmitter_->interested[j];
        gossip_our_node_ * neighbor_ = gossip_list_nodes_[neighbor_id_];
        packet_useful++;
        int chunk_stat = neighbor_->list_chunks_.size();
        all_chunks_[chunk_stat] = all_chunks_[chunk_stat] - 1;
        neighbor_->list_chunks_.push_back(transmitted_chunk_);
        chunk_stat = neighbor_->list_chunks_.size();
        all_chunks_[chunk_stat] = all_chunks_[chunk_stat] + 1;
        if (neighbor_->list_chunks_.size() == nb_chunks_) {
          vector<int>::iterator start = missing_nodes_.begin();
          vector<int>::iterator end = missing_nodes_.end();
          iter_int_ = find(start, end, neighbor_id_);
          if (iter_int_ != missing_nodes_.end()) {
            missing_nodes_.erase(iter_int_);
          }
        }
        int tata = neighbor_->list_chunks_.size();
        for (k=0; k<nb_chunks_; k++) {
          neighbor_->full_comb[k]=neighbor_->full_comb[k]+neighbor_-
>list_chunks_[tata-1][k];
```

```
            }
            if (neighbor_->aday == 0) {
              neighbor_->aday = 1;
              adaylar_queue_.push_back(neighbor_id_);
            }
        }
      }
      transmitter_->interested.clear();
    }
  }
  void run_selective_coding() {
    service_time = 0;
    packet_emitted = 0;
    packet_received = 0;
    packet_useful = 0;
    packet_useless = 0;
    all_chunks_.clear();
    missing_nodes_.clear();
    ready_queue_.clear();
    adaylar_queue_.clear();
    receiving_queue_.clear();
    gossip_list_nodes_.clear();
    all_chunks_.push_back(nb_nodes_-1);
    for (int a=1; a<nb_chunks_; a++) {
      all_chunks_.push_back(0);
    }
```

```
 all_chunks_.push_back(1);
 char temp_gossip_service_time_[100];
 char temp_gossip_completed_clients_[100];
 char temp_gossip_evolution_chunks_[100];
 char temp_gossip_packet_emitted_[100];
 char temp_gossip_packet_useless_[100];
/* Sonuç çıktılarının yazdırılması*/
 if (source_place == 'r') {
   strcpy(temp_gossip_service_time_,gossip_service_time_);
   if (kaynak_d == 1) {
     strcat(temp_gossip_service_time_,"_r.m");
   }
   if (kaynak_d == 2) {
     strcat(temp_gossip_service_time_,"_2r.m");
   }
   if (kaynak_d == 3) {
     strcat(temp_gossip_service_time_,"_3r.m");
   }
   strcpy(temp_gossip_completed_clients_,gossip_completed_clients_);
   if (kaynak_d == 1) {
     strcat(temp_gossip_completed_clients_,"_r.m");
   }
   if (kaynak_d == 2) {
     strcat(temp_gossip_completed_clients_,"_2r.m");
   }
   if (kaynak_d == 3) {
```

```
  strcat(temp_gossip_completed_clients_,"_3r.m");
}
strcpy(temp_gossip_evolution_chunks_,gossip_evolution_chunks_);
if (kaynak_d == 1) {
  strcat(temp_gossip_evolution_chunks_,"_r.m");
}
if (kaynak_d == 2) {
  strcat(temp_gossip_evolution_chunks_,"_2r.m");
}
if (kaynak_d == 3) {
  strcat(temp_gossip_evolution_chunks_,"_3r.m");
}
strcpy(temp_gossip_packet_emitted_,gossip_packet_emitted_);
if (kaynak_d == 1) {
strcat(temp_gossip_packet_emitted_,"_r.m");
}
if (kaynak_d == 2) {
strcat(temp_gossip_packet_emitted_,"_2r.m");
}
if (kaynak_d == 3) {
strcat(temp_gossip_packet_emitted_,"_3r.m");
}
strcpy(temp_gossip_packet_useless_,gossip_packet_useless_);
if (kaynak_d == 1) {
  strcat(temp_gossip_packet_useless_,"_r.m");
}
```

```
  if (kaynak_d == 2) {
    strcat(temp_gossip_packet_useless_,"_2r.m");
  }
  if (kaynak_d == 3) {
    strcat(temp_gossip_packet_useless_,"_3r.m");
  }
}
  strcpy(temp_gossip_evolution_chunks_,gossip_evolution_chunks_);
  if (kaynak_d == 1) {
    strcat(temp_gossip_evolution_chunks_,"_e.m");
  }
  if (kaynak_d == 2) {
    strcat(temp_gossip_evolution_chunks_,"_2e.m");
  }
  if (kaynak_d == 3) {
    strcat(temp_gossip_evolution_chunks_,"_3e.m");
  }
  strcpy(temp_gossip_packet_emitted_,gossip_packet_emitted_);
  if (kaynak_d == 1) {
  strcat(temp_gossip_packet_emitted_,"_e.m");
  }
  if (kaynak_d == 2) {
  strcat(temp_gossip_packet_emitted_,"_2e.m");
  }
  if (kaynak_d == 3) {
  strcat(temp_gossip_packet_emitted_,"_3e.m");
```

```
  }
  strcpy(temp_gossip_packet_useless_,gossip_packet_useless_);
  if (kaynak_d == 1) {
    strcat(temp_gossip_packet_useless_,"_e.m");
  }
  if (kaynak_d == 2) {
    strcat(temp_gossip_packet_useless_,"_2e.m");
  }
  if (kaynak_d == 3) {
    strcat(temp_gossip_packet_useless_,"_3e.m");
  }
}
ofstream time_slot(temp_gossip_service_time_, ios::out);
if (!time_slot) {
  cerr << "the file service_time_ could not be opened" << endl;
}
time_slot << "time_slot = [";
ofstream completed_clients_(temp_gossip_completed_clients_, ios::out);
if (!completed_clients_) {
  cerr << "the file gossip_completed_clients_ could not be opened" << endl;
}
completed_clients_ << "gossip_completed_clients_ = [";
ofstream out_allchunks(temp_gossip_evolution_chunks_, ios::out);
if (!out_allchunks) {
  cerr << "the file evolution_chunks could not be opened" << endl;
}
```

```
out_allchunks << "gossip_evolution_chunks_ = [";
ofstream packet_emitted_(temp_gossip_packet_emitted_, ios::out);
if (!packet_emitted_) {
  cerr << "the file packet_emitted could not be opened" << endl;
}
packet_emitted_ << "gossip_packet_emitted_ =[";
ofstream packet_useless_(temp_gossip_packet_useless_, ios::out);
if (!packet_useless_) {
  cerr << "the file packet_useless_ could not be opened" << endl;
}
packet_useless_ << "gossip_packet_useless_ = [";
gossip_create_nodes_();
gossip_generate_xy();
for (int a=0; a<nb_chunks_+1; a++) {
  out_allchunks << all_chunks_[a] << " ";
}
out_allchunks << endl;
completed_clients_ << nb_nodes_ - missing_nodes_.size() << " ";
packet_emitted_ << packet_emitted << " ";
packet_useless_ << packet_received - packet_useful << " ";
time_slot << service_time << " ";
cout << "\n_____" << endl;
cout << "start selective coding" << endl;
while ((missing_nodes_.size() > 0) && (adaylar_queue_.size() > 0)) {
  cout << "run gossip_select_transmitters" << endl;
  gossip_select_transmitters();
```

```
  cout << "run gossip_perform_transmission" << endl;
  gossip_perform_transmission();
  service_time = service_time + 1.0/nb_chunks_;
  for (int a=0; a<nb_chunks_+1; a++) {
    out_allchunks << all_chunks_[a] << " ";
  }
  out_allchunks << endl;
  completed_clients_ << nb_nodes_ - missing_nodes_.size() << " ";
  packet_emitted_ << packet_emitted << " ";
  packet_useless_ << packet_received - packet_useful << " ";
  time_slot << service_time << " ";
}
cout << "end of selective coding\n" << endl;
time_slot << "]; \n\n";
time_slot.close();
completed_clients_ << "]; \n\n";
completed_clients_.close();
out_allchunks << "]; \n\n";
out_allchunks.close();
packet_emitted_ << "]; \n\n";
packet_emitted_.close();
packet_useless_ << "]; \n\n";
packet_useless_.close();
for (int i=0; i<nb_nodes_; i++) {
  delete gossip_list_nodes_[i];
}
```

```
  missing_nodes_.clear();
  ready_queue_.clear();
  adaylar_queue_.clear();
  receiving_queue_.clear();
  gossip_list_nodes_.clear();
}
```

ÖZGEÇMİŞ

Reşat Andaç ÖZER, Tekirdağ' ın Malkara ilçesinde 19/04/1983 tarihinde doğmuştur. İlkokulu, Malkara Hüseyin Köse İlkoğretim okulunda okul birincisi olarak tamamlamıştır. Ortaokulu, Malkara Anadolu Lisesi'nde okul birincisi olarak tamamlamıştır. Fen Liseleri giriş sınavında Türkiye 124.'sü olmuştur. İstabul Atatürk Fen Lisesi' ne 8. olarak girmiştir ve 4.70 ortalama ile tamamlamıştır. Lisans öğretimini Yıldız Teknik Üniversitesi, Elektronik ve Haberleşme Mühendisliği' nde 3.38 ortalama ile tamamlamıştır. Eylül 2005 tarihinde İstanbul Teknik Üniversitesi Telekomünikayon Mühendisliği Yüksek Lisans programına kabul edilmiştir.

Haziran 2005 tarihinde Nortel Networks NETAŞ Araştırma-Geliştirme Departmanı'nda işe başlamıştır. Burada 4 seneye yakın bir süre telekomünikasyon yazılım geliştirme ve yazılım test departmanlarında uzman mühendis olarak çalışmış, uluslararası pek çok ARGE projesini başarıyla tamamlamıştır. Nisan 2009 tarihinde işyerindeki görevini askerlik gerekçesiyle bırakmıştır. Askerliğini Erzincan 3. Ordu Muhabere Alayı' nda kısa dönem olarak tamamlamıştır.

Nisan 2010 tarihinden itibaren, Alcatel-Lucent Teletaş, Araştırma-Geliştirme Departmanı' nda Wireless Management Solution ve Femtocell Management Solution yazılım geliştirme ve test projesinde uzman arge mühendisi olarak çalışmaktadır.