TÜBİTAK

İşbirlikçi Hücresel Ağlarda Veri Önbellekleme İçin Cihazdan Cihaza İletişim ile Dağıtık Depolama, Optimale Yakın Kodlama ve Protokol Tasarımı:

Program Kodu: 1001

Proje No: 119E235

Proje Yürütücüsü:
Doç. Dr. Şefik Şuayb Arslan

Araştırmacı(lar):
Dr. Öğr. Üyesi Elif Haytaoğlu

Bursiyer(ler):
Massoud Pourmandi
Erdi Kaya

MAYIS 2023
İSTANBUL

Önsöz

Bu projede, hücresel ağlarda veri önbellekleme ve düğüm onarımı gibi konular ele alınmıştır. Proje, özellikle 5G ve ötesi teknolojiler için önem arz eden önbellekleme, akıllı protokol tasrımları ve bunların gerçek zamanlı ortamlarda gerçeklenmesi üzerine yoğunlaşmış ve bu alanda önemli katkılarda bulunmuştur. Tahmin edileceği üzere popüler dosyaların cihazlarda önbelleklenmesi, cihazlar arası etkileşimi azaltarak baz istasyonlarına düşen iletişim yükünü azaltır. Ancak, bazı düğümlerin hücreye katıldığı ve bazılarının ayrıldığı durumlarda, performans için akıllı veri onarım yöntemlerine ihtiyaç duyulur. Bu nedenle, çalışma protokolü kuralları ve iletişim kısıtlamaları gibi faktörlerin yanı sıra bant genişliği ve depolama kapasitesi gibi tasarım değerlendirmeleri de gereklidir. Projemizde, bu konuları ele alan özgün kod yapıları, protokoller ve algoritmalar önerilmekte ve performansları hem analitik olarak hem de nümerik olarak değerlendirilmektedir. Projemiz TÜBİTAK tarafından desteklenmeye uygun bulunmuş ve 39 ay boyunca desteklenmiştir.

Tablo Listeleri

Şekil Listeleri

Özet

Hücresel ağlarda popüler dosyaların cihazlarda önbelleklenmesi ile, cihazlar arası etkileşim baz istasyonu (Bİ) üzerine düşen iletişim yükünü oldukça azaltmaktadır. Dağıtık veri önbellekleme işlemi popüler bir dosyanın parçalarının kodlanmamış orijinal haliyle ya da herhangi bir silinti kodu kullanılarak kodlanmış halinin mobil cihazlar içerisinde dağıtık bir şekilde depolanması yardımıyla gerçekleştirilir. Dosyanın herhangi bir parçası, komşu mobil cihazlardan ya da mümkün değilse, doğrudan Bİ'lerden, yüksek bir iletişim maliyeti pahasına indirilebilir. Bir hücresel ağda, rastgele zamanlarda bazı düğümlerin hücreye katıldığı ve bazılarının ayrıldığı göz önüne alındığında, performans için Bİ ile iletişimin minimum düzeyde olmasını sağlayacak akıllı veri onarım yöntemlerine ihtiyaç duyulacaktır. Tek bir veya birden fazla Bİ'nin sisteme katılımı, önceki onarım paradigmalarına, özellikle de işbirlikçi düğüm onarım süreçlerine farklı bir boyut eklemektedir. Bunun nedeni, çalışma protokolü kurallarının yanı sıra iletişim kısıtlamalarının da değişmesidir. Literatür, bu durum için temel bant genişliği/depolama ödünleşim uzayını inceleyen bir çalışma içermemektedir. Yeni hücre mimarileri buna göre, yeni silinti kod yapılarını, verimli protokol tasarımlarını, veri erişim gecikmesi, gerçekçi kuyruk modelleri ve gerçekçi benzetim platformları dahil ancak bunlarla sınırlı olmamak üzere farklı tasarım değerlendirmelerini gerektirmektedir. Bu projede ilk olarak, daha önceki hiçbir çalışmada düşünülmemiş işbirliği yapılan Bİ'lerin cihaz ayrılışlarında yaşanan kayıp verinin onarımı için bant genişliği ve depolama kapasitesinin iyileştirilmiş teorik sınırlarının veri akış diyagramları kullanılarak elde edilmesi amaçlanmıştır. Bununla beraber, bant genişliği ve depolama alanını en iyi kullanan kod yapılarından esinlenerek, veri önbellekleme işlemini optimale yakın bir maliyetle gerçekleştirecek tamamen özgün çizge tabanlı kod yapıları ve bu yeni kodlar için daha önce düğüm onarım problemine uygulanmamış genetik algoritma, optimize edilmiş artık veri dağıtımı gibi yeni yaklaşımlar kullanılarak önceden düşünülmemiş düğüm onarım algoritmaları önerilmiştir. Ayrıca, düğümlerin hücreye katılma ve ayrılma süreçleri için, bant genişliği ve veri depolaması gereksinimlerini en aza indirmeye yardımcı olacak enerji tüketimi odaklı ve tahmin teorisi merkezli, son derece özgün protokoller önerilmiştir. Bu protokoller, düğümlerin bir hücreden diğerine geçiş yapabileceği ve hücre içi kaynakların etkin kullanılmasına yardımcı olmak için Bİ'ler ile işbirliği yapmasını sağlayan geçiş senaryoları ile güçlendirilecektir. Bu durum, iki onarım işlemi arasındaki sürenin ayarlanması, veri erişim maliyetlerinin azaltılması, hücreye katılan düğüm içeriğinin kullanımı, artık veri kullanımı v.s. gibi yenilikleri içermektedir. Son olarak, önerilen kod yapıları ve protokol mimarisinin performansını analitik olarak türetmek için bilinen çeşitli ve daha gerçekçi kuyruklama modelleri değerlendirilmiştir. Analitik sonuçlarımızı doğrulamak için daha sonra hücresel ağ tabanlı büyük ölçekli benzetimler (ns3) yapılıp sayısal yöntemler ile toplam iletişim ve dosya onarım işlemlerinin maliyet hesaplamaları ve karşılaştırmaları yapılıp, önerilen kod, algoritma ve protokollerimizin verimliliği ispatlanmıştır.

Abstract

In cellular networks, with the popular files being cached, the interaction between devices significantly reduces the communication load on the base station (BS). The caching can be accomplished by distributing the partitions of a popular file to mobile devices either in an uncoded or coded form using erasure coding. Any piece of the chunked content can either be retrieved from the neighboring mobile devices or, if not possible, from the BS directly, at the expense of a higher communication cost. Considering a cellular network, in which a cell contains a set of nodes, some of which are arriving and some are departing at random times, intelligent data repair methods will be needed to ensure a minimum level of communication with the base station (BS). Involvement of a BS or more than one BS adds another dimension to previous repair paradigms, especially to the cooperative repair process due to the changes in the set of constraints and operating protocol rules. There is no work which studies the bandwidth/storage trade-off for this particular case. Accordingly, novel cell architectures will call for different design considerations including but not limited to novel code constructions, protocol designs, data access latency, realistic queuing models and simulation platforms. In this project, unlike the previous research, we initially aim to obtain improved theoretical bounds for the bandwidth and storage capacity using data flow diagrams while BSs are cooperatively helping to repair the lost data. In addition, inspired by the code structures that utilize bandwidth and storage space efficiently, completely genuine graph-based code constructions as well as novel repair algorithms on these codes will be proposed to achieve near-optimality. Novel approaches such as genetic algorithms and optimal left-over data distributions are proposed that have not previously been applied to the node repair problem in literature. Besides that, highly novel protocol designs based on energy efficiency and guessing theory for check-in and check-out processes, are proposed and analyzed, which will help minimize the bandwidth and data storage requirements. These protocols will be strengthened by transition (hand-off) scenarios, which allow nodes to migrate from one cell to another, while enabling the BSs to collaborate and help effectively use the intracellular resources. This involves a few novelties such as adjusting repair intervals (thresholds), reducing data access overheads, the use of incoming node contents, intelligent left-over data handling etc. Finally, various known and more realistic queuing models are used to analytically evaluate the proposed code structures and the performance of the protocol architecture. To validate our analytical findings, we conducted large-scale simulations (using NS3) on cellular networks, compared cost calculations and total communication and file repair operations with numerical techniques, and demonstrated the effectiveness of our proposed codes, algorithms, and protocols.

1. Giriş

Günümüzün veri bilimi ve iletişimi, kablosuz hücrelerin teknolojik olarak sunduğu imkanlar neticesinde ilerleyerek hayatımızı büyük oranda değiştirmiştir. 5G penceresi altında veri hızının artması ve multimedya iletişiminin yüksek çözünürlüklerde mümkün olması insanları farklı platformlarda kaliteli şekilde bir araya getirmeye davet etmektedir. Fakat bunların gerçekleşebilmesi, özellikle verinin boyutu ve akışkanlığının artmasından dolayı büyük depolama alanlarının, verinin akıllı yöntemlerle (enerji/kaynak açısından verimli) yönetilmesini ve korunmasını gerektirmektedir. Buna ek olarak, hücre içi kaynakların çoklu kullanıcı sayısı ve beklenilen hizmet kalitesinin (Quality of Service: QoS) artması doğrultusunda öneminin artması, bu kaynakların mümkün olan en verimli şekilde kullanılmasını gerektirmektedir. Ülkemizde Bilgi Teknolojileri ve İletişim Kurumu'nun (BTK) 5G ve ötesi beyaz kitap yayınında **(BTK, 2020)** ve stratejik belgelerinde de belirtildiği üzere proje konumuz 5G (ve devamı olacak 6G) teknolojisi için büyük önem arz eden şebeke kaynaklarının kullanımı, mobil aygıtların enerji tasarrufu ve nesnelerin interneti ile doğrudan ilgilidir. Bu projemizin amaç ve hedefleri iletişim teknolojimizin bir üst seviyeye çıkmasına yardımcı olmuş ve ülkemizin bu alanda yapacağı yatırım ve üretkenliğine katkılarda bulunduğu söylenebilir.

Diğer öncülleri gibi 5G teknolojisinin en güçlü yanlarından biri kendinden bir önceki nesilden çok daha yüksek bant genişliği sağlayabilmesi olacaktır. Bunun baz istasyonu (Bİ) gibi alansal kaynakların sıklaştırılmasıyla elde edilmesi tasarlanmaktadır. Son kullanıcılar ile baz istasyonları arasındaki uzaklığın azalmasıyla bant genişliğinde artış beklenmekte ve daha iyi bir servis kalitesi elde edilmesi planlanmaktadır. Bu servis kalitesinin artışı her bir son kullanıcı ile baz istasyonu arasında doğrudan bir bağlantı bulunması temeline dayanmakta, sonuç olarak kullanıcı sayısı arttıkça beklenen seviyede veri iletim hızının karşılanmasında sıkıntılar oluşabilmektedir. Ancak, 3G teknolojisi öncülüğünde hücresel ağlarda cihazlar arası baz istasyonu kullanılmaksızın cihazdan cihaza (C2C) doğrudan iletişim gerçekleştirilebilmektedir. Böylece bir cihaz tarafından elde edilmek istenen bir veri yakın bir cihaz içerisinde depolanmış ise baz istasyonu gereksinimi olmadan doğrudan elde edilebilmektedir. Bu yöntemle baz istasyonları ile haberleşmenin azaltılması dolayısıyla daha az iletişim maliyeti daha az enerji tüketiminin oluşacağı ve daha iyi bir servis kalitesi gerçekleştirilebileceği düşünülmüştür. Dolayısı ile yeni jenerasyon sistemlerde birden fazla baz istasyonu ile servis yapılması, hesaplamaların ve depolamanın ağda uç aygıtlara taşınması gibi faktörlerin işin içine girmesi ile sistem tasarımı, optimizasyonu ve servis kalitesi ve güvenliği konularında yeterli araştırmalar yapılmamış durumdadır.

1.1 Altyapı

MEF Üniversitesi öğretim üyesi Doç. Dr. Şuayb Arslan'ın (şu an Profesör) yürütücüsü olduğu ve 36 ay süren projede, Pamukkale üniversitesi Bilgisayar Mühendisliği Bölümü öğretim üyesi Dr. Elif Haytaoğlu araştırmacı olarak görev almıştır. Projede, iki doktora (Massoud Pourmandi - Boğaziçi Üniversitesi ve Erdi Kaya - Ankara Üniversitesi) öğrencisi, ve üç adet STAR programı lisans bursiyerlerimiz çalışmışlardır. Bunlara ek olarak iki farklı sunucu sistemi ve temin edilen laptoplar aracılığı ile altyapı güçlendirilmiş ve projenin başarılı olması için kullanılmıştır (özellikle pandemi döneminde).

1.2 Notasyon ve Organizasyon

Öncelikle, proje önerimizde kullanılacak notasyon açıklanacaktır. İçeriği F kadar sembolden oluşan bir dosya k eşit büyüklükte parçaya bölünüp, n>k parçaya kodlanarak (encoding) genişletilecektir. Kodlama işlemi k x n boyutunda bir üretici matrisi ile gerçekleştirilmektedir. Bu durumda kodun oranı (rate) k/n olarak belirlenir. n sembolden oluşan kodlanmış veri topluluğuna şerit denilmektedir. Dağıtık olarak tutulan parçaların her biri α sembol içerir. Ayrıca h≥k tane parça toplanıp geriçatılarak orijinal F sembollük dosya tekrar elde edilebilir. Burada h indirme yerelliği (download locality) olarak adlandırılmaktadır. Eğer herhangi bir parça kaybolduysa, herhangi r≥1 tane parçanın her birinden β kadar sembol toplanıp kaybolan parça onarılabilir. İşbirlikçi olarak F sembollük dosya (n,h,r,α,β) parametreli kod ile kodlanıp dağıtık olarak m tane hücre düğümü içerisinde depolanmaktadır. Burada m sayısı, n sayısını bölmek zorunda değildir. Tipik örnek senaryolarda m=n olarak alınabilir.

Sonuç raporumuzun ikinci bölümünde literatür özeti verilmiştir. Bu bölüm iki farklı alt bölüme ayrılarak, literatürde nelerin yapıldığını detaylı olarak açıklanarak verilmiştir. Sonraki Gereç ve Yöntem kısmında, sistem modeli, ağa varma-ayrılma modeli, veri onarımı, bilgi akış çizgeleri, optimal baz istasyonu Algoritması, alt-paketleme seviyesi, link Kapasiteleri ve bazı temel bilgi teorisi (tahmin teorisi gibi) teknikleri açıklanmış, yöntemsel olarak nasıl bir farklılık ortaya konulduğu açıklanmıştır. Aynı bölümde protokol tasarımlarımıza yer verilmiş, gelişkin teknikler önerilmiştir. Raporumuzun dördüncü bölümünde bulgular ortaya konulmuş, teorik, nümerik ve simülasyon sonuçlarımız paylaşılmıştır.

2. Literatür özeti

2.1 İyi İndirme/onarım Yerelliği ve Verimli Bantgenişliği ile İlgili Literatür Taraması

Geleneksel iletişim modelinde hücresel ağ içerisindeki veri iletişimi tamamen baz istasyonu üzerinden gerçekleştirilirken, cihazlar arası iletişim modelinde ise ilgili veri hücresel ağdaki kullanıcıların cihazlarının önbelleklerinde saklanabilmekte ve bu cihazlar baz istasyonuna

ihtiyaç duymaksızın birbirleriyle doğrudan iletişim kurabilmektedir **(Asadi vd., 2014)**. Bu nedenle cihazlar arası iletişim, artan veri trafiğine bir çözüm olarak gittikçe daha fazla önem kazanmaktadır.

Hücresel ağ içerisinde cihazların kullanımına dayanan iki ana önbellekleme stratejisi vardır. Bunların ilkinde, dosya kodlanmamış halde saklanmaktadır **(Maddah-Ali ve Niesen,2014)**, **(Shanmugam vd., 2013)**. İkinci stratejide ise daha düşük miktarda önbellekleme alanı kullanımı sağlamak için dosya kodlanmış olarak saklanmaktadır **(Pääkkönen vd., 2015, Pedersen vd., 2016, Pedersen vd., 2018, Piemontese vd., 2018, Calis ve Koyluoglu, 2016)**. Kodlama temelli önbellekleme, silinti düzeltme kodlarına (erasure correcting codes) dayanmaktadır. Bu önbellekleme türünde, veri dosyası silinti düzeltme kodunun kendi yapısına göre kodlanır (encoding) ve orijinal veriye göre genişletilmiş olan yeni veri elde edilir. Kodlanmış bu yeni verinin herhangi bir noktasında belli sayıda hata oluştuğunda, verinin yeniden iletimine gerek kalmadan eldeki fazladan kodlanmış veri çözülerek (decoding) kayıp veri kısmı yeniden elde edilebilir.

Verideki bu fazlalık (redundancy) durumunu oluşturmak için, bir dosya önce k adet veri sembolüne bölünür ve daha sonra n adet sembol olarak kodlanır (n>k). Geleneksel silinti düzeltme kodlarında, n ve k parametrelerinin değerlerindeki bir artış, düğüm onarımında ve kod çözmede karmaşıklığın da artmasına neden olur. Diğer yandan, kod oranı olarak da bilinen k/n oranı sabit tutulup n ve k'yi artırmanın, LDPC **(Gallager, 1962)** kodlarında kod çözme karmaşıklığını etkilemediği görülmüştür **(Yongmei vd., 2015)**.

Önbellekleme alanındaki çalışmalardan birinde, kablosuz ağlardaki popüler video trafiğini azaltmak için femtocaching kavramı önerilip analiz edilmiştir **(Maddah-Ali ve Niesen,2014)**, **(Golrezaei vd., 2013)**. Bu mimaride, küçük hücre erişim noktaları, yani yardımcılar, ilgili video içeriğini yerel önbelleklerinde depolar. Yardımcıların ek yükü, veri aktarım kapasitesini aşabilir ve bir darboğaza (bottleneck) neden olabilir. Bu nedenle yardımcılar, sistem darboğazı olasılığını azaltmak için düşük ana taşıyıcı kapasitesi ve yüksek depolama kapasitesi ile donatılmıştır. Yardımcı düğümlerin önbelleklerinde saklanan veriler kodlanmamış ya da kodlanmış biçimde tutulabilir.

Literatürde yenileme kodları **(Dimakis vd., 2010)** ile Reed-Solomon kodları **(Reed ve Solomon,1960)** gibi çeşitli silinti düzeltme kodlarının kullanıldığı çalışmalar da bulunmaktadır. Bunlardan biri olan **(Pääkkönen vd., 2013)**'teki çalışmada basit önbellekleme, 2-Tekrarlama ve yenilenme kodları test edilmiştir. İlgili çalışmada düğümler hücre içerisine Poisson rasgele sürecine (Poisson random process) göre girip çıkarken onarım işlemleri anlık olarak gerçekleştirilmiş olup düğümlerin enerji kullanım seviyeleri incelenmiştir. Bir diğer çalışmada, yenilenme kodlarının hücresel ağlarda onarım süreci esnasındaki toplam enerji tüketimi

incelenmiştir ve çoklu kayıp veri onarımlarında kodlamaya dayalı önbelleklemenin kodlama olmaksızın yapılan önbellekleme yöntemine göre daha iyi performans gösterdiği görülmüştür (**Pääkkönen vd., 2015)**. Hücresel ağ içerisinde kayıp düğüm onarımını inceleyen bir başka çalışmada ise anlık onarım yerine tembel onarım (lazy repair) kullanılmıştır (**Pedersen vd., 2016)**, (**Pedersen vd., 2018)**. Tembel onarım yönteminde, onarım işlemi sadece belli zaman aralıklarının sonunda gerçekleştirilmektedir.

Bir başka çalışmada, hücrede yalnızca bir düğümün dosyanın tam bir kopyasını saklaması ya da dosyayı isteyen düğümlerin tekrarlama ve yenileme kodlarını (**MBR ve MSR**) kullanarak içeriği önbelleğe alması (çoklu önbellekleme) gibi birkaç farklı önbellekleme stratejisi araştırılmıştır **(Pääkkönen vd., 2018)**. Bu çalışmadaki sistem modelinde, depolama düğümlerinin sayısını sabit tutmak için herhangi bir düğüm arızasında anlık onarım gerçekleştirilmiştir. Bu çalışmada, çoklu önbellekleme yönteminin dosya popülerlik puanı çok yüksekse diğer önbellekleme yöntemlerinden daha iyi performans gösterdiği, dosya popülerlik puanı düşük olduğunda ise tekrarlama yönteminin en iyi performansı gösterdiği gözlemlenmiştir.

Yakın zamanlı bir başka çalışmada, geleneksel MDS kodlarının onarım bant genişliğini iyileştirmek amacıyla yeni bir önbellekleme stratejisi olarak Çift Tekrarlamalı MDS kodları (DR-MDS) önerilmiştir **(Li vd., 2018)**. Geleneksel silinti düzeltme kodları yüksek onarım maliyetine sahiptir. Geleneksel MDS kodlarını kullanarak tek bir kayıp düğümü onarmak için, tüm dosya boyutu kadar içeriğe erişmek gerekir. Bu dezavantajı azaltmak için, tekrarlama ve MDS kodları birleştirilmiştir. Önceki çalışmalardan olan (**Piemontese vd., 2018)'** de, mobil içeriğin önbelleklenmesi sorunu, dağıtılmış depolamada MDS kodları kullanılarak incelenmiştir. Diğer çalışmalardan farklı olarak kullanıcıların popülerliklerine göre birden fazla dosya talep ettikleri senaryo incelenmiştir.

Verinin yeniden kullanımı açısından dağıtık depolama ile dağıtık önbellekleme arasında yakın bir ilişki bulunmaktadır. Örneğin, **(Wei vd., 2014)**'de LDPC kodları dağıtık depolama için kullanılmıştır. Bu çalışmada, kayıp sembollerin birer birer onarıldığı onarım yöntemi farklı LDPC kodları kullanılarak gerçekleştirilmiştir. Daha sonra da bu onarım süreci tekrarlama, Reed-Solomon ve LRC **(Papailiopoulos ve Dimakis, 2014)** kodları ile karşılaştırılmıştır. Bir başka çalışmada, LDPC kodlarının performans değerlendirme süreçlerinde kullanılan onarım bant genişliği, güvenilirlik ve depolama ek yükü gibi anahtar kriterler arasındaki değiş tokuş durumu analiz edilmiştir **(Park vd., 2017)**. Reed-Solomon kodlarından farklı olarak LDPC kodlarının onarım bant genişliğinin, kod uzunluğu artsa bile değişmediği gösterilmiştir. Bu çalışmaya göre, normal LDPC kodlarının ortalama anlamda optimal olduğu ve Reed-Solomon kodlarına göre daha düşük bant genişliği ve daha yüksek güvenilirlik sağladığı görülmüştür.

Tablo 1'de kodlamaya dayalı önbellekleme paradigması temelinde yapılan çalışmalar sınıflandırılmıştır.

Tablo 1: Kodlamaya dayalı önbellekleme çalışmaları

Kodlama Sistemi	Önbellekleme Metodu	Onarım Stratejisi	Depolanan Dosya	Amaç
Maddah et al. (2013)	Kodlanmamış Önbellekleme	Uygulanamaz	Çoklu Dosyalar	Global önbellekleme ile yerel önbellekleme karşılaştırması
Shanmugam et al. (2013)	Kodlanmamış Önbellekleme ve Fountain Kodlu Önbellekleme	Uygulanamaz	Çoklu Dosyalar	Optimum dosya yerleştirme yoluyla gecikmeyi en aza indirme
Paakkonen et al. (2013)	Basit Önbellekleme, 2-Tekrarlama	Anlık Onarım	Tek Dosya	Enerji tüketimini en aza indirmek
Paakkonen et al. (2015)	Basit Önbellekleme, Yenileme Kodları	Bir düğüm bir dosya talep ettiğinde gerekliyse gerçekleştirilir.	Tek Dosya	Enerji tüketimini en aza indirmek
Pedersen et al. (2016)	Reed-Solomon, Tekrarlama, Yenileme Kodları, LRC Kodları	Tembel Onarım	Tek Dosya	İletişim Maliyet Analizi
Pedersen et al. (2018)	Optimize Edilmiş MDS Silinti Düzeltme Kodları	Bir düğüm bir dosya talep ettiğinde gerekliyse gerçekleştirilir.	Çoklu Dosyalar	İçeriğin önbelleklenmesini optimize etme
Piemontese et al. (2018)	MDS Silinti Düzeltme Kodları	Tembel Onarım	Çoklu Dosyalar	Gecikme etkilerine sahip parametrelerin analizi
Pääkkönen et al. (2018)	Basit Önbellekleme, Çoklu Önbellekleme, Yenileme Kodları	Anlık Onarım	Tek Dosya	İletişim ve depolama için tüketilen enerjinin en aza indirilmesi
Li et al. (2018)	Çift MDS Kodları	Tembel Onarım	Tek Dosya	Hücresel ağlarda önbellekleme için yeni hibrit kod yapısı

2.2 Dağıtık Depolama İçin Bilgi akış ve Ağ kodlama Çalışmaları Literatür Taraması

Son zamanlarda yapılan çalışmaların büyük çoğunluğu kümelenmiş (clustered) dağıtık depolama sistemleri için düşünülmüş ve bu merkezde farklı kodlama teknikleri üretilmiştir.

Projemizin yazım tarihinden bu yana (2018 sonrası) yapılan çalışmaların en önemlilerini bu bölümde özetleyeceğiz.

Bu çalışmaların ilki olan **(Sohn vd., 2018)**'de kümeleme yapısı ortaya atılmış ve her bir küme farklı maliyet faktörleri (indirilen veri boyutu) ile ilişkilendirilmiştir. Bu veri boyutu tabanlı maliyet tam olarak bizim maliyet tanımımızın özel bir durumu olsa da bu çalışmanın ele aldığı problem bazı yönleriyle bizim projede bahsini ettiğimiz problemle benzerlik teşkil etmektedir. Fakat bu çalışmada, tüm kümelerde onarım yapılabilmesi, aslında önerilen yapının lokal kümenin katman katman düşünüldüğünü ortaya çıkarmaktadır. Bizim önerdiğimiz modelde onarım yalnızca yerel kümede merkezsiz şekilde gerçekleştirilirken, farklı seviyelerdeki düğümler sadece yapılan onarıma dışarıdan katılarak yardımcı olabilecek veri aktarımını sağlayabilirler. Daha sonraki buna bağlı çalışmalarda **(Sohn vd., 2018) (Chen vd., 2019) (Tebbi vd.,2019) (Hou vd.,2019)**, küme-içi ve küme-dışı ayrımı yapılmış, küme içerisinde aynı dağılımı esas almıştır. Fakat bu çalışma herhangi bir işbirlikçi yapıyı sistemin parçası olarak düşünmemiştir. Ayrıca bizim çalışmamızdaki küme-içi ve dışı bulunan düğümler için farklı maliyet düşünülebilir. Bazı sonraki çalışmalarda **(Wang vd.,2019)** ayrık düğümler (separate nodes)'in kümelenmiş depolama sistemleri için kapasiteyi düşürdüğü gözlemlenmiştir.

Tablo 2: Ağ kodlama ve bilgi akiş diagrami çalişmalari

Kodlama Sistemi	Kümeleme	İşbirlikçilik	Merkezi Onarım	Hierarşik Yapı	Onarım yeri	Temel Farklılık
J. Sohn, et al. (2018,2019)	Evet	Yok	Yok	Çoklu düğüm kaybı	Her bir küme	Veri depolama sistemleri, kısıtlı akış diagramı
Chen and Barg (2019)	Evet	Yok	Yok	Yok	Her bir küme	Kod oluşturmaya odaklanma
Tebbi et all, (2019)	Evet	Yok	Yok	Var	Her bir küme	Kod oluşturmaya odaklanma, lokal onarım.
Hou, (2019)	Evet	Yok	Yok	Var	Her bir küme	Maliyet fikri olsa da aynı değil.
Wang (2019)	Evet	Yok	Yok	Var	Lokal küme	Extra ayrık düğüm eklenmiş Kümelenmiş depolama
Prakash, (2018)	Evet	Yok	Yok	Var	Her bir küme	Veri depolama sistemleri,tek düğüm tamiri, maliyet fikri açık değil.

Yu et al, (2019)	Evet	Yok	Yok	Var	Lokal küme	Kooperasyon ve maliyet fikri eksik.
Liu et al (2020)	Yok	Var	Yok	Yok	Lokal küme	Performans düşüşüne rağmen kısmi onarım düşünülmüş ve MSR, MBR noktalarına odaklanılmıştır.
Zorgui et al (2019)	Yok	Yok	Var	Yok	Lokal küme	Önceki bir çok dağıtık onarıma kıyasla bu çalışma merkezi mimari düşünmüştür. Fakat bu durumda sadece MSR ve MBR noktaları formülize edilmiştir.

Bu çalışmaların paralelinde (mesela **(Prakash vd.,2018)**) kodlanmış veri farklı küme yapılarının tamamı üzerine dağıtılırken, küme-içi ve küme-dışı veri transferlerinin maliyetleri farklı olarak kabul edilmiş, işbirliği hem lokal hem de kümeler (cluster) arasında temel bir bileşen olarak düşünülmemiştir. Yine bu çalışmaya benzer çalışmaların bir kısmında ise hiyerarşik yapı veri depolama sistemleri bağlamında düşünülmüş (mesela **(Yu vd., 2019)**), heterojen onarım işin içine katılmış ve kapasite hesaplamaları yapılmıştır. Fakat işbirlikçi yapı ve maliyet konusuna detaylı yer verilmemiş ve dolayısı ile üretilen sonuçlar gerçekçi veri depolama sistemleri için sınırlı kalmıştır. Son dönemlerde kısmi onarım **(Liu vd., 2020)** ve merkezi mimari **(Zorgui vd., 2019)** kullanılarak onarım gibi teknikler de önerilmiştir. Bunlarda kısmi onarım kümelenmiş veya onarıma katılan düğümlere erişimin hazır olmaması durumlarını işin içine katarak kısmi yardım talebi altında kapasiteyi incelerken, merkezi mimari yapısının onarım için gereken bant aralığının gözlemlenebilir şekilde azaltılabileceğini göstermiştir. Fakat bu yöntemler merkezi olmayan tam onarım gerektiren bizim düşündüğümüz hücresel ağlar için pek uygulanabilir değildirler. Ayrıca, bizim çalışmamızda maliyet unsuru ile onarımın kısmileştirilmesi optimizasyon problemi ile aslında gerçekleştirilmektedir Bu çalışmalar ile bizim önerdiğimiz mimari yapısal diğer önemli farkı ise depolama sınırlamalarından dolayı her katmanda dosyanın sadece bir bölümünün tutulmasıdır. Dolayısı böyle bir kısıt elimizdeki optimizasyon problemini de değiştirecektir. Bu çalışmalar ve bunlara benzer diğerleri Tablo 2'de özetlenmiş, bizim çalışmamız ile temel farklılıkları belirtilmiştir.

3. Gereç ve yöntem

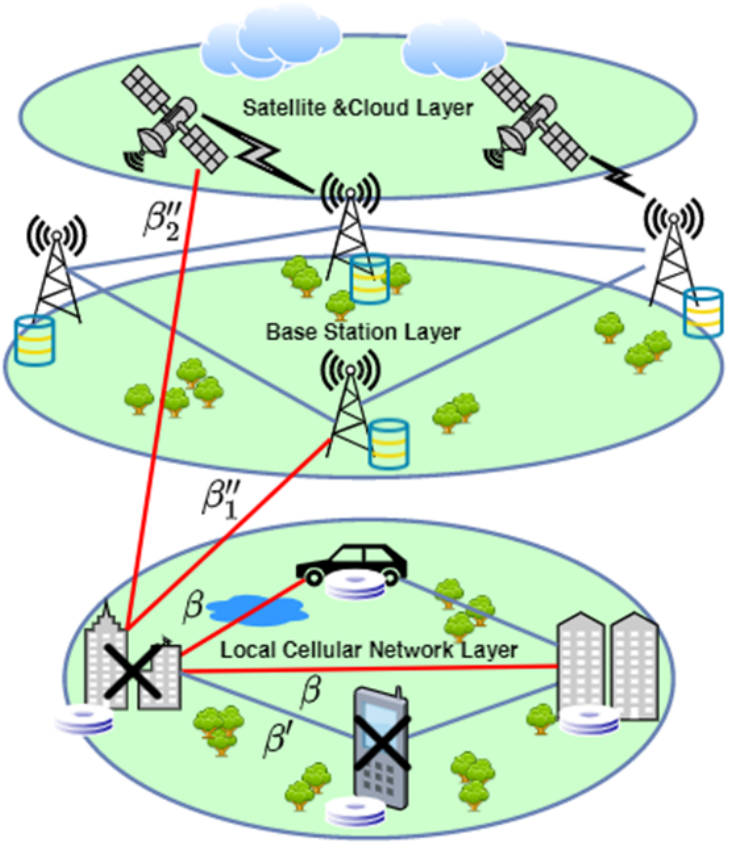
Bu bölümde projemizde kullanılan temel bilgiler, gereçler ve yöntemler özetlenmiştir.

3.1 Sistem Modeli

Hücre içerisinde başlangıçta toplamda N adet düğümün bulunduğu ve düğümlerin hücre içine girme ve dışına çıkma durumları olduğu varsayılmaktadır. Hücredeki düğümler birbirleri ile D2D iletişim yolu ile mesaj gönderilebilmektedir. Hücrede F byte boyutunda tek bir dosyanın hücredeki bazı düğümler tarafından ön-belleklendiği varsayılmıştır. Daha açıkça belirtilirse, $m<N$ olmak üzere m adet düğümün bu dosya ile ilgili kodlanmış parçaları depoladığı varsayılmaktadır. Bu dosya kodlama işleminden önce k adet parçaya bölünmekte olup bu parçalar silinti kodlama ile kodlanarak $n>k$ olmak üzere n adet parçaya genişletilmiştir. Son olarak bu n parça, m adet depolama düğümüne bölüştürülerek depolanmaktadır.

LDPC kodlarının kullanıldığı sistemler dışında her düğüm n/m adet sembol depolanmaktadır ve m|n olarak kullanılmıştır. Ancak **(Eleftheriou ve Olcer, 2002)**'da belirtilen LDPC kodu oluşturma kısıtları nedeniyle LDPC kodlarının kullanıldığı sistemlerde, düğümler $\lceil n/m \rceil$ adet sembol depolanmaktadır. Ek olarak, orijinal dosyanın F byte'ı ve kodlanmış tüm n sembolün baz istasyonunda da bulunduğu düşünülmektedir. Aynı zamanda hücreye yeni gelen düğümlerin önbelleklenmiş dosya ile ilgili bir bilgisi bulunmamaktadır. Bu bölüm boyunca verilen semboller **Tablo 3**'de açıklanmıştır.

Projemizde hiyerarşik baz istasyonu (istasyonları) düşünerek, her bir alt seviye üst seviye için bir önbellekleme görevi gördüğü ve dolayısı ile bir dosyanın tamamının sadece en üst seviyede tutulduğunu, alt seviyelerde ise dosyanın sadece bir parçasının tutulduğunu varsaydığımız özellikle 5G mimarisi ile daha uyumlu bir sistem varsayıyoruz. Bu sistemin iki katmanlı bir örnek gösterimi **Şekil 1**'de verilmiştir. Bu şekilden de anlaşılacağı üzere lokal hücresel ağdaki tamir gerekiyor ise üst katmanların da yardımı ile gerçekleştirilebilmektedir. Fakat tabi üst katmanlar (baz istasyonu ya da bulut katmanı) ile olan iletişimin maliyeti ve çekilen veri oranı farklılıklar göstermektedir. Her katmanın kendine özel depolama oranı üzerinde bir kısıtı olduğunu düşünerek ve tutulan dosyaların bir kopyasının üst seviyede bulunduğunu göze önüne alınmıştır.



Şekil 1. Hücresel iletişim sistemlerinde çok katmanlı (M=2) hiyerarşik depolama.

3.2 Varma-Ayrılma Modeli

Sistemin ilk anında hücre içerisinde adet düğüm olduğunu varsayalım. Ancak zaman içinde gerçekleşen düğüm ayrılışları ve düğüm gelişleri neticesinde $N_{iç}$ yani hücre içerisindeki düğüm sayısı değişmektedir. Düğüm ayrılışları ve varmaları **(Pedersen vd.,2015, Pääkkönen vd., 2013)** çalışmasında olduğu gibi Poisson sürecine dayanmaktadır. Düğümler hücre dışından içerisine λ oranında girmektedir, bu orana geliş oranı denmektedir. Buna göre iki düğümün art arda hücre içerisine gelme anları arasındaki fark $f(x)=\lambda e^{-\lambda x}$ olasılığı ile hesaplanmaktadır. Benzer şekilde, düğümlerin hücre içerisinden çıkma oranları μ ile temsil edilebilmekte ve hücre içinden dışarıya art arda çıkan iki düğüm arasındaki sürenin olasılığı f(y)=μe^(-yμ) ile hesaplanmaktadır. Buna göre bir anda hücre içerisinde x adet düğümün bulunması olasılığı p(N_iç=x)=Nλ/μ.. Hücre içerisindeki depolama düğümü sayısının sabite yakın tutulabilmesi için geliş oranı ve ayrılış oranları aynı olarak seçilmiştir.

Tablo 3: Semboller ve Anlamları

Sembol	Anlamı	Sembol	Anlamı
m	Hücre içerisindeki depolama düğümü sayısı	N	Toplam düğüm (cihaz) sayısı
ρ_{D2D}	Bir sembolün düğümden indirilme maliyeti	ρ_{Bs}	Bir sembolün baz istasyonundan indirilme maliyeti
$N_{iç}$	Hücre içerisindeki toplam düğüm sayısı	a	Şimdiye kadar tamir edilmiş sembol sayısı
F	Orijinal dosya boyutu	l	Sistemdeki toplam kayıp sembol sayısı
Δ	Düğüm tamiri işlemleri başlatılmadan geçen süre	p	Bir düğümün sistemde kalma olasılığı
B_1	MBR kodunda bir şerit (*stripe*) için kodlanacak orijinal sembol sayısı	B_2	MSR kodunda bir şerit (*stripe*) için kodlanacak orijinal sembol sayısı
r^z	Yüksek kod oranına sahip MSR kodlarında bir düğümün depoladığı sembol sayısı	$q(a)$	a adet sembol tamir edildiğinde aynı kayıp düğümdeki daha tamir edilmemiş bir LDPC kodu sembolünün kayıp olma olasılığını veren fonksiyon.

3.3 Düğüm Tamiri

Bir depolama düğümü hücreden ayrıldığında, düğümde depolanan önbelleklenmiş veri kaybedilmektedir. Aynı dayanıklılık (realibility) seviyesinin devamının sağlanabilmesi için bu kayıp verinin tamir edilmesi gerekmektedir. Sistemde tamir etme işlemleri belirli aralıklarla gerçekleştiği tembel tamir etme yöntemi **(Bhagwan vd., 2004)**'de anlık tamir etme yöntemine bir alternatiftir. Bu aralık Δ olarak temsil edilmiştir. Bu noktada tamir etme işlemlerinin belirli periyotlar sonunda gerçekleştiği strateji seçilmiştir. Ayrıca, tamir işlemleri boyunca düğüm varış ve ayrılış olaylarının olmadığı varsayılmaktadır. Çıkan depolama düğümlerinin depoladığı içeriklerin aynen yeniden elde edilip depolanması için boş düğümlerden biri rastgele seçilmektedir. Bu düğüm tamir işlemi ardından kayıp verinin depolandığı bir depolama düğümü haline gelmektedir. Bir düğümde aynı şerite (stripe) ait birden fazla sembol depolandığı için kayıp veriyi elde etmek çoklu veri çözme işlemi yapmayı gerektirebilir. Bunun yanı sıra, Δ aralığı boyunca çoklu düğüm ayrılışları gerçekleşebilmektedir. Bu düğümlerin tamir edilmesi işlemlerinde birbirleri ile iletişimde bulunmadıkları yani işbirliği halinde olmadıkları tamamen dağıtık bir strateji incelenmiştir.

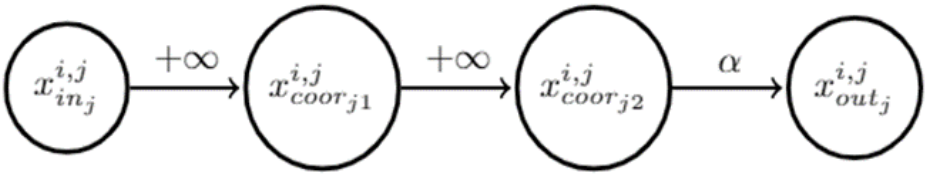
3.4 Bilgi Akış Çizgeleri (Information-Flow Diagramları)

Bilgi akış grafiği, bazı bilgi kaynaklarının (**Şekil 2**'de görülen S ile gösterilen düğüm) ara cihazlar aracılığıyla bir dizi hedef havuza (DC) yayın yaptığı bir ağın grafiksel gösterimidir. Bu nedenle, bilgi akış grafiği, (1) kaynak düğümler, (2) ara düğümler ve (3) havuz düğümleri olmak üzere üç tür düğümle tanımlanabilir. Genel olarak, tek kaynaklı çoklu havuz düğüm problemi durumunda, ağın ulaşılabilir kapasitesi, maksimum-akış minimum-kesme teoremi ile karakterize edilir. Genel olarak çoklu kaynak düğümleri durumunda, erişilebilir kapasite hala açık bir sorundur. Öncü bir çalışma olarak, Alswede ve ark. minimum-kesme akışını ağ kodlamasıyla (network coding) karakterize etmiş ve doğrusal ağ kodlarının tek bir kaynakla çok noktaya yayın probleminde minimum-kesme sınırını elde etmek için yeterli olduğunu kanıtlamıştır.

Dağıtık bir depolama sisteminde, bir S kaynağından, ara düğümler ve havuz düğümlerinden oluşan yönlendirilmiş döngüsel olmayan grafiklere bir bilgi akış grafiği G denir. Veri toplayıcıları (havuz) DC_j depolanmış dosyanın tam olarak kurtarılması için herhangi k cihazla (düğümle) bağlantı kurar. **(Shum, 2011)** 'e benzer şekilde, önerdiğimiz bilgi akış grafiğimizi Şekil 5'de gösterilen s=-1 aşamasından başlayarak (n,k,d,t,α,γ) şeklinde ve farklı n onarım aşamaları ile dinamik karakterize ediyoruz. Ayrıca, belirli bir kod parametre seti için (n,k,d,t,α,γ), tüm olası grafikleri G(n,k,d,t,α,β,β',{β_l",l=1,...,M}) ile gösteriyoruz. Burada,

- F boyutundaki veri dosyası, s = -1 aşamasında S kaynak düğümü tarafından temsil edilmektedir.
- İlk depolama düğümleri, s = 0 aşamasında (In_1, Out_1) çiftleriyle temsil edilir. Yine s > 0 içim, bu çift arasına Coop_1'i ve Coop_2'i gibi düğümler konularak iki farklı kooperasyon temsil edilebilir.
- Veri toplayıcılar - havuzlar (DC), s>0 aşaması ile tek düğüm ile temsil edilir.

Çalışmamızda, her bir ikincil düğüm M + 2 yatay düğümler ile temsil edilir, burada M farklı iş birliği türlerinin sayısını belirtir. Ayrıca, çok katmanlı bir depolama ortamında, her bir küme katmanı bir q sayısıyla indekslenir ve içinde n_q kadar düğüm olduğu varsayılır. Mesela, q = 0 değerinin yerel düğümlerin yaşadığı yerel hücresel ağ katmanına karşılık geldiğini varsayıyoruz. S kaynak düğümünden her bir depolama düğümüne α sembol gönderilerek depolanması sağlanır. Ayrıca dosyanın tamamı olan F sembol de hiyerarşinin en üst katmanına yerleştirilir. Çoğu zaman bu çevrimdışı gerçekleşir.



Şekil 2. Baz istasyonları düşünüldüğünde (M=2) kooperatif onarım için bir düğümün ifadesi.

Onarım yönteminde, ilk veri onarım aşaması, d kadar diğer yerel canlı düğümlerle iletişim kurar ve her bir düğümden β sembol indirmeyi içerir. Bir sonraki aşamada, onarım düğümü M üst katman ile bağlantı kurar (içlerindeki tüm n_1 düğümleri kapsayarak) ve $\beta_l^{''}$ sembollerini l katmanından indirir. Burada *l*-inci katman yalnızca F_l sembol tutarken j-inci düğüm sembol indirme başına w_l^j maliyetiyle ilişkilendirilir. Son aşamada, tüm onarım düğümleri $\beta^{'}$ sembollerini aynı yerel hücre içindeki en fazla t-1 diğer onarılan düğümlerden indirerek ortak bir yerel koordinasyona tabi tutulur. Onarım düğümü bütün bu süreçten sonra α kadar sembolü üreterek depolar. Depolanan dosyanın yeniden oluşturulması için, Veri Toplayıcı (DC) adlı bir tepe noktası eklenir. Yeniden oluşturma işlemi için tane "out" düğümlerine bağlanılıp, veri çekilerek gerçekleştirilir.

Bir bilgi akış grafiğinin kesimi (*cut*) $(U,\underline{U})$, grafikte bulunan düğümlerin S∈U ve DC∈$\underline{U}$ olacak şekilde ikiye bölünmesi ile oluşur. Her kesim tipik olarak *U* içindeki düğümler ve $\underline{U}$ içindeki düğümler arasındaki yönlendirilmiş kenarların ağırlığının kapasite eşdeğeriyle ilişkilidir. Tek-kaynaklı çok-noktaya yayın problemi için maksimum akış minimum kesim sınırı, minimum kesim kapasiteleri *C*'den büyük değilse, kaynaktan her veri toplayıcıya gönderebileceğimiz veri miktarı da 'den fazla olamaz. Demakis *at al.* **(Dimakis vd., 2010)**, dağıtılmış depolama sistemlerindeki onarım sorununun çok noktaya yayın sorunuyla eşleştirilebileceğini fark etmiştir. Dağıtılmış depolama sistemleri ile ilgili bu konudaki tüm önceki çalışmaların bu genel muamelenin özel bir durumu olduğunu fark ederek, çalışmalarımızı bu bilgi akış grafiği üzerine yoğunlaştırmış durumdayız. Örneğin, M=2 ile bir düğümün/cihazın tipik bir temsili, Şekil 1 de gösterilmiş ve bir onarım senaryosunda bilgi akış grafiği içerisinde kullanılarak Şekil 2 oluşturulmuştur. Akış grafiği *G*, zaman içerisinde hatalar oluşup onarımlar gerçekleştikçe evrimleşerek gelişir ve büyür. Hatalı olan ya da hücreden ayrılan düğümle pasif hale gelirler, yeni düğümler grafiğe eklenerek onarım gerçekleştirilir. Onarım olan düğümler pasif düğümlere bağlanmazlar.

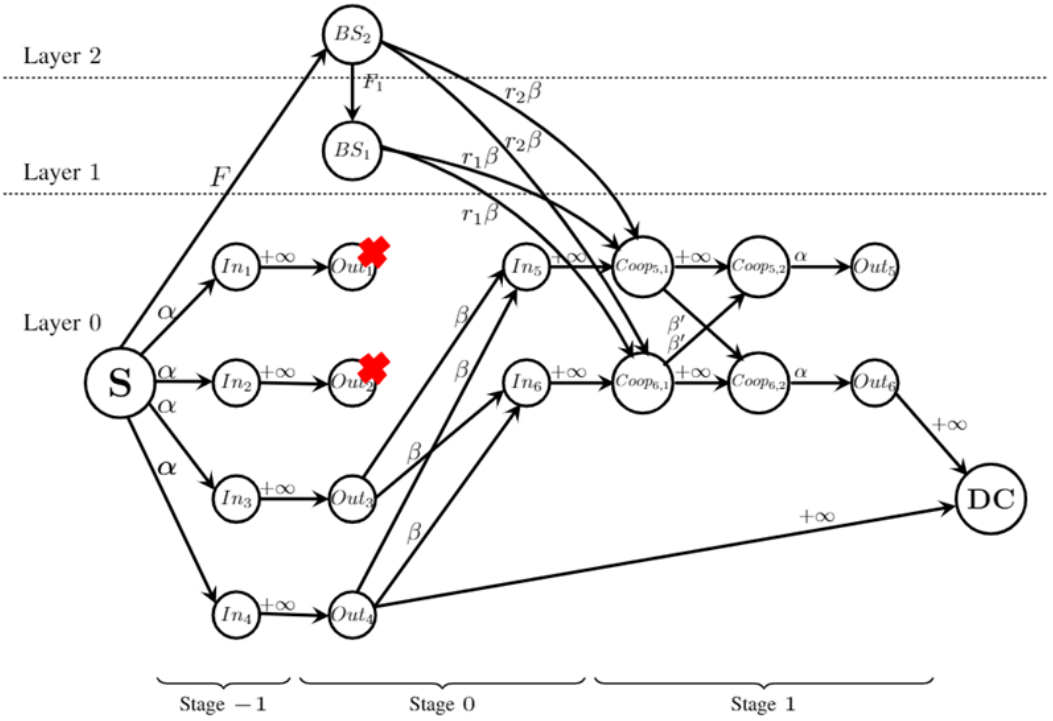
Depolama düğümleri ve çoklu üst katmanlara sahip yerel bir hücrenin, her biri n_l, l = 1, .., M düğüm içermek şartı ile, M tane kümeden oluştuğunu ve her birine bir maliyet faktörü atandığını varsayıyoruz. Kümelerin maliyet faktörlerine göre yığıldığını varsayıyoruz, bu nedenle 0 etiketli küme yerel kümeye karşılık gelirken, M etiketli küme, en yüksek maliyet faktörüne sahip hiyerarşide en yukarıda bulunan baz istasyonu kümesine tekabül eder. Ayrıca, her bağlantının sınırlı kapasiteye sahip olduğunu varsayıyoruz, yani l'inci katmandaki baz istasyonlarından indirilebilecek sembol sayısı $\beta_l^{''} \leq b_l$ β şeklinde sınırlandırılmıştır.

Bilgi akış grafiği, bazı bilgi kaynak düğümlerinin (Şekil 3'de görülen S ile gösterilen düğüm) ara cihazlar aracılığıyla bir dizi hedef veri toplayıcıları (DC ile gösterilmiştir) yayın yaptığı bir ağın grafiksel gösterimidir. Bir bilgi akış grafiği, (1) kaynak düğümler, (2) ara düğümler ve (3) veri toplayıcıları olmak üzere üç tür düğümle tanımlanabilir. Genel olarak, tek kaynaklı çoklu çoklu veri toplayıcı problemi durumunda, ağın ulaşılabilir kapasitesi, maksimum-akış minimum-kesme teoremi ile karakterize edilir. Öncü bir çalışma olarak, Alswede v.d. minimum-kesme akışını ağ kodlamasıyla (network coding) karakterize etmiş ve doğrusal ağ kodlarının tek bir kaynakla çok noktaya yayın probleminde minimum-kesme sınırını elde etmek için yeterli olduğunu kanıtlamıştır **(Ahlswede vd., 2000)**.

Onarım konfigürasyonumuzda, α kadar sembolün ilk veri onarım aşaması, diğer yerel canlı düğümlerle iletişim kurmayı ve β simgelerin indirilmesini içerir. Bir sonraki aşamada, M onarım düğümü katmanlarına (içlerindeki tüm düğümler) ulaşarak, l'inci katmandan toplam β_l^'' sembollerini indirir. Böylece l'inci katman F_l sembol içerirken, bu katmandaki j'nci düğümden bir sembol indirme maliyeti w_l≥1 olarak kabul edilmiştir. Son aşamada, tüm onarım düğümleri aynı yerel hücre içindeki en fazla (t-1) diğer başarısız düğümlerden semboller indirerek birleşik yerel koordinasyonla onarım tamamlanır.

Dağıtık bir depolama sisteminde, bir S kaynağından, ara düğümler ve veri toplayıcılardan oluşan yönlendirilmiş döngüsel olmayan grafiklere bir bilgi akış grafiği G denir. Veri toplayıcıları (havuz) DC_j depolanmış dosyanın tam olarak kurtarılması için herhangi k cihazla (düğümle) bağlantı kurar. Burada **(Shum ve Hu, 2013)**'e benzer şekilde, önerdiğimiz bilgi akış grafiğini Şekil 1'de gösterilen s = -1 aşamasından başlayarak (n, k, d, t, α) şeklinde ve farklı n onarım aşamaları ile dinamik bir yapı olarak karakterize ediyoruz. Ayrıca, belirli bir kod parametre seti için (n, k, d, t, α, γ), tüm olası grafikleri G(n, k, d, t, α, β,β',{ $\beta_l^{''}$,l=1,...,M}) şeklinde gösteriyoruz. Burada,

- F boyutundaki veri dosyası, s = -1 aşamasında S kaynak düğümü tarafından temsil edilmektedir.
- İlk depolama düğümleri, s = 0 aşamasında (In_i,Out_i) çiftleriyle temsil edilmiştir. Yine s > 0 için, bu çift arasına Coop_(i,1) ve Coop_(i,2) gibi düğümler konularak iki farklı kooperasyon temsil edilebilir.
- Veri toplayıcılar - havuzlar (DC), s>0 aşaması ile tek düğüm ile temsil edilir.



Şekil 3: M=2 için örnek bir bilgi akış diyagramı.

Genelliği kaybetmeden, her katmanın tek bir baz istasyonu içerdiğini varsayalım. Ayrıca, önceki çalışmalara [3] benzer şekilde, $\beta_l^{''}$ terimini β cinsinden $\beta_l^{''} = r_l\beta$ şeklinde ifade edebilmek için yardımcı değişkenleri (r_l) ekledik. Burada verili bir reel sayı için b_l , $0 \leq r_l \leq b_l$ şeklinde bir ilişki ile sınırlandırıldığını düşünüyoruz.

Yaptığımız yeni karakterizasyon neticesinde, her bir onarım gören düğüm başına düşen toplam onarım bant aralığı maliyeti şu şekilde elde edilecektir,

$$\gamma_c(\mathbf{s}) = d\beta + (t-1)\beta' + \sum_{l=1}^{M} s_l w_l r_l \beta$$

Burada $\gamma_c(\mathbf{s})$ bant aralığı masraf fonksiyonunu belirtir ve s={s_l} herhangi verili bir tam sayı 0≤ρ≤M için,

$$\mathbf{s} = \left\{ (\underbrace{1,...,1}_{\rho}, \underbrace{0,...,0}_{M-\rho}) : \rho = 0,...,M, \sum_{l=1}^{M} s_l = \rho \right\}$$

şeklinde tanımlanmıştır. Bu vektördeki her bir "0"' girdisinin onarıma katılmayan, "1" girdisinin onarım sürecine katılım gösteren baz istasyonlara denk geldiğini söyleyebiliriz. Her ne kadar 1'ler s vektörünün ilk ρ girdisinde gösterilse de aslında ilk minimum pindirme maliyetine denk gelen girdilerde 1'ler bulunmaktadır. Burada örtük şekilde indirme sembol maliyetlerinin w_(l+1)≥w_l ilişkisini sağladığı aslında farz edilmiştir. Burada **s** kaynağı ile veri toplayıcı DC arasındaki maksimum akışa (max-flow) göre çok katmanlı işbirlikli onarım sürecindeki dosya boyutu için G grafiğinin minimum kesimini bulabiliriz. Bu raporda farklı olarak bu kesimi öncelikle aşağıdaki gibi ifade edilebileceğini göstermiş durumdayız:

$$\min_{r_l, \mathbf{u} \in P} \left\{ u_0\alpha + \sum_{j=1}^{g} u_j(d' - \sum_{i=0}^{j-1} u_i)\beta + u_j\,(t - u_j)\,\beta' \right\} \geq F$$

Burada P={u=(u_j)_(1≤i≤g):1≤u_i≤t ve ∑_(i=1)^g u_i =k}. Başka bir deyişle, u_i, onarım sırasında her bir t'lik onarım grubunda temas edilen düğüm sayısıdır ve g onarım aşamalarının sayısıdır. Ayrıca d'=d+∑_(i=1)^M(s_l r_l) şeklinde verilebilir. Dikkat edilmelidir ki burada d', d gibi bir tam sayı olmak zorunda değildir. Son olarak, **(Shum,2013)**'deki Teorem 5'inin uygulanması ile g=0,1,2,...,k değerleri için aşağıdaki dosya üzerindeki üst sınırlar bulunabilir,

$$\alpha(k-g) + g\beta(d' - k + \frac{g+1}{2}) + g\beta'(t-1) \geq F$$
$$\alpha(k-g) + \beta(g(d'-k) + \frac{g^2 + \psi_{g,t}}{2}) + \beta'(gt - \psi_{g,t}) \geq F$$

ve ψ_(g,t)=⌊g/t⌋t^2+(g-⌊g/t⌋t)^2.

Belirli bir dosya boyutu F ve sistem parametreleri için α, d, k, t, **w** =(w_l) ,1≤l≤M, **b** = (b_l), bant aralığı maliyet-depolama ödünleşim uzayı çözümü aşağıda belirtilen kısıtlı optimizasyon problemi ile verilir,

$$\min_{\mathbf{s}, r_1, ..., r_M, \beta, \beta'} \left\{ d\beta + (t-1)\beta' + \sum_{l=1}^{M} s_l w_l r_l \beta \right\}$$

Bu problem yukarıda belirtilen kısıtların yanında aşağıdaki kısıtlara da tabidir.

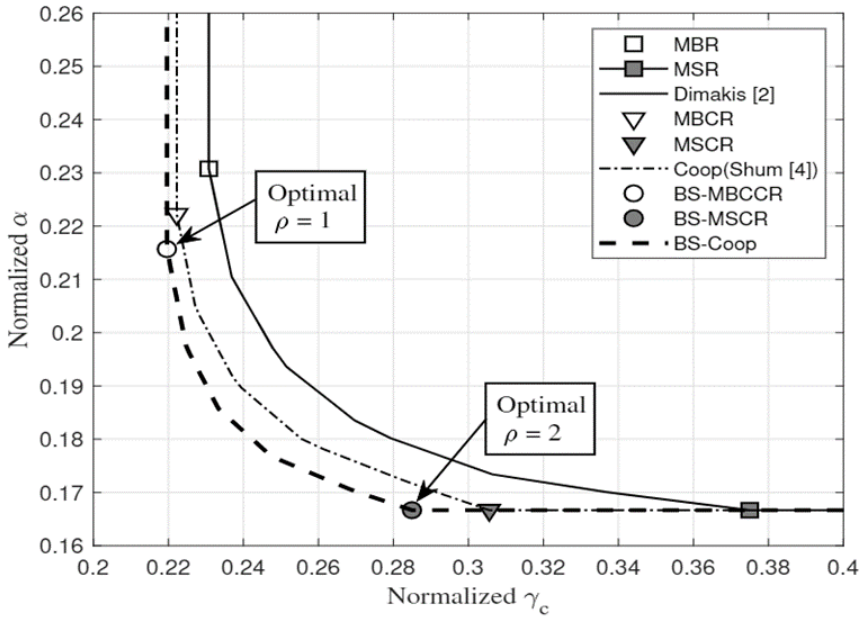
$$0 \leq \beta, \beta' \leq \frac{F}{d} \;\; b_l \in \mathbb{R}, r_l \in [0, b_l], \quad l = 1, ..., M \;\; \mathbf{s} = [s_1, ..., s_M], s_l \in \{0, 1\}$$

Bu probleme analitik çözüm bulmak zorluklar ihtiva etmektedir. Özellikle nümerik lineer-olmayan yazılımsal çözücüler ile problemimizi çözmüş durumdayız. Aşağıda (Şekil 2'de görülen) bir örnek nümerik sonuç görülmektedir. Bu sonuçta literatürde yaygın olarak bilinen iki farklı sistem ile baz istasyonlu (4 adet) işbirlikçi sistemin oluşturdukları ödünleşim uzayları karşılaştırılması gösterilmektedir. Ayrıca bütün durumlar için minimum depolama ve bant aralığı noktaları da işaretlenmiştir. Görüleceği üzere baz istasyonlu senaryo için bu noktalar (BS-MSCR ve BS-MBCCR) da optimal sayıdaki baz istasyonları da bulunup gösterilmiştir. Mesela BS-MBCCR noktasında ρ=1 optimal iken MSCR noktasında ρ=2 optimal olmuştur. Son olarak ifade edilmesi gerekir ki BS-MSCR ve BS-MBCCR noktaları analitik olarak kapalı formda ifade edilebileceği gösterilebilir. Bu ifadeler aşağıdaki gibi verilebilir.

$$(\gamma_{\text{BS-MSCR}}, \alpha_{\text{BS-MSCR}}) = \left(\frac{F(d + \sum_{l=1}^{\rho_{\text{BS-MSCR}}} w_i b_i + t - 1)}{k(d + \sum_{l=1}^{\rho_{\text{BS-MSCR}}} b_i + t - k)}, \frac{F}{k} \right)$$

$$(\gamma_{\text{BS-MBCCR}}, \alpha_{\text{BS-MBCCR}}) = \left(\frac{F(2(d + \sum_{l=1}^{\rho_{\text{BS-MBCCR}}} w_i b_i) + t - 1)}{k(2(d + \sum_{l=1}^{\rho_{\text{BS-MBCCR}}} b_i) + t - k)}, \frac{F(2(d + \sum_{l=1}^{\rho_{\text{BS-MBCCR}}} b_i) + t - 1)}{k(2(d + \sum_{l=1}^{\rho_{\text{BS-MBCCR}}} b_i) + t - k)} \right)$$

burada ρ_(BS-MSCR) ve ρ_(BS-MSCR)BS-MSCR ve BS-MBCCR noktaları için en iyi bant aralığı maliyetini veren baz istasyonu sayılarıdır. Bu sayıların nasıl bulunacağı bir sonraki bölümde gösterilmiştir.



Şekil 4: Depolama ve onarım bant aralığı maliyeti ödünleşim uzayı. Bu uzay için k = 6, d = 9, t = 3, b = [1,0.75,0.5,0.25] ve w = [1.2,1.4,1.8,1.84] olarak alınmıştır. Normalize edilmiş metrikler, α ve γ_cdosya boyutu ile bölünmüş demektir.

3.5 Optimal Baz İstasyonu Algoritması, Alt-paketleme seviyesi ve Link Kapasiteleri

Optimal olan baz istasyonu sayısı **(Shum,2013)** çalışmasında bulunan analitik olarak ifade edilen ödünleşim uzayının üzerindeki bütün operasyonel noktalar ile ve bizim bulduğumuz analitik sonuçların karşılaştırılması ile bulunabilir. Buna örnek olarak önceki bölümde analitik olarak bulduğumuz BS-MSCR ve BS-MBCCR noktalarını gösterebiliriz. Ara noktalar için bu çalışma daha kompleks hale gelecektir. Öncelikle tek bir baz istasyonu durumunu düşünelim. İlk olarak, bir BS ile iletişim kurmanın ne zaman yararlı olacağını, yani sadece yerel işbirliği kullanılan duruma göre daha az maliyetli olacağını belirleyelim. Önceki çalışmalardan MSCR noktasını **(Shum,2013)** hatırlarsak, bu noktada t düğümünü onarmak BS ile iletişim kurarak daha az maliyet ancak aşağıdaki eşitsizlik sağlanırsa başarabileceğimizi fark edebiliriz. Bant aralığı maliyetini ρ'nun bir fonksiyonu olarak düşünürsek,

$$\gamma_c(\rho) \leq \frac{F}{k}\frac{d+t-1}{d+t-k}$$

Pratik bir durum olarak, hiyerarşide yükseldikçe her indirilen sembolün masrafının arttığı (w_i≤w_(i+1)) düşünülürse, aşağıdaki eşitsizliğin sağlandığı durumda i-inci BS kullanmanın yararlı olduğunu gösterebiliriz.

$$w_\rho \leq \frac{k\gamma_c(\rho-1)}{F}$$

Dolayısı ile, önce w_1≤(d+t-1)/(d+t-k)olup olmadığı kontrol edilir, eğer öyleyse bir sonraki kontrol yani w_2≤(d+t-1 + b_1 w_1)/(d+t-k+b_1) gerçekleştirilir. Sağlanmadığı index ρ ise, optimal baz istasyon sayısı yani ρ^*= ρ-1 olarak bulunur. Yani bu değer (ρ^*) aşağıdaki eşitsizliği sağlayan en küçük değerdir.

$$w_{\rho^*+1} > \frac{(d+t-1)+\sum_{j=1}^{\rho^*} w_j b_j}{(d+t-k)+\sum_{j=1}^{\rho^*} b_j}$$

Tabi burda BS-MSCR noktası örnek olarak düşünülmüştür. Aynı mantık ile BS-MBCCR noktası içinde en iyi baz istasyon sayısı bulunabilir. Bu iki noktayı aynı anda düşünerekten bir p_t ϵ{0,1} değişkeni kullanarak (p_t=0 BS-MBCCR, p_t=1 BS-MSCR noktaları için) algoritma üretilmiştir. Bu algoritmanın detayını aşağıda **Algoritma 1**'de bulabilirsiniz.

Algoritma 1: Optimal baz istasyonu sayısı (ρ_{min})

function OptBSNumCal$(k, d, t, M, \mathbf{b}, \mathbf{w}, p_t)$
$\rho_{min} \leftarrow 0$
for $i = 1 : M$ **do**
  $\bar{d} \leftarrow d + \sum_{l=1}^{i-1} w_l b_l \quad , \quad \bar{b} \leftarrow d + \sum_{l=1}^{i-1} b_l$ ▷ $\mathbf{b} = \{b_l\}$
  $\bar{w}_t \leftarrow p_t \left(\frac{\bar{d}+t-1}{\bar{b}+t-k}\right) + (1-p_t)\left(\frac{2\bar{d}+t-1}{2\bar{b}+t-k}\right)$ ▷ $p_t \in \{0,1\}$
  if $w_i > \bar{w}_t$ **then** ▷ $\mathbf{w} = \{w_l\}$
    break;
  end if
  $\rho_{min} \leftarrow \rho_{min} + 1$
end for
return ρ_{min}

Son olarak ifade etmek isteriz ki bu algoritmanın doğru olabilmesi için 0≤r_l≤b_l eşitsizliğinde r_l=b_l durumunun bant aralığı maliyetini minimum yapması gerekmektedir.

Bunların yanında pratik hayatta kullanılan herhangi bir hiyerarşik yapıda iletişim kurulan linklerin kapasiteleri bulunabilir. Bu bağlamda bu tip bir hiyerarşik mimari yapı içerisinde her katmanda bulunan baz istasyonlarının toplam etkisinin tek bir sanal baz istasyonu düşünülerek bulunabileceği öngörülebilir. Bu durumda, belirli bir katman için indirilen sembollerin masrafı bir tek "sanal" baz istasyonunun sembol indirme masrafına eşit olmakta, ve baz istasyonları için verili olan link kapasiteleri ise tekli sanal baz istasyonu için bu kapasitelerin toplamına eşit bir link kapasitesi düşünmenin istenilen alt-üst sınır ve doğru optimal sonuçları bulmak için yeterli olacağı gösterilebilir.

Bu bağlamda, i-inci baz istasyonu ile olan iletişim kapasitesi b_l=h_l/r_l, h_l,r_l∈{0,1,2,...}şeklinde ifade edelim. Dikkat edilmelidir ki herhangi bir reel b_l'ye uygun tam sayı h_l,r_l seçilerek yeteri kadar yakınsanabilir. Eger ρ adet BS kullanılırsa, sanal baz istasyonu bu durumda

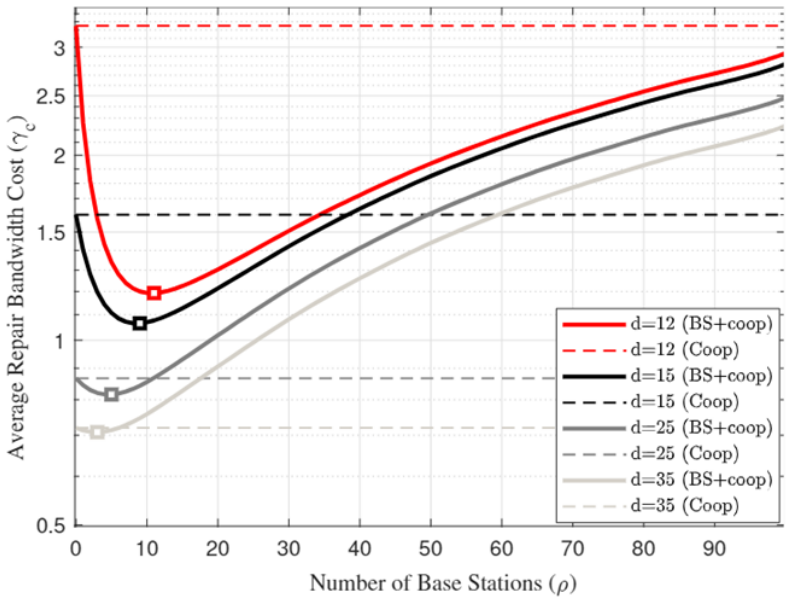
$$\frac{h_\rho}{r_\rho} := \sum_{l=1}^{\rho} \frac{h_l}{r_l}$$

toplanmış link kapasitesine sahip olacaktır. Daha önceki raporlama döneminde belirtildiği üzere i-inci katmanda bulunan BS'den veri çekme maliyeti (ağırlıkları) w_i olan bir sistemde optimal ρ(diğer bir gösterişle ρ^*) aşağıdaki eşitsizliği sağlayan minimum ρ olarak tanımlanmıştır.

$$w_{\rho^*+1} > \frac{d + t - 1 + \sum_{j=1}^{\rho^*} \frac{w_j h_j}{r_j}}{d + t - k + \frac{h_{\rho^*}}{r_{\rho^*}}}$$

Bu rapor döneminde optimal ρ seçimini görselleştirme açısından nümerik bir örnek vermek istiyoruz. Aşağıda görülen grafikte ağırlıkların Gauss dağılımına (ortalaması 0, standart sapması 10 olan ~ N(0,10)) göre dağıtıldığını varsayarak, ortalama onarım maliyetini baz istasyonu sayısının bir fonksiyonu olarak göstermekteyiz. Ayrıca, U(a,b)'nin tek tip (uniform) dağılımı temsil ettiği ve {a,a+1,...,b}'nin her bir öğesinin 1/(b-a+1) olasılık ile seçildiği düşünerek h_i~U(2,7) ve r_i~U(6,13) varsaymış durumdayız. Son olarak hiyerarşide yükseldikçe uzaklığın artışına bağlı olarak kapasitenin düştüğünü varsaymaktayız, diğer bir değişle h_i/r_i ≥ h_(i+1)/r_(i+1) ifadesi sağlanmaktadır. Hem BS destekli hem de saf kooperatif metotları için ortalama onarım maliyetini baz istasyonu sayısının bir fonksiyonu olarak gösterilmiştir. Optimum baz istasyonu sayısı sırası ile 11(d= 12),9(d= 15) ve 5(d= 25) olduğu görülmektedir. Bu gösterimde ortalama bant maliyeti aşağıdaki gibi verilebilir.

$$\gamma_c(\rho) = \frac{F}{k} \left[\frac{d+t-1+\sum_{i=1}^{\rho} \frac{w_i h_i}{r_i}}{d+t-k+\bar{h}_\rho / \bar{r}_\rho} \right]$$



Şekil 5: Baz istasyonu sayısı ile ortalama bant aralığı masrafı arasındaki ilişki.

Görüldüğü gibi yerel düğüm sayısı arttıkça baz istasyonlarına olan ihtiyaç azalmaktadır. Özellikle d büyük olsa da lokal düğümlerin güvenilir olmadığı durumlarda baz istasyonunu kullanımının (d'nin k'ya yakın olduğu durumlarda) büyük oranda tercih edilebilir olduğu görülmektedir.

Son olarak link kapasitelerinin h_l, r_l ile ifadeli şekli ile MSCR noktasında d=k durumu için kod tasarımları yapılmış ve alt paketleme seviyesi x∈{0,1} olmak üzere (d-k+t)r^x+xh bulunmuştur. Bu durum belirlendikten sonra daha önceki raporlama dönemindeki kod tasarımı genişletilmiş ve benzer şekilde yukarıdaki senaryoya uygun hale getirilmiştir.

3.6 Kod Tasarımı

Tam (exact) onarımın, önceki çalışmalarda belirtildiği gibi, işlevsel onarıma kıyasla azaltılmış bakım maliyeti gibi birçok avantajı vardır **(Shum ve Hu, 2013)**. Belirli bir rastgele kod parametreleri için kesin MSR ve MSCR kod yapıları oluşturulmuştur **(Ye ve Barg, 2017)**. Bu bölümde, ρ ve d = k ≤ n − t parametreleriyle BS destekli ortak onarım için bir yenilenebilir kod ailesi sunuyoruz. Basitlik açısından, b_l=1 olduğunu da varsayıyoruz. Önceki yapılara benzer şekilde, kod yapımızı, asal p ve pozitif tam sayı q için n ≤ p^q ile GF(p^q) 'den sembollerden oluşan, k boyutuyla n kod blok uzunluğu olan MDS kodlarına dayandırıyoruz.

Önerdiğimiz kod yapımız, aynı k × n jeneratör matrisine **G**=[g_1,g_2,...,g_n] sahip birden çok MDS kodu tabanlıdır. Burada g_i, **G** matrisinin i'inci kolonunu temsil etmektedir. Boyutu F olan bir dosya öncelikle k(t+ρ) bloklara bölünür. Buradaki ρ optimal baz istasyon sayısını

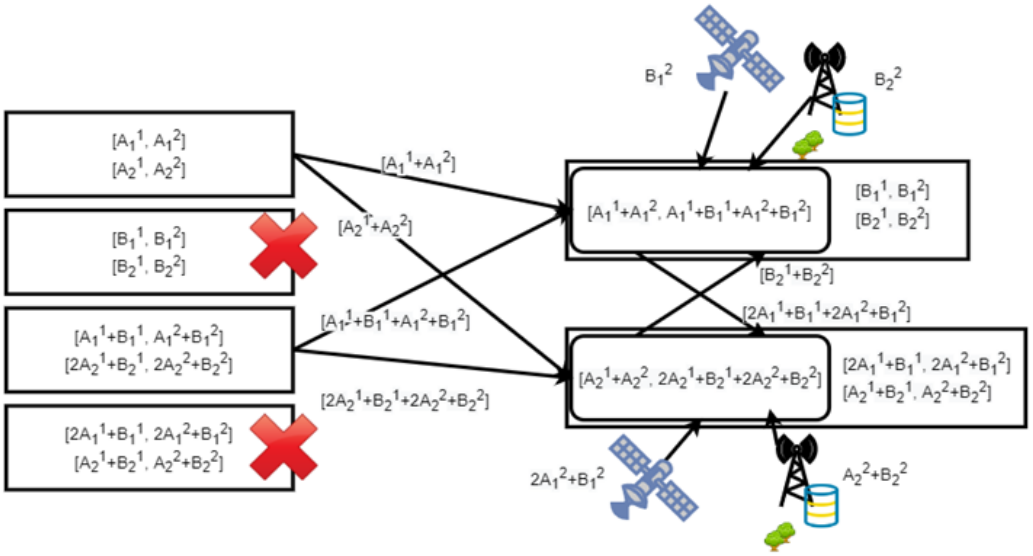
vermektedir. Her bloktaki semboller GF (p^q)'den seçilir. Bu bloklar (t+ρ)k boyutunda bir veri matrisi **M** şeklinde yapılandırılmaktadır. Böylece, **MG** hesaplanıp, j=1,2,...,n için j-inci kolonu j-inci düğümde saklanmak için dağıtılır. Farzedelim ki **M** matrisinin her sırasını m_1^T,m_2^T,...,m_(t+ρ)^T şeklinde ifade ediyoruz. Her bir sıra içinde k kadar veri sembolü bulunur, **m**_1^T=[m_1,...,m_k], m_i ϵ GF (p^q) ve bu vektör **m**_i^T**G** kod kelimesine dönüştürülür. Bu yapıda j-inci düğüm **m**_i^T **g**_j bütün i=1,2,...,t+ρ için verilidir.

Verili bir Δ zaman aralığı sonunda, farz edelim ki j_1,j_2,....,j_t indeksli düğümler hücreyi terk etmiş durumda olsunlar. Yenilemenin ilk aşamasında, *l*-inci yeni gelen j_l, kalan herhangi bir d = k düğümü bağlanır, diyelimki bunlar π_l (1),π_l (2),..,π_l (k) şeklinde gösterilsinler, ve **m**_i^T g_(π_l (1)),**m**_i^T g_(π_l (2)),...,**m**_i^T g_(π_l (k)) sembollerini indirerek **m**_l^T oluşturur. Sonraki onarım aşamasında *l*-inci yeni gelen j_l, önce **m**_l^T g_(j_h) hesaplar ve bunu h≠l olmayan bütün j_h yeni gelene iletir. Böylece t-1 kadar veri bloğu her bir yeni helen tarafından diğerlerinden elde edilir. Bu işbirlikçi ikinci aşamadan sonra *l*-inci yeni gelen düğüm j_l, t tane, açıkça yazacak olursak bütün h=1,2,..,t için **m**_h^T g_(j_l) veri bloklarını elde etmiş olur. Onarımın en son aşamasında kalan ρ veri bloğunu, her biri bir blok bilgi sağlayan mevcut ρ baz istasyonlarından indiriyoruz. Burada belirtmek gerekir ki, yenilenebilirliğin son iki aşamasının birbirinin yerine geçebilir durumdadır. Son olarak, her bir veri bloğunun F/(k(t+ρ)) büyüklüğünde olduğunu ve her bir yeni gelen düğümün toplamda d+ρ+t-1 veri bloğu indirerek minimum bant aralığı masrafını sağlamış olur.

Bu durumu örneklendirmek açısından ρ=2 için bir kodlama örneği sunmak isteriz. Bu örnekte n=4, k=2 ve depolanan dosyamızın A ve B diye iki farklı dosyadan F = 4MB olarak tutulduğunu varsayıyoruz. Bu durumda her düğüm 4 adet 0.5MB (toplam 2MB) lık paket tutmaktadır. Burada paket sayısı en düşük sayıda tutulursa kontrolü ve protokol tasarımı kolaylaşacaktır. Fakat istenen operasyonel noktaları başarabilmek içinde paket sayısı artırılması gerekecektir. Dolayısı ile bu ikisi arasında da bir ödünleşim bulunmaktadır. Bu ödünleşim üzerine de çalışmalarımız devam etmektedir.

Bu örnek senaryomuzda baz istasyonu lokal düğümler arasında iletişimi sağlanan β sembol sayısının r katı kadar onarıma fayda sağlayabilmektedir. Diğer taraftan uydu katmanı bütün veriye ait içerik depolanmaktadır. Aşağıda gösterilen resimde örnek kodlama sistemimiz görülebilir. Burada A dosyası A_1^1,A_2^1,A_1^2,A_2^2 *0.5 MB* lık paketler şeklinde, B dosyası B_1^1,B_2^1,B_1^2,B_2^2 *0.5 MB* lık paketler şeklinde iki farklı düğümde tutulmaktadır. Diğer iki düğüm ise bu paketlerin linear kombinasyonları tutmaktadırlar. Dikkat edilecek olursa herhangi iki düğümün içeriği kullanılarak hem A hem de B paketleri

oluşturulabilir. Diyelim ki ikinci ve dördüncü düğümlerimiz bozulsunlar ve bunları onarmak için en az bant aralığı masrafı kullanalım.



Şekil 6: Örnek bir onarım senaryosu. Burada β=β'=β''=0.5, r=1.

Öncelikle yeni gelen birinci düğüm önce A_1^1+A_2^1 ve A_1^1+B_1^1+A_1^2+B_1^2 olmak üzere iki adet *0.5 MB*'lık paketler indirmektedir. Bu iki paketin farkına bakarak B_1^1+B_1^2 oluşturulur. İkinci işbirlikçi fazda diğer onarım gören düğümden B_2^1+B_2^2 *0.5MB* lık paketi indirmektedir. B dosyasını oluşturabilmek için son olarak hem uydudan hem de baz istasyonunda 0.5 MB lık B_1^2 ve B_2^2 paketlerini sırası ile indirir. Dolayısı ile bu senaryoda β=0.5, β'=0.5 ve r=1 olmuş olur. Diğer onarım gören düğüm de benzer şekilde gösterildiği (Şekil 6) gibi paketler indirip onarımını tamamlamaktadır. Bu örnekte bu kodlama yönteminin optimal olacağı w_1=1.1 ve w_2=1.7 ağırlıkları seçilmiştir. Bu senaryo için bizim sistemimiz toplamda 2.9 bant aralığı maliyeti ile onarım sağlarken, bu maliyet işbirliksiz lokal onarım için 4, işbirlikli lokal onarım için 3 değerini alacaktır. Bu örnekte görüleceği üzere bant aralık masrafını düşürmenin yanında BS-MSCR noktasında çalışan basit bir kod tasarımı da yapmış bulunmaktayız. Bu örnek önceki tasarımımızın özel bir örneği olduğu görülebilir.

Son olarak ifade edilmelidir ki mesela ağırlıkları değiştirirsek, diğer sistemler aynı kalırken bizim sistemimiz bu ağırlıklara göre adapte olacak ve daha farklı parametreler seçerek (optimizasyon sonucu olarak) minimum bant aralığı masrafı sağlayacaktır. Mesela w_1=1.3 ve w_2=2.1olduğu durumda, bant aralığı masrafını 2.867'e düşürebiliriz. Fakat bu durumda Dosyalarımızı (A ve B) daha küçük parçalara bölmek gerekecektir mesela bu durumda üçe bölmek gerekecektir çünkü β=β'=β''=2/3 çıkmaktadır.

3.7 Protokol Tasarımı için Tahmin teorisi ve Tahmin etme maliyeti

Çalışmalarımız sırasında lokal düğümlerin bulunduğu aynı hücre içerisinde onarım gerçekleştiğinde onarımı yapılan düğümün halihazırdaki ağda bulunan düğümler hakkında anlık bilgi sahibi olamayacağı (hızlı mobil bir ortamda bu tür durumlar ile sık sık karşılaşabileceği gibi) gözümüze çarpmıştır. Farkına varıldığı üzere, önceki bölümlerde sunulan çalışmalarımızda onarım gören düğümlerin lokal düğümler hakkında tam bilgiye sahip olduğu varsayılmıştır. Bu bölümde bu varsayımın doğru olmadığı durumlar düşünülmüştür ki bu durum protokol tasarımını doğrudan etkileme durumu vardır. Bu senaryoda içinde bulunulan hücrede hangi düğümün aktif, hangisinin kayıp (dolayısı ile pasif) olduğu bilgisi eksik kaldığı durumda, onarımı yapılan düğümün yardım alacağı düğümleri (daha önceden paylaşılan ağ adres bilgileri aracılığı ile) deneyerek bulması gerekecektir. Çizge kodları bağlamında bu işlemin (protokolün) optimize edilebileceği ortaya çıkmıştır. Bu problemi bilgi teorisinde sıkça araştırılan “guessing” (tahmin etme) konusu **(Massey, 1994) (Arıkan, 1996)** ile ilgili olduğunu görüp temel bir çalışma gerçekleştirme durumu oluşmuştur **(Arslan ve Haytaoglu, 2020)**. Bu çalışmamızın üretmek istediğimiz optimal kod tasarımı ve protokollere yardımcı olmuştur.

Bu projede, tahmin maliyeti kavramını tanıtıp, sonlu bir küme üzerinde değerler alan rastgele bir değişkeni tahmin etmek için optimal bir strateji sunduk. Böylece her seçim pozitif bir sonlu maliyet değeriyle ilişkilendirilmiştir. Dahası, tahmin probleminin maliyeti için asimptotik olarak sıkı üst ve alt sınırlar üretilmiştir. Standart tahmin üzerine yapılan önceki çalışmalara benzer şekilde, momentler üzerine kurulan sınırlar, bilinmeyen seçimi doğru bir şekilde tanımlamak için gereken birikmiş tahmin maliyetini nicelleştirir ve Rényi'nin entropisi cinsinden ifade edilebilir. Tahmin maliyeti ile standart tahmin arasında köprü kurmak ve optimum tahmin anları üzerinde tahmin maliyeti üssü oluşturmak için yeni bir rasgele değişken tasarlanmıştır. Ayrıca, bu sınırların, seyrek grafik kodlarının kullanılabileceği dağıtılmış veri depolaması için onarım gecikme maliyetini bulmak için oldukça yararlı olduğu gözlemlenmiştir.

Veri korumasının bir (n, k) LDPC kodu **(Gallager, 1962)** ile sağlandığı, her düğümün tek bir sembolü depoladığı bir dağıtık veri depolama senaryosu için bir ana-yardımcı (master-slave) birim konfigürasyonu düşünelim. Ek olarak, bu ana düğüm yedekleme sistemi oluşturup ve tüm sembollerin kopyasını tutsun. Bağımlı düğüm arızası durumunda, birden fazla onarım seçeneği olacaktır. Anlık güvenilirliği sürdürebilmek için, tüm arıza bilgilerini aynı ağ içinde hızlı bir şekilde almak mümkün olmayabilir (diğer arızalar veya ağ bağlantısı kesintilerinden dolayı) veya bu bilgi için ana birimle iletişim kurmak çok zaman ve bant genişliği maliyetli olabilir. Bu nedenle, arızalı olan düğümün yerine geçecek olan düğüm en iyi tahmin stratejisini

uyarlaması ve onarım sürecini olabildiğince çabuk tamamlamak için çoklu onarım seçenekleri arasından seçim yapması gerekecektir.

3.8 Çizge Kodları Tabanlı Protokol ve Algoritma tasarımları

3.8.1 Çizge tabanlı kodlarda tekli ve çoklu düğüm onarımı için algoritma tasarlanması

Bu paket kapsamında hem algoritma geliştirilmesi gerçekleştirilmiş hem de yeni bir LDPC kod tasarımı üzerinde çalışılmıştır. Bu iki alt başlık aşağıda sırasıyla verilmiştir.

3.8.1.1 Algoritma Tasarımı

LDPC **(Eleftheriou ve Olcer, 2002)** kodlarında tek bir sembolün onarım süreci, düğümlerin birden fazla sembolü önbelleklerinde tuttuğu ve kayıp sembollerin bazılarının baz istasyonundan indirilebildiği durumda klasik sembol tamir etme sürecine göre farklılık gösterebilmektedir. Örnek olarak (8,4) bir LDPC kodunu düşünelim. Bu örnekte ham sembollerin kodlanması ile genişletilen yeni kodlanmış sembollerin 4 depolama düğümü üzerinde dağıtıldığını varsayalım. Diğer bir deyişle, her iterasyonda alınan 4 baytlık sembol grubu kodlanarak 8 bayta genişletilmekte ve ikişer ikişer sırasıyla bu 4 depolama düğümüne dağıtılmaktadır. Bu senaryoda bütün kodlanmış sembolleri saklayan tek bir baz istasyonun bulunduğunu da varsayalım. Aşağıda da bu örnek için kullanılan eşlik kontrol matrisi görülmektedir.

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Bu senaryoda birinci ve ikinci düğümlerin kaybolduğunu ve her i. düğümün $\{s_{2(i-1)+1}, s_{2(i-1)+2}\}$ aralığına düşen sembolleri önbelleğinde sakladığını düşünelim. Bu durumda birinci düğümde $s_1$ve $s_2$sembolleri, ikinci düğümde de $s_3$ve $s_4$sembolleri tutulmaktadır. Birinci düğümü onarmak için, bu düğümde tutulan kayıp sembollerin $\{s_1, s_2\}$ onarılması gerekmektedir. Bu kayıp sembollerin onarılmasında kullanılacak kurtarma denklemleri de yukarıda verilen H eşlik kontrol matrisine (parity check matrix) göre oluşturulmaktadır. Örneğin $s_1$sembolünün onarımı için kullanılabilecek alternatif kurtarma denklemleri verilen H matrisine göre $r_{1,1} = \{s_3 + s_7\}$ ve $r_{1,2} = \{s_2 + s_4\}$ olmaktadır. Burada $r_{i,j}$ notasyonu $s_i$sembolünün onarımı için kullanılabilecek j'inci kurtarma denklemini temsil etmektedir. Baz istasyonundan indirilecek sembol sayısı, $r_{1,1}$ kurtarma denklemi ve $r_{1,2}$

kurtarma denklemi kullanılır ise 2 olmaktadır. Aynı şekilde s_2 sembolünün onarımı için olası kurtarma denklemleri de $r_{2,1} = \{s_1 + s_4\}$ ve $r_{2,2} = \{s_4 + s_7 + s_8\}$'dir. Kayıp semboller $\{s_1, s_2, s_3, s_4\}$ düşünüldüğü zaman, baz istasyonu kullanımı da $r_{2,1}$ ve $r_{2,2}$ için sırasıyla 2 ve 1 olmaktadır. Onarım sürecinde $r_{1,1}$ ve $r_{2,2}$ denklemleri kullanılır ise s_3 ve s_4 sembolleri baz istasyonundan indirilmektedir. Eğer $r_{1,2}$ ve $r_{2,2}$ denklemleri kullanılır ise sadece s_4 sembolü baz istasyonundan indirilmektedir. Bu seçimde s_2 sembolü ilk olarak onarılır ise, bu sembol $r_{1,2}$denklemi üzerinden s_1 sembolünün onarımında kullanılabilir. Bu örnekte de görüldüğü gibi kurtarma denklemlerinin kullanım sırası baz istasyonundan indirilecek sembol sayısını etkilemektedir.

Minimum baz istasyonu kullanımı hedeflenerek kayıp düğümlerin onarımı, NP-Complete bir problemi temsil etmektedir. Algoritma tasarımımız da yukarıdaki örnekteki gibi senaryolarda baz istasyonu kullanımını minimize etmeyi amaçlamaktadır.

Greepair Algoritması

Kayıp düğümlerin tamiri için geliştirmiş olduğumuz Greepair algoritması, baz istasyonu kullanımını minimize etmeyi amaçlayan iki aşamalı (Algoritma 1 ve Algoritma 2) yeni bir açgözlü (greedy) algoritmadır. Bu aşamalara ait algoritmalarda kullanılan notasyona değinilecek olursa, L ilgili kayıp düğümdeki sembol indekslerini, G de kayıp düğümlerin tamamında önceden depolanmış olan bütün kayıp sembol indekslerini tutmaktadır. Greepair algoritmasının çalışma mantığını genel olarak özetlemek gerekirse, kayıp bir düğümün onarımı sürecinde bu düğüme ait her kayıp sembol onarıldıktan sonra L'den ilgili sembol indeksi çıkarılır ve L içerisinde hiç sembol indeksi kalmayana kadar bu işlem yinelemeli olarak tekrar edilir. Baz istasyonu kullanımı olmadan onarımı gerçekleştirilebilecek olan kayıp sembollere ait indeksler L_1 kümesinde tutulmaktadır ve baz istasyonsuz bu onarımlar Greepair algoritmasının ilk aşamasında (Algoritma 1) gerçekleştirilmektedir. Birinci fazda baz istasyonsuz gerçekleştirilebilecek bütün sembol onarımları tamamlandıktan sonra, onarım sürecinde baz istasyonu iletişimine ihtiyaç duyulan diğer kayıp semboller de ikinci fazda (Algoritma 2) onarılmaktadır.

Birinci Faz (Algoritma 1)

Bu aşamada baz istasyonsuz maksimum sayıda sembol onarımı ve bu onarımlar süresince diğer düğümlerden minimum sayıda yardımcı sembolün (helper symbol) indirilmesi amaçlanmaktadır. Bu noktada belirtmek gerekir ki yardımcı semboller bu kurtarma

denklemleri içerisindeki sembolleri ifade etmektedir. Bu aşamaya ait algoritma, L_1 kümesinde indeksleri tutulan kayıp semboller için baz istasyonu iletişimi gerektirmeyen kurtarma denklemlerinin eşlik denetimi matrisi kullanılarak tanımlanması ile başlar. L_1 kümesindeki kayıp sembollere ait kurtarma denklemleri R_1 dizisi içerisinde tutulmaktadır. Her sembol onarımının ardından R_1 dizisi güncellenmektedir. Bu sembol onarımlarından önce ise orijinal kurtarma denklemlerini tutmak için R_1, $R_2^{''}$ dizisine kopyalanır. Bu aşamada seçilecek kurtarma denklemlerinde aranan temel özellik ise minimum kardinalitedir (minimum cardinality). Minimum kardinalite ile kastedilen ise kurtarma denklemlerinde yardımcı sembol sayısının minimum olmasıdır.

L_1 kümesindeki her bir kayıp sembolün onarımından sonra R_1 denklem kümesi güncellenirken, Greepair Algoritmasının ikinci aşamasında kullanılacak olan ve baz istasyonu kullanımına ihtiyaç duyan kurtarma denklemlerinin tutulduğu R_2 kümesi de oluşturulur. R_2 kümesinde $L \backslash L_1$içindeki kayıp sembollerden en az bir adet bulunmaktadır. R_2 kümesinin bir kopyası tıpkı R_1ve $R_1^{''}$arasındaki ilişkiye benzer şekilde $R_2^{''}$ içinde tutulur. Sürecin devamında R_1 ve R_2 kümesindeki kurtarma denklemleri, kardinalitelerine göre artan bir sırada dizilir. Greepair Algoritmasının her iki aşamasında da $F(X)$ olarak adlandırılan bir fonksiyondan faydalanılır. Bu fonksiyonun etki alanı (domain) kurtarma denklemleri kümesi iken, değer kümesi (codomain) ise kayıp semboller kümesidir. Bu fonksiyonun görevi kayıp bir sembol ile bu kayıp sembolün onarımında kullanılabilecek olan kurtarma denklemini eşlemektir. Örneğin, s_i kayıp sembolü onarıldıktan sonra ilgili indeksi L ve L_1 kümelerinden çıkarılır. $F(r) = s_i$ koşulunu sağlayan r kurtarma denklemine artık ihtiyaç kalmadığı için bu r denklemi R_1 ve $R_1^{''}$'den çıkarılır. Bu süreç Algoritma 1'de 9 ve 10. satırlarda görülebilir. Onarılacak olan bir sonraki kayıp sembolün seçiminden önce, onarılan sembollerin yanı sıra önceki onarımlarda kullanılan yardımcı semboller de R_1 ve R_2 kümelerinden çıkarılır. Bu işlemler Algoritma 1'de 11. ve 12. satırlarda görülebilir. Onarımı gerçekleştirilen sembolün R_2 kümesindeki kurtarma denklemlerinden baz istasyonu kullanımını ortadan kaldırıp kaldırmadığı kontrol edilir. Eğer onarılan sembol sayesinde R_2 kümesinde artık baz istasyonu iletişimine gereksinim duymayan bir r_i denklemi bulunursa, bu denklem R_2 ve $R_2^{''}$

kümelerinden çıkarılıp R_1 ve $R_1^{''}$ kümelerine eklenir. Ayrıca $F(r_i)$ L_1 kümesine eklenir. İlk aşamadaki tüm bu işlemler R_i kümesinde hiç bir denklem kalmayana kadar tekrarlanır.

Algoritma 2: Greepair Algoritması (İlk Faz)

input $\mathcal{G}$, $\mathcal{L}$, $\mathcal{L}_1$ ▷ $\mathcal{G}$ is the set of the lost symbols in other lost nodes
output $\mathcal{R}_2, \mathcal{R}_2''$
function FirstPhase($\mathcal{G}$, $\mathcal{L}$, $\mathcal{L}_1$)
construct $\mathcal{R}_1$ and $\mathcal{R}_2$
sort $\mathcal{R}_1$ and $\mathcal{R}_2$ according to recovery equation set sizes in ascending order
$\mathcal{R}_1'' \leftarrow \mathcal{R}_1$, $\mathcal{R}_2'' \leftarrow \mathcal{R}_2$
while $\mathcal{R}_1 \neq \emptyset \wedge \mathcal{L}_1 \neq \emptyset$ **do**
  $R \leftarrow \mathcal{R}_1''[0]$
  repair symbol $\mathcal{F}(R)$ using R
  $\mathcal{L} \leftarrow \mathcal{L} \setminus \mathcal{F}(R)$
  $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \setminus \mathcal{F}(R)$
  $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \setminus r, \forall\, r \in \mathcal{R}_1 \wedge \mathcal{F}(R) = \mathcal{F}(r) \;\wedge\; r \neq \mathcal{R}_1[0]$
  $\mathcal{R}_1'' \leftarrow \mathcal{R}_1'' \setminus r, \forall\, r \in \mathcal{R}_1'' \wedge \mathcal{F}(R) = \mathcal{F}(r)$
  $r \leftarrow r \setminus (\mathcal{F}(R) \cup R), \forall\, r \in \mathcal{R}_1$
  $r \leftarrow r \setminus (\mathcal{F}(R) \cup R), \forall\, r \in \mathcal{R}_2$
  while $\exists\, r \mid r \cap ((\mathcal{L} \setminus \mathcal{L}_1) \cup \mathcal{G}) = \emptyset \wedge r \in \mathcal{R}_2$ **do**
    $idx \leftarrow i \mid \mathcal{R}_2[i] = r$
    $\mathcal{R}_2 \leftarrow \mathcal{R}_2 \setminus r$
    $\mathcal{R}_1'' \leftarrow \mathcal{R}_1'' \cup \mathcal{R}_2''[idx]$
    $\mathcal{R}_2'' \leftarrow \mathcal{R}_2'' \setminus \mathcal{R}_2''[idx]$
    $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \cup r$, $\mathcal{L}_1 \leftarrow \mathcal{L}_1 \cup \mathcal{F}(r)$
    $\mathcal{R}_2 \leftarrow \mathcal{R}_2 \setminus r_i, \forall\, r_i \in \mathcal{R}_2 \wedge \mathcal{F}(r_i) = \mathcal{F}(r)$
    $\mathcal{R}_2'' \leftarrow \mathcal{R}_2'' \setminus r_i, \forall\, r_i \in \mathcal{R}_2'' \wedge \mathcal{F}(r_i) = \mathcal{F}(r)$
  end while
  $\mathcal{R}_1'' \leftarrow \mathcal{R}_1'' \setminus R$
  $\mathcal{R}_1 \leftarrow \mathcal{R}_1 \setminus \mathcal{R}_1[0]$
  Update orders of $\mathcal{R}_1$ and $\mathcal{R}_2$ with respect to the cardinalities of recovery equations
  Reorder $\mathcal{R}_1''$ and $\mathcal{R}_2''$ according to $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively.
end while
SecondPhase($\mathcal{G}$, $\mathcal{L}$, $\mathcal{R}_2$, $\mathcal{R}_2''$)

İkinci Faz (Algoritma 2)

Birinci aşama tamamlandıktan sonra geriye R_2 kümesinde baz istasyonu iletişimine ihtiyaç duyan kurtarma denklemleri kalır. L kümesinde de bu kurtarma denklemleri ile eşleşen kayıp sembol indeksleri bulunur. İlk olarak R_2 kümesinden en az baz istasyonu iletişimine sahip kurtarma denklemi seçilir. Eğer minimum baz istasyonu kullanımına sahip birden fazla kurtarma denklemi var ise, bu denklemlerden R_2 kümesinde en fazla yardımcı sembol olarak kullanılan kayıp sembole ait olanı seçilir. Seçilen bir kurtarma denkleminde en azından bir yardımcı sembolün dahi baz istasyonundan indirilmesi gerekiyor ise, kayıp sembol bu denklem kullanılmaksızın baz istasyonundan doğrudan indirilir. Her kayıp sembol onarımının

ardından R_2 ve $R_2^{''}$ kümeleri güncellenir. Bu noktada belirtmek gerekir ki kayıp bir sembol onarıldıktan sonra bir sonraki kayıp sembol onarımında yardımcı sembol olarak kullanılabilir. Diğer bir deyişle, kayıp bir sembol onarımının ardından başka bir kayıp sembol için kullanılabilecek kurtarma denklemleri içerisinde daha önceden onarılan bu kayıp sembol bulunabilir. İlgili kurtarma denklemlerinde baz istasyonu iletişimi gerektiren tek sembol de daha önceden onarılan bu sembol ise, bu kurtarma denklemleri artık baz istasyonundan veri indirmeye ihtiyaç duymayacaktır. Kısaca Greepair Algoritmasının bu ikinci aşamasında her iterasyonun ardından onarımlarda kullanılabilecek olası kurtarma denklemlerinin baz istasyon kullanımı daha da düşürülebilmektedir .

Algoritma 3: Greepair Algoritması (İkinci Faz)

input $\mathcal{G}$, $\mathcal{L}$, $\mathcal{R}_2$, $\mathcal{R}_2''$
output $BSScore$
function SecondPhase($\mathcal{G}$,$\mathcal{L}$, $\mathcal{R}_2$,$\mathcal{R}_2''$)

while $\mathcal{L} \neq \emptyset$ **do**
  $R \leftarrow \mathcal{R}_2''[0]$
  $\mathcal{R}' \leftarrow \mathcal{R}_2[0]$
  if $|\mathcal{R}' \cap (\mathcal{L} \cup \mathcal{G})| \neq 0$ **then**
    download $F(\mathcal{R}')$ from BS
    $BSScore = BSScore + 1$
  else
    repair the lost symbol $F(\mathcal{R}')$ without BS
    $r \leftarrow r \setminus \mathcal{R}', \forall\ r \in \mathcal{R}_2$
  end if
  $\mathcal{L} \leftarrow (\mathcal{L} \setminus \mathcal{F}(\mathcal{R}))$
  $r \leftarrow r \setminus (\mathcal{F}(\mathcal{R})), \forall\ r \in \mathcal{R}_2$
  $\mathcal{R}_2 \leftarrow \mathcal{R}_2 \setminus r, \forall\ r \in \mathcal{R}_2 \wedge \mathcal{F}(\mathcal{R}) = \mathcal{F}(r)$
  $\mathcal{R}_2'' \leftarrow \mathcal{R}_2'' \setminus r, \forall\ r \in \mathcal{R}_2'' \wedge \mathcal{F}(\mathcal{R}) = \mathcal{F}(r)$
  Update the order of $\mathcal{R}_2$ with respect to their BS needs and correlation.
  reorder $\mathcal{R}_2''$ according to $\mathcal{R}_2$
end while

Aşağıdaki bölümde iki alt başlık olarak sırasıyla bant genişliği ve karmaşıklık analizi ile ilgili alt başlıklar verilmiştir.

3.8.1.2 Algoritmanın ve Rakip Kodların Bant-genişliği Analizi

Bu paket ile ilgili konulardan bant genişliği ile ilgili olan incelemeler önceki raporda da incelenmiş olup aşağıda bir özeti verilmiştir:

Bu durumlardan birincisinde, hücresel ağ içerisindeki çevrimiçi düğüm sayısı ve hangi verilerin tutulduğunun bilindiği varsayımı ile masraf fonksiyonu farklı kodlar için (RS, MSR ve MBR) kapalı formda hesaplanmış durumdadır.

Hücresel sistemimizde kullanılan (n,k,d) kodu ve üretilen n sembolün m farklı düğüme dağıtıldığı, ve ayrıca l tane düğümün anlık olarak hücreden ayrıldığı ya da çevrimdışı olduğu varsayıldığında, RS kodlar için bulunan koşullu masraf fonksiyonu

C_{RS} aşağıdaki gibi ifade edilebilir.

$$C_{\mathrm{RS}}(F,m,n,k,l) = \begin{cases} F\rho_{\mathrm{D2D}}, & \text{if } k \le n-l \\ \frac{F}{k}(k-n+l)\rho_{\mathrm{BS}} + \frac{F}{k}(n-l)\rho_{\mathrm{D2D}}, & \text{if } n-l < k \ldots \\ & \wedge\ k < \frac{n}{m}+n-l \\ \frac{n}{m}\frac{F\rho_{\mathrm{BS}}}{k}, & \text{if } k \ge \frac{n}{m}+n-l \end{cases}$$

Burada F, dosya boyutunu, ρ_{BS} ve ρ_{D2D}sırası ile baz istasyonu ve aygıtlar arası iletişim maliyetlerini göstermektedir. Benzer şekilde bu masraf fonksiyonu (C_{MBR}) aşağıdaki gibi bulunmuştur.

$$C_{\mathrm{MBR}}(F,m,n,k,d,B_1,l) = \begin{cases} \sum_{a=0}^{d-n+l-1}\left(\frac{F}{B_1}\rho_{\mathrm{D2D}}(n-l)\ldots\right. \\ \left.+\frac{F}{B_1}\rho_{\mathrm{BS}}(d-n+l-a)\right)\ldots \\ +\sum_{a=d-n+l}^{\frac{n}{m}-1}\frac{F}{B_1}\rho_{\mathrm{D2D}}(d-a), & \text{if } d > n-l \\ \sum_{a=0}^{\frac{n}{m}-1}\frac{F}{B_1}\rho_{\mathrm{D2D}}(d-a) & \text{if } d \le n-l\} \end{cases}$$

Burada $B_1 = kd - \frac{k(k-1)}{2}$ ve a kayıp düğüm onarımı sırasında onarım görmüş paketlerin sayısını göstermektedir. Bunların yanında MSR kodları için bulduğumuz maliyet fonksiyonu kod oranı ve oluşturulmasına doğrudan bağımlılık göstermektedir. Aşağıda sunulan masraf fonksiyonu $d \ge 2k-2$, $\beta = 1, \alpha = d-k+1$ ve $B_2 = k(d-k+1)$ parametreleri için verilmiştir.

$$C_{\mathrm{MSR_{LR}}}(.) = \begin{cases} \sum_{a=0}^{\frac{n}{m}-1}\frac{F}{B_2}\rho_{\mathrm{D2D}}(d-a), & \text{if } d \le n-l, \\ \left(\sum_{a=0}^{d-n+l-1}\left(\frac{F}{B_2}(\rho_{\mathrm{D2D}}(n-l)+\rho_{\mathrm{BS}}(d-n+l-a))\right)\mathbb{1}_A\ldots\right. \\ \left.+\left(\frac{F}{B_2}\rho_{\mathrm{BS}}(d-k+1)\right)(1-\mathbb{1}_A)\right)\ldots \\ +\left(\sum_{a=d-n+l}^{\frac{n}{m}-1}\frac{F}{B_2}\rho_{\mathrm{D2D}}(d-a)\right) & \text{otherwise} \end{cases}$$

Burada A durumu $d-n+l-a < d-k+1$durumunu belirtmekte ve $\mathbb{1}_A$ ise indikatör fonksiyonunu göstermektedir. dikkat edileceği üzere bu kodlama sisteminin kod oranı düşük

olduğundan ötürü LR kullanılmıştır. Diğer taraftan **(Wang v.d. ,2016)** çalışmasında sistematik ve yüksek oranlı kodlar önerilmiştir. Bu çalışmada ayrıca $\frac{n}{m} = 1$olarak kabul edilmiştir. Bu kodlar için masraf fonksiyonu aşağıdaki gibi kapalı formda bulunmuştur.

$$C_{\mathrm{MSR}_{\mathrm{HR_{sys}}}}(F,z,n,d,t,B_2,l) = \begin{cases} t^{z-1} d \frac{F}{B_2} \rho_{\mathrm{D2D}}, & \text{if } d \le n-l, \\ \frac{F}{B_2}(t^{z-1}(n-l)\rho_{\mathrm{D2D}} + (d-n+l)t^{z-1}\rho_{\mathrm{BS}}), & \text{if } d-n+l < t, \\ t^z \frac{F}{B_2}\rho_{\mathrm{BS}}, & \text{otherwise} \end{cases}$$

Bu ifadede n=(t+1)z+t, d=n-1, k=(t+1)z, $\alpha = t^z, B_2 = kt^z$olarak verilmiştir. RS, MBR ve MSR kodlarının yanında proje bünyesinde önerilen (d_c, d_v)düzenli (regular) LDPC kodları için masraf fonksiyonu stokastik doğası gereği ortalama masraf fonksiyonu cinsinden bulunmuştur. Fakat, LDPC kodlarının karmaşık ve olasılıksal doğası nedeni ile genel kapalı formda ifadeler yerine ortalama masraf fonksiyonu için bir üst sınır bulunmuştur. Bu üst sınırı $q(a)$ aşağıdaki gibi ifade edebiliriz.

$$\begin{aligned} \mathbb{E}[C_{\mathrm{LDPC}}(F,m,k,n,l)] \le & \sum_{a=0}^{\lceil \frac{n}{m} \rceil - 1} (d_c-1) d_v (1-q(a))^{d_c-1} \frac{F}{k} \rho_{\mathrm{D2D}} \cdots \\ & + (1-(1-q(a))^{d_c-1}) \frac{F}{k} \rho_{\mathrm{BS}} \qquad (8) \\ = & \sum_{a=0}^{\lceil \frac{n}{m} \rceil - 1} \frac{F}{k} \left(\rho_{\mathrm{BS}} + (1-q(a))^{d_c-1} \left((d_c-1) d_v \rho_{\mathrm{D2D}} - \rho_{\mathrm{BS}} \right) \right) \end{aligned}$$

Bu ifadeleri özel kılan aslında hücresel sistemde ayrılmalar sonucu kalan düğüm sayısının d ya da k dan az olduğu durumlarda kullanılan baz istasyonu temelli protokollerdir. Önceki çalışmalarda genel olarak bu durumlar tamamen baz istasyonu kullanılarak çözülmüş ve LDPC kodları gibi daha sofistike kod yapıları kullanılmamıştır. İfadelerimizin detaylı türetilmesi ve kodlar arasındaki karşılaştırmalar basılan makalemizde **(Haytaoglu v.d., 2022)** bulunabilir.

3.8.1.3 İşlem karmaşıklığı açısından incelemeler

TCOM makalesinde önerilmiş olan LDPC kodlarının kullanıldığı Greepair algoritmasının karmaşıklık analizi aşağıda verilmiştir:

Burada tüm kodlardaki düğüm tamiri karmaşıklıkları kıyaslanmıştır. Burada düğüm tamiri kullanılan kodun oluşturulmasına çok bağlı olduğu için, aynı tür kodların farklı gerçekleştirimleri daha farklı düğüm tamiri hesaplama karmaşıklığına neden olmaktadır. Düzenli (d_v,d_c) LDPC kodları; düğüm tamiri O(n/m) düğüm tamiri gerektirmektedir. Burada, O((d_v-1)(n/m)) XOR işlemi gerekmektedir. Ancak önerdiğimiz düğüm tamiri algoritmasında fazladan başka özel veri yapıları üzerinde gerçekleşen işlemler gerçekleştirmektedir. Bunlardan ilki kayıp bir sembol için oluşturulan tamir eşitliklerini tutan bir hashmap'tir. Diğer bir hashmap bu eşitliklerin terim sayısını tutmak için kullanılmaktadır. Dahası, üçüncü hashmap; bu eşitliklerden tamir edilen sembole doğru bir eşleştirme için kullanılmaktadır. Başlangıçta; BS kullanımı gerektirmeyen eşitlikler kullanılmaktadır.

Kayıp bir sembolü tamir etmek için içerisinde dc adet sembol içeren dv adet farklı eşitlik bulunmaktadır. Buna ek olarak, bu eşitliğin baz istasyonu gerektirip gerektirmediğini ayırt etmek gerekmektedir. Bu işlem en fazla O((nd_vd_c)/mmin(|[n]\L|,|L|)) kıyaslama gerektirmektedir. Bu ayırt etme işleminde bir heap yapısı kullanılmıştır. Daha tamir etme işlemi başlamadan bahsedilen heap'in büyüklüğü n/m'dir. Heapin oluşturulması O(n/mlog(n/m)) adım gerektirmektedir. Her bir sembol tamiri en fazla O(d_c-1) XOR işlemi gerektirmektedir. Her sembol tamirinden sonra; dvdc adet güncelleme işlemi gerçekleştirilmektedir. Ayrıca heap yapısı en fazla min(n/m, d_vd_c) defa güncelleme işleminden geçirilmektedir. Sonuç olarak tüm tamir işlemleri en fazla O((nd_vd_c)/min([n]\L, |L|)+d_vd_cn/m+min(d_vd_c,n/m)n/mlog(n/m)+(n/m)^2d_v) kıyaslama gerektirmektedir. Buna göre, sabit depolama düğümü sayısı için blok uzunluğunu arttırmak tamir karmaşıklığını arttırmaktadır. Bu tamir işlemleri dosya boyutuna bağlı olarak F/k adet tekrarlanmalı, ve dolayısıyla bir kayıp sembolün içeriğinin tamamen tamir edilmesi işlemi O(F/k((nd_vd_c)/min([n]\L, |L|)+d_vd_cn/m+min(d_vd_c,n/m)n/mlog(n/m)+(n/m)^2d_v)) kıyaslama ve O((n/m)d^3)) XOR gerektirmektedir.

Geleneksel (n,k) RS kodları, n/m sembol O(k^3+Fk+(n/m)k) sonlu alan toplaması ve çarpma işlemi gerektirmektedir. Böylece, bir düğümü tamir etme işlemi O(k^3+Fk+F(n/m)) işlem gerektirmektedir. MBR kodlarında **(Dimakis v.d., 2010), (Rashmi v.d., 2011)** ise, tamir edilen bir düğüm için her bir yardımcı düğüm O(F/B_1d(n/m)) sonlu alan çarpması ve toplaması işlemi gerçekleştirilmektedir. MSR kodlarında **(Dimakis v.d., 2010), (Rashmi v.d., 2011)** ise kayıp bir düğümü tamir etmek için MBR kodlarında olduğu gibi O((d^3+d^2F/B_2)n/m) sonlu alan toplaması ve çarpması işlemi gerçekleştirilmektedir. Böylece, küçük dv ve dc, min(|[n]\L|,L) ve n/m işlemi için Greepair algoritması düşük karmaşıklığı nedeni ile rakiplerinden daha üstün bir performans sağlamaktadır.

3.8.2. Artık veri uygulanması

Klasik bir LDPC'de kodlama (encoding) süreci şu şekilde gerçekleştirilir. k sembolden oluşan veri, üretici matris G üzerinden XOR işlemlerinden geçirilerek $(n - k)$ kadar genişletilir. Aşağıdaki formülizasyonda bu XOR işlemi gösterilmektedir. Henüz kodlanmamış veri yığını

$\bar{b} = [b_1, b_2, ..., b_k]$ içindeki j. sembol b_j ile temsil edilir iken kodlanmış veri vektörü $\bar{d} = [d_1, d_2, ..., d_n]$ içindeki j. kodlanmış sembol d_j ile temsil edilmektedir.

$$[b_1\ b_2 \dots b_k] \oplus \begin{bmatrix} g_{11} & \cdots & g_{1n} \\ g_{21} & \ddots & g_{2n} \\ \vdots & & \vdots \\ g_{k1} & \cdots & g_{kn} \end{bmatrix} = [d_1\ d_2 \dots d_n]$$

$$\bar{b} \oplus \mathrm{G} = \bar{d}$$

Bu kodlama süreci ardından elde edilen n adet kodlanmış sembol $(\overline{d})$, kayıp bir düğümün tamir süreci için hücre içerisinde hiçbir veri taşımayan boş düğümlerden rastgele olarak seçilmiş düğümlere belirlenen belli bir oranda kaydedilmeye başlanır. Standart sembol dağıtım yöntemi, İlk senaryoda 15 adet kodlanmış sembolün 4 adet düğüme depolanmak üzere dağıtılmak istendiğini varsayalım. Bu senaryoda $15/4 = 3.75$ değeri 4'e yuvarlanır ve her düğüme 4 adet sembol gelecek şekilde dağılım gerçekleştirilir. Bu senaryoya göre standart sembol dağıtım yöntemi aşağıdaki şekilde gösterilmektedir. Kodlanmış olan 15 sembol ilgili depolama düğümlerine dağıtıldıktan sonra ekstra olarak 16. sembol sıfır değeri ile son depolama düğümüne eklenir. Böylece fazladan eklenen veriler ile bütün depolama düğümlerinde eş sayıda sembolün tutulması sağlanır.

Sembol İndeksi	1. Döngü	2. Döngü	3. Döngü
1	1. düğüm	1. düğüm	1. düğüm
2	1. düğüm	1. düğüm	1. düğüm
3	1. düğüm	1. düğüm	1. düğüm
4	1. düğüm	1. düğüm	1. düğüm
5	2. düğüm	2. düğüm	2. düğüm
6	2. düğüm	2. düğüm	2. düğüm
7	2. düğüm	2. düğüm	2. düğüm
8	2. düğüm	2. düğüm	2. düğüm
9	3. düğüm	3. düğüm	3. düğüm
10	3. düğüm	3. düğüm	3. düğüm
11	3. düğüm	3. düğüm	3. düğüm
12	3. düğüm	3. düğüm	3. düğüm
13	4. düğüm	4. düğüm	4. düğüm
14	4. düğüm	4. düğüm	4. düğüm
15	4. düğüm	4. düğüm	4. düğüm
16	4. düğüm	4. düğüm	4. düğüm

Şekil 7: Standart sembol dağıtım yöntemi.

Fazladan kaydedilen bu sıfır “0” verisinin ortadan kaldırılması için önerilen artık veri kullanımı kapsamında, düğümlere bu kodlanmış sembollerin dağılımında iki farklı yöntem önerilmiştir. Birincisi artık verili sembol dağıtım yöntemine dair süreç bir önceki örnek üzerinden anlatılacak olur ise her düğüme kaydedilecek sembol sayısı $15/4 = 3.75$değerinin alt tam sayıya yuvarlanması ile 3 olarak belirlenir. Her düğüme sırasıyla 3 adet sembol kaydedilir ve ilk 12 kodlanmış sembol ilgili düğümlere aktarılmış olur. Bu noktadan itibaren ise artık veri aktarımı başlar. Bu örnek üzerinden devam edilecek olur ise 13. kodlanmış sembolden 15. kodlanmış sembole kadar olan ve her sembolünün farklı düğümlere kaydedildiği veri kısmı artık veri olarak tanımlanır. 13’ten 15’e kadar olan 3 sembol sırasıyla ilk 3 düğüme kaydedilir. Burada 4. düğüme herhangi bir artık veri aktarımı yapılmadığına dikkat edilmelidir. Böylece her 15’lik kodlanmış sembolde ilk 3 düğüme 4 sembol, 4. sembole ise 3 sembol kaydedilmiş olur.

İkinci artık verili sembol dağıtım yönteminde ise artık verinin tutulacağı sembol indeksleri hep aynı düğümde depolanacak şekilde dağıtılır. Yine aynı örnek üzerinden göstermek gerekirse, 15 adet kodlanmış sembolün ilk 12 sembolü üçerli gruplar halinde 4 depolama düğümüne

dağıtılır. 13, 14 ve 15. sembol indeksleri ise artık verilerin tutulacağı semboller olur. 13. sembol indeksinden başlayarak ilk depolama düğümünden sırasıyla her sembol bir düğüme gönderilir. Her iterasyonda da aynı sembol indeksine denk gelen semboller aynı depolama düğümünde saklanır. Şekil 6'da ikinci artık veri sembol dağıtım yöntemi örneği gösterilmektedir. Bu sembol dağıtım yöntemi ile de tıpkı ilk versiyonda olduğu gibi fazladan veri tutulmasının önüne geçilmektedir. İlk artık veri sembol dağıtım çalışmasından farklı olarak bu versiyonda depolama düğümleri üzerinde tutulan veri miktarı farklılık gösterebilmektedir.

Sembol İndeksi	1. Döngü	2. Döngü	3. Döngü
1	1. düğüm	1. düğüm	1. düğüm
2	1. düğüm	1. düğüm	1. düğüm
3	1. düğüm	1. düğüm	1. düğüm
4	2. düğüm	2. düğüm	2. düğüm
5	2. düğüm	2. düğüm	2. düğüm
6	2. düğüm	2. düğüm	2. düğüm
7	3. düğüm	3. düğüm	3. düğüm
8	3. düğüm	3. düğüm	3. düğüm
9	3. düğüm	3. düğüm	3. düğüm
10	4. düğüm	4. düğüm	4. düğüm
11	4. düğüm	4. düğüm	4. düğüm
12	4. düğüm	4. düğüm	4. düğüm
13	1. düğüm	1. düğüm	1. düğüm
14	2. düğüm	2. düğüm	2. düğüm
15	3. düğüm	3. düğüm	3. düğüm

Şekil 8: İkinci artık verili sembol dağıtım yöntemi.

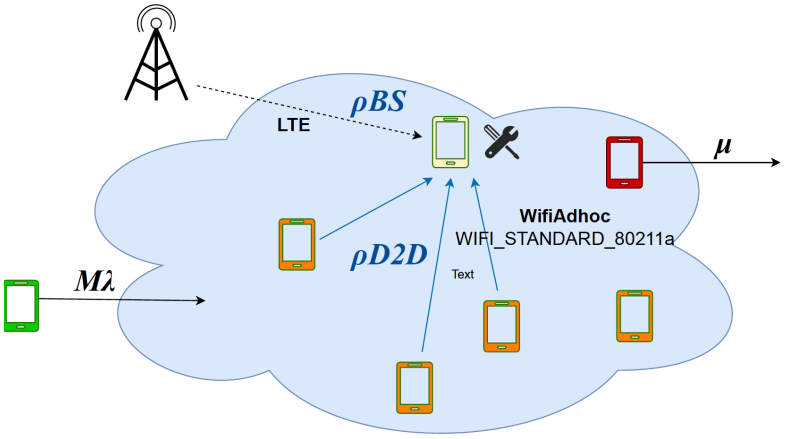
Bu 3 örnekte de görülebileceği gibi standart sembol dağılımında 1 adet sembol her iterasyonda fazladan düğümlere dağıtılmak üzere eklenmektedir. Diğer yandan artık verili sembol dağıtım yöntemlerinde ise bu fazladan sembol tutma durumu gözlemlenmemektedir ve hafıza alanından tasarruf sağlanmaktadır. İterasyon sayısı arttıkça tasarruf yapılan sembol sayısı da artmaktadır. Diğer yandan sembol onarımlarında gerçekleştirilen işlemlerin süresi artık verili dağıtım yöntemlerinde, standart dağıtım yöntemlerine göre daha uzun sürmektedir. Çünkü artık veri kısmında gerçekleştirilen XOR operasyonları standart versiyonda olduğu gibi geçici bellek (buffer) üzerinden toplu olarak gerçekleştirilememekte ve iterasyonlarda tek tek

gerçekleştirilmektedir. Bu da onarım zamanını etkilemektedir. İkinci artık verili sembol dağıtım yöntemi bu onarım sürelerini iyileştirmek adına geliştirilmiştir.

3.8.3 Önerilen algoritmanın ağ benzetim ortamında ns-3 ile ilgili çalışmalar

Bu iş paketi kapsamında önerilen Greepair algoritması ns-3 ağ benzetim ortamında **(ns-3, 2023)** geliştirilmiş ve yerel benzetim sonuçları ile kıyaslanmıştır.

Ağ topolojisi aşağıdaki Şekil 23'deki gibi tek bir hücre içerisinde düğümler arasında IEEE 802.11a standardına göre Ad Hoc ağı kurulmuştur ve kapsama ağının içerisinde olan düğümler baz istasyonu ile LTE ağı aracılığıyla haberleşebilmektedir. Bu topoloji modelinde düğümler iki boyutlu düzleme rastgele bir şekilde dağıtılacak biçimde oluşturulmuştur.



Şekil 9: Ağ benzetim ortamının kullandığı topoloji modeli.

Sistem modeli için bir hücresel ağ içerisinde depolama düğümleri ve boş düğümlerden oluşan toplamda N düğüm düşünülmüştür ve bu düğümlerin baz istasyonu (Bİ) yardımı olmadan birbirleriyle doğrudan veri alışverişi yapabildikleri (D2D) varsayılmıştır. Ayrıca bu hücresel ağ içerisinde ihtiyaç halinde kullanılmak üzere bir adet Bİ olduğu varsayılmıştır. Önbellekleme için F sembol (bayt) büyüklüğünde bir dosya kullanılmıştır ve bu dosya kodlanarak genişletildikten sonra (encoding)

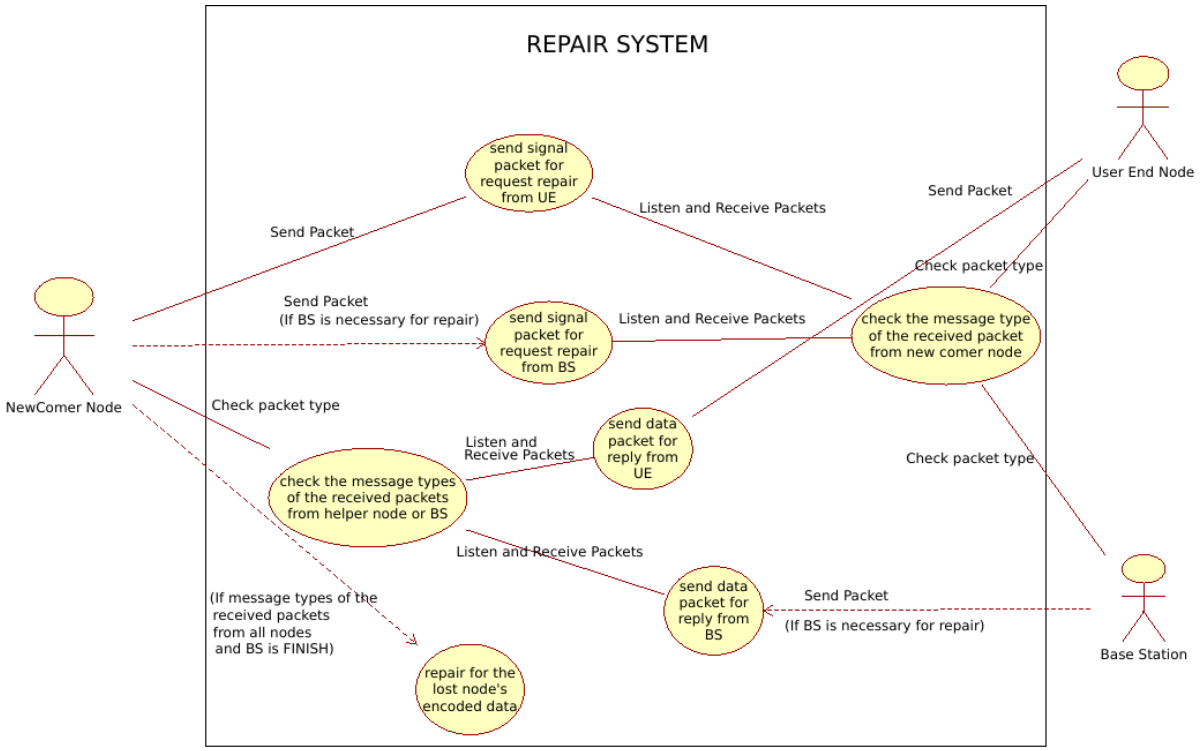
$m \leq N$ bağıntısını koruyacak şekilde m adet depolama düğümüne dağıtılmıştır. Bu genişletme sürecinde dosya aşamalı olarak başlangıçta k eşit boyutlu veri parçalarına bölünür ve sonrasında $n > k$ bağıntısına göre n adet sembole genişletilir. Bu n adet sembol de m adet depolama düğümüne eşit şekilde dağıtılır. Bir diğer deyişle, her depolama düğümü eşit sayıda olacak şekilde her iterasyonda n / m kadar sembol alır.

Şekil 7'de sistem modeline dair örnek bir onarım gösterimi verilmiştir. Hücresel ağ içerisine yeni gelen düğümler herhangi ön belleklerinde veri bulundurmazlar ve boş düğüm statüsünde

değerlendirilirler. Burada ρ_{D2D} yardımcı bir düğümden bir sembolün indirilme maliyetini, ρ_{BS} ise baz istasyonundan (Bİ) bir sembolün indirilme maliyetini ifade etmektedir. N başlangıçta hücresel ağ içerisindeki toplam düğüm sayısını ifade ederken λ bir düğümün hücresel ağ içerisine girme olasılığını ifade etmektedir. Diğer yandan μ ise bir düğümün hücresel ağ içerisinden ayrılma oranını ifade eder. Hücresel ağdaki toplam düğüm sayısını N ortalamasında tutmak amacıyla λ ve μ değerleri eşit alınmıştır. Sistem modelimizde kaybolan içeriği hemen onarmak yerine (instantaneous repair) tembel onarım (lazy repair) olarak bilinen periyodik Δ zaman aralıkları sonunda onarım işlemleri başlatılmıştır. Düğüm onarımları Δ zaman aralıklarında tembel onarım yöntemi kullanılarak gerçekleştirilmiştir

Düğüm tamir modelinde user end (UE) düğümler hücre içerisindeki sıradan önbelleklemeye sahip düğümleri temsil etmektedir. Newcomer düğüm ise hücre içerisinde bulunan ama halihazırda önbellekleme için kullanılmayan ve belli bir süre içerisinde kayıp düğümlerin yerine geçmek için seçilmiş bir düğümü temsil etmektedir. Baz istasyonu ise user end düğümlerdeki sembollerin yeterli gelmediği zaman kopya çekilecek ham sembollerin çekilebildiği düğümler temsil etmektedir. Bu topoloji modelinde tamir modelinin use case diyagramı aşağıda verilmiştir. Newcomer düğüm tamir işlemini başlatmak için baz istasyonu ve userend düğümlere gerekli çağrı mesajlarını göndererek kayıp düğümün depoladığı veriyi yeniden oluşturmaya çalışmaktadır. Basitlik açısından topoloji modeli ve use case diyagramında sadece bir adet newcomer düğüm gösterilmesine rağmen delta süresi içerisinde birden fazla newcomer düğüm ayrık ya da ayrık olmayan yardımcı UE'ler kullanarak tamir işlemlerini gerçekleştirmektedir. Şekil 8'deki use case diyagramında tamir işleminin adımları daha ayrıntılı bir şekilde görülmektedir. Burada kayıp bir düğümün verisinin yeniden oluşturulması yani tamiri işlemini gerçekleştirecek olan newcomer düğüm tamir için gerekli sembollerin neler olduğuna karar vermekte ve gerekirse de baz istasyonundan da ham sembol çekimi işlemi gerçekleştirmektedir. Bu durumlar newcomer düğümı **(Haytaoglu v.d., 2022)** makalesinde verilen Greepair algoritmasına çalıştırarak kara vermekte ve gerekli yardımcı UE düğümlerine ve gerekirse baz istasyonuna istediği sembolleri bildirmektedir. Tamir isteği ile ilgili mesajı alan düğümler kendilerinde depoladıkları istedikleri sembolleri kullanılan ağ arayüzü üzerinden; UE ise wifiAdhoc, baz istasyonu ise LTE üzerinden TCP paketleri kullanarak göndermektedir. Bu düğüm tüm cevapları aldığında Greepair algoritmasına göre tamir etme işlemini gerçekleştirmekte ve benzetimin ileriki sürelerinde depolama işi yapan bir UE olarak yaşamına devam etmektedir. Bu düğüm ileride başka kayıp düğümlere yardımcı UE düğüm olabildiği gibi kendisi de kayıp düğüm haline gelebilir. Bu işlemler her Δ süresi sonunda markov modeline göre hücreden ayrılan depolama

düğümlerinin kayıp sembollerini oluşturmak için seçilen newcomer düğümler tarafından eş zamanlı olarak gerçekleştirilmektedir.



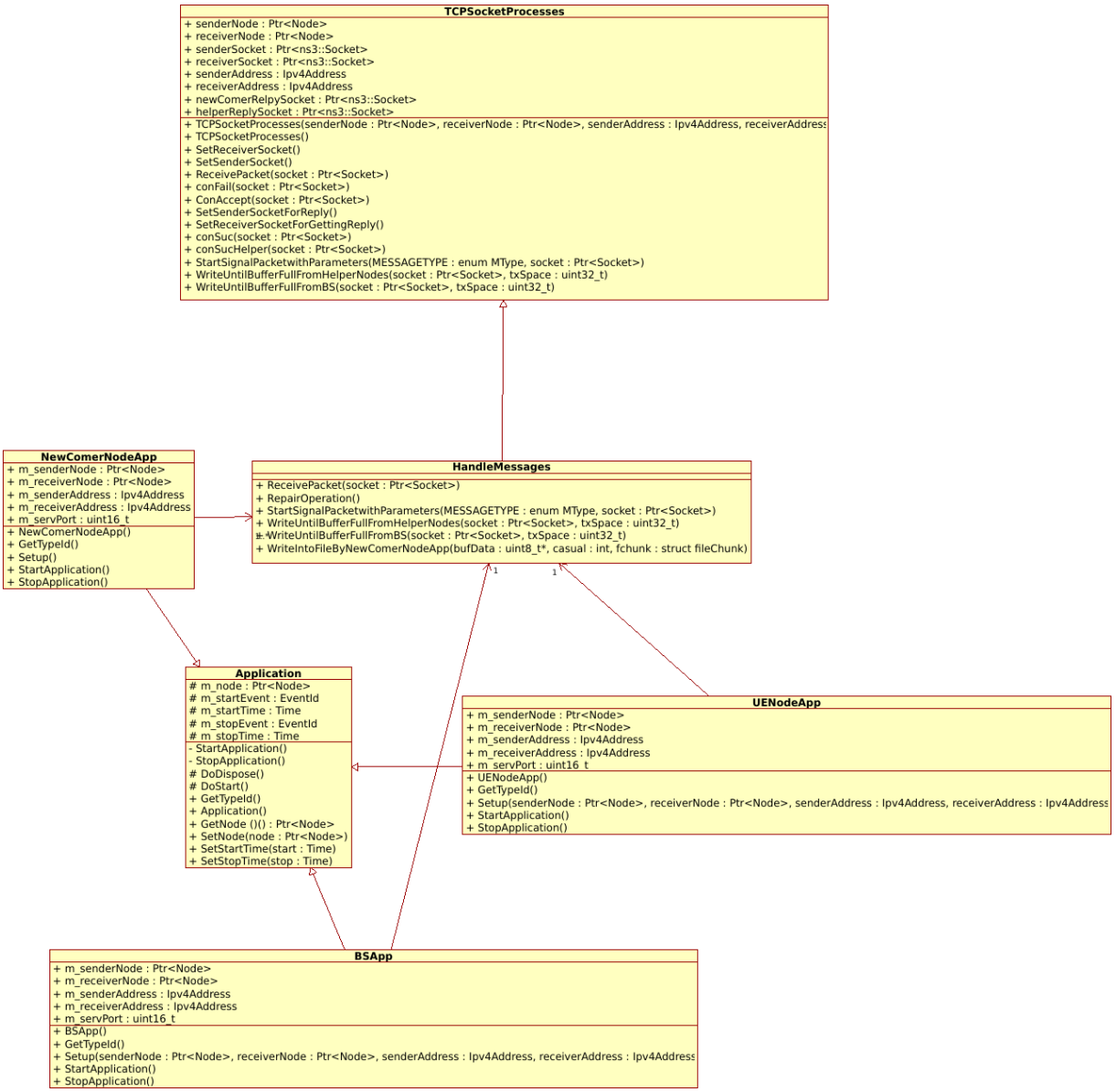
Şekil 10: Düğüm tamiri use case diagramı.

Ağ benzetim ortamında beş adet temel sınıf gerçekleştirilmiştir. Bunlar aşağıdaki gibi sıralanabilir.

- TCPSocketProcesses,
- HandleMessages,
- UENodeApp,
- NewComerNodeApp
- BSApp

Bu sınıflardan ilk iki tanesi düğümler arası paket gönderimi için kullanılan sınıflardır. TCPSocketProcesses sınıfı düğümler arası TCP soketlerinin açıldığı ve bu soketlerin bağlantılarının yapıldığı; HandleMessages sınıfı ise TCPSocketProcess'inden türetilmiş olup; düğümler arası gelen paketlerinin türlerine göre ayrıştırıldığı ve bu türlere göre düğüm tamirinin ilgili işlemlerinin gerçekleştirildiği bir sınıftır. Bunların yanında, NewComerNodeApp, UENodeApp ve BSApp ise user end ve baz istasyonunu düğümlerinin çalıştırdığı uygulamaları içermektedir. Bu uygulamalar iletişim ve tamir işlemi mekanizmasını gerçekleştirebilmek için HandleMessages sınıfını kullanmaktadır.

Aşağıdaki Şekil 10 ’da bu sınıfların kısaltılmış yani sadece gerekli yerlerinin verildiği UML diyagramı verilmiştir.



Şekil 11: Ağ Simülatöründe gerçekleştirilen UML Sınıf Diyagramı.

NewComerApp sınıfında birden çok HandleMessages sınıfı bulunmakta iken BSApp ve UENodeApp’de tek bir tamir süreci için newcomer düğüm ile tek bir iletişim kanalı olacağından bir adet HandleMessages sınıfı üyesi bulunmaktadır. Bunun dışında markov modelinin oluşturulması içinde ayrı sınıflar oluşturulmuştur. Newcomer düğüm ve UEler arasında gönderilen paket türleri aşağıdaki şekildedir:

- REQUESTREPAIRFROMUE = 1, //User End'den tamir için semboller istenilebilir
- REPLYREPAIRFROMUE = 3, //User end'den bize cevaplar gelebilir. Yani user end node'lardan paket gelirken tag bu olacak.
- FIN = 4 UElerden gelen kod paketlerinin bittiğini gösterir.

Newcomer düğüm ve Bİ arasındaki mesaj türleri de aşağıdaki listelenmiştir:

- REPLYREPAIRFROMBS = 2, //Bİ'den newcomer'a ham semboller geldiğinde tag bu olacak.
- REQUESTREPAIRFROMBS = 0, //BS'den requestRepair yapılabilir yani baz istasyonundan paket istenilebilir
- REQUESTSENDINGFROMUETOBS = 5,
- REPLYSENDINGFROMUETOBS = 6

Bu mesaj türleri kullanılarak tamamen dağıtık bir şekilde newcomer düğümler ve Bİ ve yardımcı UE' düğümleri kayıp düğümlerin içeriklerinin tamir edilmesi işlemlerini gerçekleştirmektedir. Aynı anda faklı newcomer düğümler eş zamanlı tamir işlemleri başlatabildikleri gibi, bu newcomer'ların yardımcı UE düğümleri arasındaki kesişim kümesi boş olmayabilir. Bu durumda yardımcı UE düğümlerinin farklı newcomer düğümleri ile HandleMessages nesneleri farklı olacağından tamir işlemleri birbirini bozmadan gerçekleştirilebilmektedir.

Daha detaylı inceleme için ağ benzetim ortamının kaynak koduna **(kaynakKodu, 2023)**'ten erişebilirsiniz.

4. Bulgular

Bu bölümde proje boyunca elde edilen bulgularımız özetlenecektir. Detaylar için gelişme raporlarına bakılmasını rica ederiz.

4.1 Protokol Tasarımı (Tahmin Teorisi ile ilgili Bulgular)

Diyelim ki bir rastlantısal değişken X, şöyle bir setten $\{x_1,x_2,...,x_M\}$ değerleri $\{p_1,p_2,...,p_M\}$ olasılıklar ile alsın. Burada her bir seçimin maliyeti c_j, $1\leq j\leq M$ ile gösterilsin. Başlangıç aşamasındaki çalışmamızda herhangi bir optimal tahmin fonksiyonunun ortalama tahmin maliyetini minimum yapması için $c_i\ p_j\leq c_j\ p_i$ ilişkisini bütün $i,j\in\{1,2,...,M\}, i\leq j$ için

sağlaması gerektiği gösterilmiştir. Herhangi verilen bir sıralamayı optimal seçime dönüştüren algoritmamız aşağıdaki gibi özetlenebilir.

Algoritma 4: Optimal Strateji

function OptimalCostGuess(p, c)
$M \leftarrow |\mathrm{p}|$
$\mathcal{I} \leftarrow \{1, 2, 3..., M\}$ ▷ Selection Order
$swapped \leftarrow true$
$i \leftarrow 1$
while $swapped$ **do**
    $swapped \leftarrow false$
    for $j = 1 : M - i$ **do**
        if $c_j p_{j+1} > c_{j+1} p_j$ **then** ▷ If condition holds
            swap(c_j, c_{j+1}),swap(p_j, p_{j+1}),swap$(\mathcal{I}_j, \mathcal{I}_{j+1})$
            $swapped \leftarrow true$
        end if
    end for
    $i \leftarrow i + 1$
end while
return $\mathcal{I}$

Bunun yanında ortalama tahmin maliyeti E[C_G(X)^ρ] ile gösterildiği durumda, aşağıda Tablo 4'te belirtilen momenti ρ − için verilen üst ve alt limitler bulunmuş ve detaylı şekilde karşılaştırılmış ve iyileştirmeler açıkça gösterilmiştir.

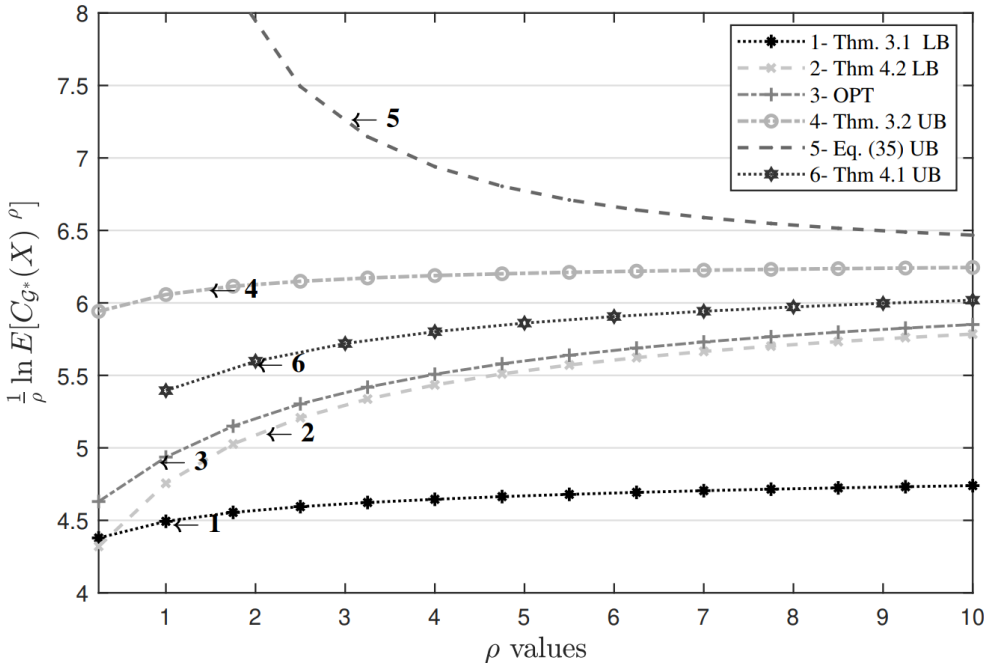
Tablo 4: Ortalama tahmin maliyeti için üretilmiş alt ve üst sınırlar.

$\mathbb{E}[C_{\mathcal{G}}(X)^\rho] \geq \mathbb{E}[C_{\mathcal{G}^*}(X)^\rho] \geq \left(\frac{M}{1+\gamma^*}\right)^{-\rho} \exp\left\{\rho H_{\frac{1}{1+\rho}}(X)\right\}$	**(1) Alt Sınır**
$\mathbb{E}[C_{\mathcal{G}}(X)^\rho] \geq \mathbb{E}[C_{\mathcal{F}}(Z)^\rho] \geq \sup_{\beta\in(-\rho,\infty)-\{0\}} \exp\left\{\frac{\rho}{\beta}\left[H_{\frac{\beta}{\beta+\rho}}(Z) - \log u_{\sum_x \lfloor c_x \rfloor}(\beta)\right]\right\} = \sup_{\beta\in(-\rho,\infty)-\{0\}} \left[u_{\sum_x \lfloor c_x \rfloor}(\beta)\right]^{-\frac{\rho}{\beta}} \exp\left(\frac{\rho}{\beta} H_{\frac{\beta}{\beta+\rho}}(Z)\right)$	**(2) Alt Sınır**
$\mathbb{E}[C_{\mathcal{G}^*}(X)^\rho] \leq \exp\{\rho H_{\frac{1}{1+\rho}}(Y)\}$	**(4) Üst Sınır**
$\mathbb{E}[C_{\mathcal{G}^*}(x)^\rho] \leq \frac{1}{1+\rho}\left[\exp\left\{\rho H_{\frac{1}{1+\rho}}(Y)\right\} - 1\right] + \exp\left\{(\rho-1)^+ H_{1/\rho}(Y)\right\}$	**(5) Üst Sınır**
$\mathbb{E}[C_{\mathcal{G}}(X)^m] \leq \frac{1}{\bar{c}_X(m+1)(m+1)}\left[\left[\sum_{k=1}^{M'} q_k^{\frac{1}{m+1}}\right]^{m+1} + \sum_{l=0}^{m-1}\binom{m+1}{l}\mathbb{E}[C_{\mathcal{G}}(X)^l](-\bar{c}_X(m+1))^{m+1-l}\right]$	**(6) Üst Sınır**

Burada $(z)^+ \triangleq \max\{z, 0\}$, $H_\alpha(X)$ -derecesinde Rényi's entropisini **(Campbell, 1965)** ifade ederken, $\gamma \approx 0.5772$ Euler Mascheroni sabiti için $u_{\sum_x \lfloor c_x \rfloor}(\beta)$ aşağıdaki gibi verilebilir. Burada $\beta > 1$ için, $\zeta(\beta) = \sum_{n=1}^{\infty} \frac{1}{n^\beta}$ Rienmann'ın Zeta fonksiyonunu göstermektedir.

$$u_{|\mathcal{Z}|}(\beta) = \begin{cases} \ln|\mathcal{Z}| + \gamma + \frac{1}{2|\mathcal{Z}|} - \frac{5}{6(10(|\mathcal{Z}|)^2+1)} & \beta = 1 \\ \min\{\zeta(\beta) - \frac{(|\mathcal{Z}|+1)^{1-\beta}}{\beta-1} - \frac{(|\mathcal{Z}|+1)^{1-\beta}}{2}, u_{|\mathcal{Z}|}(1)\} & \beta > 1 \\ 1 + \frac{1}{1-\beta}\left[(|\mathcal{Z}| + \frac{1}{2})^{1-\beta} - \left(\frac{3}{2}\right)^{1-\beta}\right] & |\beta| < 1 \\ \frac{(|\mathcal{Z}|)^{1-\beta}-1}{1-\beta} + \frac{1}{2}(1 + |\mathcal{Z}|^{-\beta}) & \beta \leq -1 \end{cases}$$

Bu ifadeler ışığında **(Sason ve Verdú, 2018)'** çalışmasında farz edildiği gibi olası her seçimin olasılığının a=0.9 ve M=32 kabul edildiği geometrik dağılımı $P_X(x) = (1-a)a^{x-1}/(1-a^M)$ şeklinde ifade edilebilir. Masrafların ise kesimlenmiş (1 ile 100 arasında) Gauss dağılımı ($\mu = \sigma^2 = 16$) kabul edilmiştir. Bu durumda ortaya çıkan alt ve üst sınırlar, gerçek sonuç (OPT) ile beraber Şekil 12'de verilmiştir.



Şekil 12: Bu görselde ortalama masrafın (OPT) bulunan alt ve üst limitler ile karşılaştırılması verilmiştir.

Görüleceği üzere (2) numaralı alt sınır (1) numaralı alt sınıra göre sırası ile %5.842 (ρ = 1 için) ve %18,473 (ρ = 5 için) daha iyi sınırlar bulmaktadır. Ayrıca elde edilen yeni üst sınırlar (4) numaralı üst sınıra göre %10.935 ve %5.541 (sırasıyla ρ = 1 ve ρ = 5 için) daha düşük değerler sunmaktadır. Her ne kadar (6) numaralı üst sınır sadece tam sayılı değerleri için geçerli olsa da, diğer bütün sınırlar herhangi bir reel değeri için geçerli olması açısından önem arz etmektedir. Bunun yanında daha önceki bir ISIT bildirimizde önerdiğimiz (5) numaralı üst sınırımızın aslında iyileştirilebilir olduğu da gösterilmiş durumdadır.

4.2 Çizge Tabanlı Kodlarının Farklı Protokoller ile Depolama-Bantgenişliği Limitleri

Geleneksel olarak olarak üretilen veri depolama kodları cebirsel yapılara sahip durumdadırlar. Bu projenin de temel amaçlarından biri olan çizge kodları ile veri depolama sistemleri için onarım bant aralığına riayet eden ve cebirsel sistemler gibi veri geri çatımını verimli (hem karmaşıklık hemde veri depolama yeri açısından) şekilde çalışması gerçekleştirilmiştir. Bunun yanında, cebirsel yapılar temelinde oluşan klasik yenileme kodlarının çözümleme karmaşıklığı hala çözülmesi gereken bir sorun olarak durmaktadır **(Park vd., 2017)**. Bir alternatif olarak, düşük çözümleme karmaşıklığı sunan düşük yoğunluklu parite kontrol kodları (LDPC) kullanılması önerilmiştir.

4.2.1 Temeller

Genel olarak, bir LDPC kodu bir $m \times n$ eşitlik denetim matrisi $H = \left(h_{i,j}\right)$ tarafından tanımlanabilir veya eşdeğer olarak, değişken düğümleri $v_j, j = 1, \ldots, n,$ ve denetim düğümleri $c_i, i = 1, \ldots, m,$ içeren bir Tanner çizgesi tarafından tanımlanabilir. Değişken düğümü v_j ve denetim düğümü c_i arasında $h_{i,j} = 1$ olduğunda bir kenara (edge) tekabül etmektedir. Derece dağılımlarına dayalı iki tür LDPC kodu vardır: Düzenli ve düzensiz **(Richardson ve Urbanke, 2008)**. Bir Tanner çizgesinde, maksimum değişken düğümü ve maksimum denetim düğümü dereceleri ayrıca $d_{v_{max}}$ ve $d_{c_{max}}$ olarak belirtilmiştir. Düzensiz LDPC kodlarının bir kümesi genellikle düğüm derecesi dağılımları kullanarak tanımlanır. Kenar-açısından değişken (variable) ve kontrol (check) düğüm derecesi dağılımları sırası ile $\lambda(x) = \sum_{d=2}^{d_{v_{max}}} \lambda_d x^{d-1}$ ve $\rho(x) = \sum_{d=2}^{d_{c_{max}}} \rho_d x^{d-1}$ şeklinde verilebilirler, burada λ_d (sıra ile. ρ_d) derece-d olan tüm değişken (kontrol) düğümlerinin tüm kenarlara oranıdır. Tanner çizgesi içindeki tüm kenarlara **(Richardson ve Urbanke, 2008)** bu nedenle, ortalama değişken düğüm derecesi ve ortalama denetim düğüm derecesi $\bar{d}_v = \left(\sum_{i=2}^{d_{v_{max}}} \frac{\lambda_i}{i} \right)^{-1}$ ve $\bar{d}_c = \left(\sum_{i=2}^{d_{c_{max}}} \frac{\rho_i}{i} \right)^{-1}$ olarak belirtilir. Bu çalışmada, notasyon basitliği için λ (sıral ile. ρ) ve $\lambda(x)$ (sıra ile. $\rho(x)$) değişmeli olarak kullanılacaktır. Ayrıca, bir LDPC kodunun kod oranı λ ve ρ 'ye bağlı olarak açık bir fonksiyon olarak verilebilir, yani $R(\lambda, \rho) = 1 - \frac{\int_0^1 \rho(x)dx}{\int_0^1 \lambda(x)dx}$. Alternatif olarak, kod oranı $R = 1 - \frac{\bar{d}_v}{\bar{d}_c}$ şeklinde ifade edilebilir. Depolama sistemlerinde veri kaybı en iyi ikili silme kanalı (BEC) modeli **(Park vd., 2017)** ile açıklanabilir, bu nedenle yoğunluk evrimi, yani $x_l = \epsilon\lambda\left(1 - \rho\left(1 - x_{l-1}\right)\right)$, inanç yayılması (BP) çözümlemesi altında bir LDPC kodunun silme düzeltme yeteneğini açıklamak için kullanılır. Yoğunluk evriminde, $x_0 = \epsilon$ girdi kanalı silme olasılığıdır ve x_l, BP çözümleme algoritmasının l tane iterasyonu tamamladıktan sonra herhangi bir bilgi bitinin silme olasılığıdır **(Richardson ve Urbanke, 2008)**. Ayrıca, bir LDPC kodunun çözüm eşiği(decoding threshold) ϵ^* , $\epsilon^* = sup\left\{\epsilon \in (0, 1] : x_l = 0\right\}$ şeklinde belirtilebilir. Burada sup supermum operatörünü belirtmektedir. Son olarak, **(Dimakis vd., 2010)**'da varsayılan gibi, F bit boyutunda veri nesneleri $n\alpha$ bit'e kodlanır ve sonra n depolama

düğümüne dağıtılır eşit şekilde dağıtılır. Burada α düğümün depolama boyutudur. Bu iş paketi içerisinde, konuyu daha iyi anlama adına ve işlemleri basitleştirmek için $\alpha = 1$ olarak alınmıştır.

4.2.2 Onarım bantgenişliği

Bu bölümde, iki farklı tamir protokolü altında LDPC kodların ortalama tamir bant genişliği limitlerini belirleyeceğiz. Bu tamir protokollerinin biri onarım için yardımcı düğümlerin rastgele seçtiği diğeri ise "guessing" bağlamında optimal strateji ye denk gelmektedir. Bu sonraki protokol detaylı şekilde **(Arslan ve Haytaoglu, 2020)**, **(Arslan ve Haytaoglu, 2022)** bildirilerimizde ve revizyonda olan (**Arslan ve Haytaoglu, 2020 b.**) makalemizde anlatılmış ve bilgi teorisi açısından incelenmiştir. Bu alt bölüme bir tanım ile başlayalım.

Tanım 1. (Rastgele erişim onarım protokolü) Herhangi bir değişken düğümü arızalandığında, yeni gelen düğüm arızalı düğümün verilerini eşitlik denetim denklemleri tarafından tanımlanan mevcut değişken düğümler setinden rastgele seçerek yeniden oluşturur **(Park vd., 2017)**. Rastgele erişim tamir protokolünde derecesi $d_{c,r}$ olan bir kontrol düğümü $r \in \{1, 2, \ldots, m\}$, $d_{c,r}$ adet değişken düğümüne bağlıdır. Bu nedenle, her değişken düğüm için ortalama tamir bant genişliği, kapalı formda $\bar{\gamma} = \sum_{i=1}^{m} \frac{d_{c,i}\left(d_{c,i}-1\right)}{E}$ olarak ifade edilebilir, burada m toplam denetim düğümü sayısıdır ve E ilişkili Tanner çizgesi içinde toplam kenar sayısını ifade etmektedir. **(Park vd., 2017)**'de, kontrol düğümlerinin düzenli olduğunda $\bar{\gamma}$'nin minimumu elde edilebileceği gösterilmektedir. Aşağıda, sabit E için, $\bar{\gamma}$'nin minimuma indirebilen $\rho(x)$'i belirlemek için bir prosedür/algoritma önerilmektedir. **(Richardson ve Urbvabke, 2001)**'de, çözümleme eşiğini maksimize etmek için yakın optimum derece dağılımları tasarlamak için yazarlar, sürekli dereceleri (tam sayılı kısıtların bir tür rahatlama oluşturması ile) kullanmanın önerildiğini, bu fraksiyonel phantom dağılımı kullanarak yorumlanabileceğini belirtmiştir. Burada $\bar{\gamma}$'nin minimumunu belirlemek için benzer bir teknik kullanmaktayız. Öncelikle bir kuram ile başlayabiliriz.

Kuram 1: LDPC kod kümesi için, ortalama tamir bant genişliği minimumuna olan $\bar{\gamma}_{min}^{(rand)} = d - 1$ ulaşmak için, kontrol düğüm derecesi dağılımı aşağıdakini sağlamalıdır:

$$\rho(x) = \begin{cases} x^{d-1}, & d \in \mathbb{Z}, \\ \left(\lceil d \rceil - d\right) x^{\lfloor d \rfloor - 1} + \left(d - \lfloor d \rfloor\right) x^{\lfloor d \rfloor}, & d \notin \mathbb{Z}. \end{cases} \quad (1)$$

Bu kuramın ispatı **(Pourmandi vd., 2023)** makalemizde bulunabilir. Hem kontrol düğümleri hem de değişken düğümlerinin tümünün düzenli olduğu durumda $\bar{\gamma}_{min}^{(rand)}$ değerinin $d_{c_{max}} = \frac{d_{v_{min}}}{1-R}$ ve $d_{v_{max}} = d_{v_{min}}$ ile elde edilebileceği gözlemlenebilir. Burada $d_{v_{min}}$, değişken düğüm derecelerinin minimum değerini tanımlamaktadır **(Park vd., 2017)**. Yine aynı şekilde $d_{v_{min}} = 2$ kullanıldığında, $\bar{\gamma}_{min}^{(rand)}$ değeri $\frac{2}{1-R} - 1$ olarak elde edilebilir. Ancak, bu sonuç kontrol düğümlerinin düzensiz olduğu durumda uygulanabilir değildir. Kod oranı- R ve ϵ^* arasındaki çarpan fark δ **(Park vd., 2017, Tanım. 3.84)** şeklinde tanımlanır ve $R = (1 - \delta)\left(1 - \epsilon^*\right)$ için özel bir pozitif sayıdır, $\delta \geq \frac{J\left(R,\bar{d}_c\right)}{\left(J\left(R,\bar{d}_c\right)+1\right)}$ ve $J\left(R,\bar{d}_c\right) = R^{\bar{d}_c - 1}(1 - R)$ **(Richardson ve Urbanke, 2008 - Thm. 3.85)**. $Q\left(R,\bar{d}_c\right) = \frac{J\left(R,\bar{d}_c\right)}{\left(J\left(R,\bar{d}_c\right)+1\right)}$ olarak tanımlanırsa, $Q\left(R,\bar{d}_c\right)$ R için artan bir fonksiyon olduğu görülür. Bu gözlem sonucu, $\bar{d}_c > 2$ sabit ve $\bar{d}_v \geq 2$ olduğunda $Q\left(R,\bar{d}_c\right)$ değişken düğümlerin $\bar{d}_v = 2$ ile düzenli olduğunda maksimuma ulaşacaktır. Bu nedenle, maksimum kod oranı için (yani $R = 1 - \frac{2}{\bar{d}_c}$), çözümleme eşiği ϵ^* en düşük değerine ulaşabilir. Aşağıdaki teoremde, sabit kod oranı- R ve $\epsilon^* = \epsilon_{min}^*$ için herhangi bir LDPC kodu için $\bar{\gamma}_{min}^{(rand)}$ kapalı bir formül sunulmaktadır.

Kuram 2. Rastgele erişim tamir protokolü kullanarak, bir kod oranı- R için, $\epsilon^* = \epsilon_{min}^*$ olduğunda minimum ortalama tamir bant genişliği $\bar{\gamma}_{min}^{(rand)}$, şöyle belirlenir.

$$\bar{\gamma}_{\min}^{(\mathrm{rand})} = f(R) - 1,$$

burada $f(R) = \lfloor \phi_R \rfloor + \lceil \phi_R \rceil - \frac{1}{\phi_R}\lfloor \phi_R \rfloor \lceil \phi_R \rceil$ *and* $\phi_R = \frac{2}{1-R}$. Aynı şekilde bu kuramın ispatı **(Pourmandi vd., 2023)** makalemizde bulunabilir.

Tanım 2. (Ideal Tamir Protokolü) Bu protokol içinde, bozuk düğümü kurtarmak/tamir etmek için denklem seçimi düğüm derecesine dayalı olarak gerçekleştirilir. Tamir için en az elemanı (en küçük dereceye sahip) olan düğümü (denklemi) seçmek bant genişliği maliyetini minimum tutacağı gösterilmiştir **(Arslan ve Haytaoglu, 2020)**.

Kuram 3. Verili $(\lambda(x), \rho(x))$ çifti tarafından tanımlanan bir LDPC kod kümesi için, ideal tamir protokolü kullanılarak bozuk bir düğümün ortalama tamir bant genişliği aşağıdaki gibi verilebilir.

$$\bar{\gamma}^{(\text{ideal})}(\lambda,\rho) = \bar{d}_v \sum_{j=2}^{d_{v\max}} \frac{\lambda_j}{j} \left(\sum_{\underline{n}\in C(j)} P(\underline{n}, j)\gamma_{\underline{n}} \right)$$

Burada $\underline{n} = \left\{ n_i : \rho_i > 0, i = 2, \ldots, d_{c_{max}} \right\}$, $C(j) = \left\{ \underline{n} : \sum_{i=2}^{d_{c_{max}}} n_i = j, \rho_i > 0 \right\}$ ve n_i i-dereceli kontrol düğümleri ile oluşan kenar sayısını ifade etmektedir. Ayrıca kenarların rastgele seçildiği durumda,

$$P(\underline{n}, j) = \frac{j!}{n_{i_1}! \ldots n_{i_{|\underline{n}|}}!} \prod_{k\in\{i_1,\ldots,i_{|\underline{n}|}\}} (\rho_k)^{n_k}$$

$\gamma_{\underline{n}} = min\left\{ i_1, \ldots, i_{\underline{n}} \right\} - 1$ ve $i_k \in \left\{ 2, \ldots, d_{c_{max}} \right\}$. Aynı şekilde bu kuramın ispatı **(Pourmandi vd., 2023)** makalemizde bulunabilir. Bu bölümde bulunan Tanım ve Kuramlar kod tasarımı ve ödünleşim uzayının belirlenmesi için yardımcı olacaktır.

4.2.2 Kod tasarımı ve numerik bulgular

Bu bölümde düşük tamir bant genişliği ve yüksek veri güvenilirliği özelliklerine sahip LDPC kod kümelerini bulmak için genetik optimizasyon yöntemlerine dayalı bir eniyileme tekniği önerilmektedir. Rastgele erişim tamir protokolü için, $\bar{\gamma}_{min}^{(rand)}$ bir sabit olarak varsayılırsa, **Kuram 1** kullanılarak $\rho(x)$ açıkça türetilebilir. Ancak, ideal tamir protokolü için, değişken düğüm derecesi dağılımının yanı sıra kontrol düğüm derecesi dağılımını belirlemek gerekir. Ayrışık (differential) evrim (DE) tabanlı optimizasyon tekniklerinin verimli LDPC kodlarının tasarımında yararlı olduğu kanıtlanmıştır **(Richardson vd., 2001), (Shokrollahi, 2000)**.

Bu altbölümde tamir verimli bir kod tasarımı aramak için DE algoritmasının bir varyantı kullanılmaktadır. DE'de ana amaç fonksiyonu korunurken, geri kalanlar kısıtlanır. Sonra, kısıtlamalar bir kısıtlama işleme yöntemiyle ele alınır ve $g(x) \geq g_0$ kısıtlaması, ilgili maliyeti, yani $max\left(1 - \frac{g(x)}{g_0}, 0\right)$, ana amaç fonksiyonuna bir çarpan kullanarak zorunlu hale getirilir **(Deb, 2001)** olarak formüle edilir, burada $\epsilon_t(\lambda, \rho)$ hedef çözümleme eşiğidir. Ancak ideal onarım tamir protokolü için, optimizasyon problemi epsilon kısıtlama yöntemi kullanılarak tek

bir amaçlı bir probleme dönüştürülür **(Deb, 2001)**. Örneğin, $\bar{\gamma}^{(ideal)}(\lambda, \rho)$ ana maliyet fonksiyonu olarak kabul edildiğinde, optimizasyon problemi şöyle formüle edilebilir:

$$(\hat{\lambda}, \hat{\rho}) = \arg \min_{(\lambda,\rho)\in S} \Big\{ \bar{\gamma}^{(\text{ideal})}(\lambda, \rho) + \xi \max(1 - \epsilon_t(\lambda, \rho)/\epsilon_0, 0) + \xi \max(1 - R(\lambda, \rho)/R_0, 0) \Big\}$$

Burada $S = \{(\lambda, \rho) : \sum_{i=2}^{d_{v_{\max}}} \lambda_i = 1, \sum_{i=2}^{d_{c_{\max}}} \rho_i = 1, \lambda_i \geq 0, \rho_i \geq 0\}$, $\epsilon_t(\lambda, \rho) \geq \epsilon_0$ ve $R(\lambda, \rho) \geq R_0$ sağlarken, ϵ_0 optimizasyonun tolerans gösterebileceği en küçük geriçatım eşik değerini temsil edecektir.

DE'de tek bir amacın gerçekleşmesine rağmen, üç temel kontrol parametresi vardır, çaprazlama katsayısı $C_r \in (0, 1)$, mutasyon katsayısı $\beta \in [0, 1]$ ve nüfus büyüklüğü NP **(Yang, 2020)**. DE/rand/1 ve DE/rand-to-best/1 gibi farklı mutasyon stratejileri bulunmaktadır **(Deng vd., 2021)**. Ayrıca, daha iyi bir performans göstermesi için aşağıdaki değişiklikler uygulanır.

(1) İlk olarak, sabit bir mutasyon stratejisi yerine, algoritma iki mutasyon stratejisi arasında rastgele olarak iterasyon yapar, sentetik bireylerin çeşitliliğini arttırır.

(2) İkincisi, tamsayı olan değişken düğüm dereceleri ve kontrol düğüm dereceleri, fraksiyonel phantom dağılımını kullanarak düğüm derecesi dağılımını ihlal etmeden gerçek derecelere dönüştürülür **(Richardson vd., 2001)**.

İdeal tamir protokolü için önerilen DE tabanlı algoritmanın pseudocode'u **Algoritma 5**'de özetlenmiştir.

Algoritma 5: $\gamma_{min}^{(ideal)}$ **'a yaklaşan** $(\lambda_s(x), \rho_s(x))$ **değerleri bulma.**

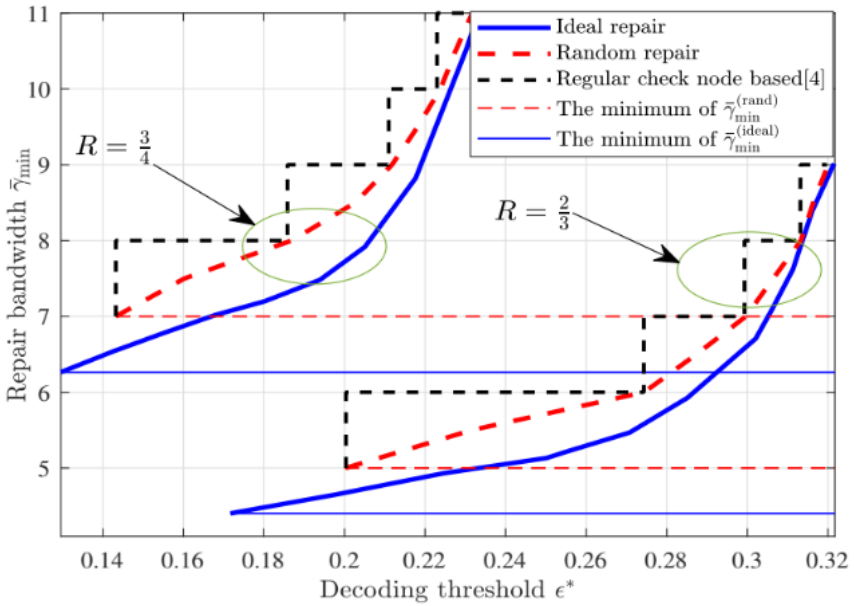
Input: Lower bound for code rate: R_0, lower bound for decoding threshold: ϵ_0, maximum variable node degree: $d_{v_{\max}}$, maximum check node degree: $d_{c_{\max}}$, number of non-zero variable node degree: $Numd_v$, number of non-zero check node degree: $Numd_c$, maximum number of iterations: $MaxIt$, population size: N_P, crossover coefficient: C_r
Output: $(\lambda_s(x), \rho_s(x))$

Generate initial random population with individuals $(\lambda_i(x), \rho_i(x))$, $i = 1, \ldots, N_P$ based on the fractional phantom distribution for given $d_{v_{\max}}$, $d_{c_{\max}}$, $Numd_v$, $Numd_c$ [15].
Calculate the input argument of (11) for each individual as its cost function for given R_0 and ϵ_0. ▷ $\epsilon_t(\lambda_i(x), \rho_i(x))$ is calculated by the bisection method.
Save the best solution $(\lambda_s(x), \rho_s(x))$ possessing the minimum cost function.
for $i = 1$ to $MaxIt$ **do**
  for $j = 1$ to N_P **do**
    Generate random mutation coefficients $\beta_j \in [0, 1]$.
    sFlag ← rand ▷ A binary random generator is used to switch between two different strategies.
    Create $\mathbf{v}_j = \mathbf{x}_{r_1} + \beta_j(\mathbf{x}_{r_2} - \mathbf{x}_{r_3} + \text{sFlag}(\mathbf{x}_{\text{best}} - \mathbf{x}_{r_1}))$, where the different indices $r_1 \neq r_2 \neq r_3$ are selected uniformly at random in $\{1, \ldots, N_P\}\backslash\{j\}$. ▷ $\mathbf{x_i}$ is the vector representation of $(\lambda_i(x), \rho_i(x))$.
    $(\lambda_t(x), \rho_t(x))$←Crossover$((\lambda_j(x), \rho_j(x)), \mathbf{v_j}, C_r)$
    if $cost_t < cost_j$ **then**
      Update j_{th} individual, $(\lambda_j(x), \rho_j(x)) \leftarrow (\lambda_t(x), \rho_t(x))$.
      if $cost_j < cost_s$ **then**
        Update s_{th} individual, $(\lambda_s(x), \rho_s(x)) \leftarrow (\lambda_j(x), \rho_j(x))$.
      end if
    end if ▷ Cost is computed based on (11).
  end for
end for

Son olarak daha önceden karakterize edilmiş olan optimizasyon problemi çözülerek, ortalama onarım bant genişliği (veya kod oranı) ile geri çatım ya da çözme eşiği (decoding threshold) arasındaki ödünleşim uzayı eğrisi bulunarak, görselleştirilecektir. Şekil 13'de, $R = \frac{2}{3}$ ve $R = \frac{3}{4}$ için LDPC kod aileleri için iki farklı protokol kullanarak minimum ortalama onarım bant genişliği ve ϵ^* için çözme eşiği arasındaki ödünleşim uzayını göstermektedir.

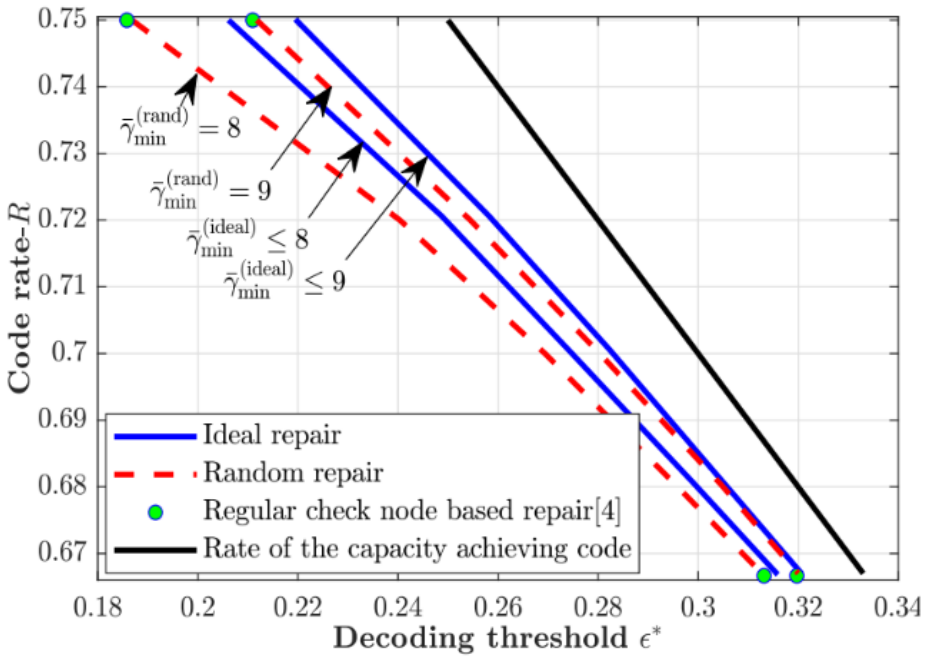
Şekil 13'de, rastgele protokolü ile onarım için, $\bar{\gamma}_{min}^{(rand)} \notin Z$ **(Park vd., 2017)** sonucuna kıyasla önemli ölçüde gerekli bant genişliği azaltılmış olduğunu gösterilmiştir. Bu durum, kod tasarımı optimizasyonlarında önerilen kontrol düğümlerinin derece seçimini rahatlatması ile etkili olduğunu göstermektedir. Şekil 14'de ayrıca, ideal onarım protokolünün rastgele onarım protokolünden bant genişliği açısından üstün olduğunu da gözlemleyebiliriz. Ayrıca, yüksek

kod oranları (code rate) beklenen şekilde daha büyük onarım bant genişliklerini gerektirmektedir. Ek olarak, $\bar{\gamma}_{min}^{(rand)}$ ve $\bar{\gamma}_{min}^{(ideal)}$ için alt sınırlar da aynı Şekle dahil edilmiştir.



Şekil 13: İki farklı protokol için onarım bant genişliği ile çözüm eşiği arasındaki ödünleşim ilişkisi. Ayrıca, düzenli kontrol dereceli düğümlü LDPC kodlarının performansıda ayrıca karşılaştırmalı olarak gösterilmiştir.

Şekil 14'de, iki sabit bant genişliği (8 ve 9) için kod oranı arttıkça çözme eşiğinin nasıl değiştiğini göstermektedir, burada maksimum kontrol düğüm derecesinin $d_{c_{max}} \leq 15$ olduğu varsayılmış, daha büyük üst limit değerler kullanıldığında herhangi önemli iyileştirme gözlemlenmemiştir.



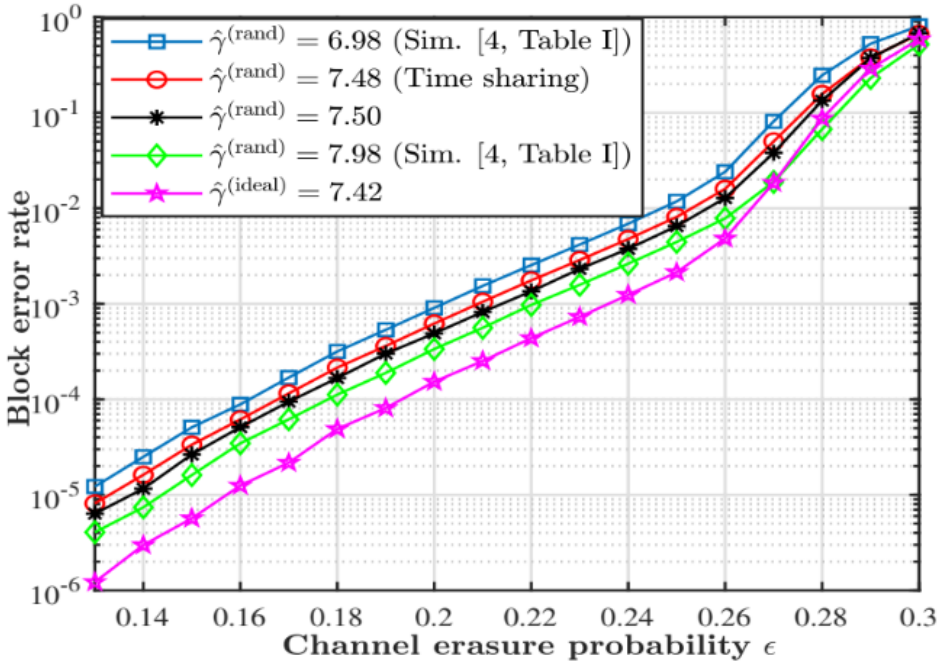
Şekil 14: Farklı iki onarım bant aralığı ve protokol için kod oranı ve veri geri çatımı eşik değeri arasındaki ödünleşim uzayı.

Görüldüğü gibi, kod oranı arttıkça, iki protokol arasındaki fark artmaktadır. Örneğin, $\frac{3}{4}$ kod oranıyla (0.75), ideal onarım protokolü eşik değeri (ϵ^*) en az 0.02'lik fazladan iyileştirme sağlamaktadır. İdeal onarım protokolünün performansı, daha çok veri depolama sistemleri için uygun olan yüksek kod oranlarında önemli iyileştirmeler göstermiştir. Aynı şekilde, silinti kanalları için kapasiteyi yakalayan kodun oranı (1 - ϵ^*) da aynı şekilde verilmiştir. Görüldüğü üzere, kod tasarımını düşük onarım bant genişliğine sınırlamak, kodların, özellikle yüksek kod oranları için, optimal veri geri çatılması özelliklerini kaybetmelerine neden olmaktadır. Bu gibi bir olay veri depolama açısından ne yapılabilir ve ne yapılamaz açısından ilginç bir perspektif sunmaktadır. Bunun yanında $\frac{2}{3}$ kod oranı (0.667) için, optimal derece dağılımları aşağıdaki Tablo 5'de verilmiştir.

Tablo 5: Kod oranı ⅔ olan kodlar için iki farklı protokol için optimal derece dağılımları

Protocol	ϵ^*	$\bar{\gamma}$	Node degree distributions
rand	0.306	7.5	$\lambda(x) = 0.4303x + 0.1918x^2 + 0.0707x^3 + 0.1913x^4 + 0.1159x^5$ $\rho(x) = 0.5x^7 + 0.5x^8$
ideal	0.312	7.39	$\lambda(x) = 0.3506x + 0.1673x^2 + 0.1704x^3 + 0.1568x^6 + 0.0540x^7 + 0.1008x^8$ $\rho(x) = 0.0095x^5 + 0.1477x^6 + 0.1138x^7 + 0.1459x^8 + 0.1847x^9 + 0.1575x^{10} + 0.1132x^{11} + 0.0355x^{12} + 0.0922x^{13}$

Kodların sınırlı uzunluklu (finite length rejim) veri geri çatım performansları da Monte Carlo simülasyonları kullanarak elde edilmiş ve Fig. 3'de sunulmuştur. Simülasyonlar, Tablo 5 ve **(Park vd., 2017)** (Bölüm 4, Tablo 1) de belirtilen, 8 ve 9 düzenli kontrol düğüm dereceleri verilen düğüm derece dağılımları ile elde edilmiştir. Diğer taraftan, Şekil 15'de, rastgele onarım protokolü için kodun veri geri çatımı özelliklerini negatif olarak etkileyen onarım bant genişliğinde bir azalmanın olduğunu gözlemlenmiştir. Ayrıca, kontrol düğümü düzensizliği sayesinde, neredeyse aynı (veya daha az) bant genişliği için, ideal protokol için tasarlanan kod, rastgele protokol için tasarlanan kodun veri geri çatımı performansından daha iyi olduğu açıkça gösterilmiştir. Bunun temel nedeni ideal onarım protokolünün doğası gereği inanç yayılımı geri çatım algoritmasına uygun olmasıdır.



Şekil 15: R = 2/3 için blok hata oranı ile kanal silme olasılığının karşılaştırılması. Simülasyonda, oluşturulan kodların blok uzunluğu 2000 olarak seçilmiştir. Blok hata oranı, bir inanç yayılımı algoritmasi çalıştırılarak elde edilmiştir. Onarım bant genişlikleri, değiştirilmiş PEG algoritması [20] kullanılarak eşlik kontrolünün elde edilen düğüm derecesi dağılımları tarafından belirlenmiştir.

Dolayısı ile bu sonuçlar ışığında denilebilir ki onarım bant genişliği iyileştirilen kodlarımız aynı zamanda sınırlı uzunluklu daha pratik şartlarda da önceki çalışmalarda önerilen kodlara göre veri geri çatımı performansı açısından daha iyi sonuçlar vermiştir.

4.3 Çizge Tabanlı Kodlarda Tekli Ve Çoklu Düğüm Onarımı Için Algoritma Ile Ilgili Bulgular

Baz istasyonu (Bİ) ve komşu cihazlardan sembollerin indirilerek düğüm (node) onarımlarının gerçekleştirildiği dağıtık bir önbellekleme sistemine dayalı bu çalışmada, LDPC kodları

(Gallager, 1962) kullanılarak düğüm onarım maliyeti (repair cost) analiz edilmiştir. Bu analizlere ek olarak yeni bir düğüm onarım algoritması LDPC kodlar üzerinden test edilerek önerilmiştir. Kullanılan LDPC kodları **(Eleftheriou ve Olcer, 2002)**, dizi LDPC kodları (array LDPC codes) ailesindendir ve RS kodları **(Reed ve Solomon, 1960)**, Minimum Bant Genişliği Yenileme (MBR) ve Minimum Depolama Yenileme (MSR) kodları **(Dimakis v.d., 2010), (Wang v.d., 2016), (Rashmi v.d., 2011)** ile onarım performansları açısından kıyaslanmıştır.

Yapılan çalışma kapsamında dağıtık veri önbelleklemede baz istasyonu etkileşimini minimize etmek ve hücresel ağ içerisindeki düğümler arasında cihazlar arası iletişimi ön plana çıkarmak amacıyla Greepair adlı yeni bir onarım algoritması önerilmiştir **(Haytaoglu v.d., 2022)**. Bu onarım algoritması kullanılarak da LDPC kodları klasik silinti düzeltme kodları ile onarım bant genişliği maliyeti (repair bandwidth cost) gibi parametreler üzerinden performans karşılaştırmalarına tabi tutulmuşlardır.

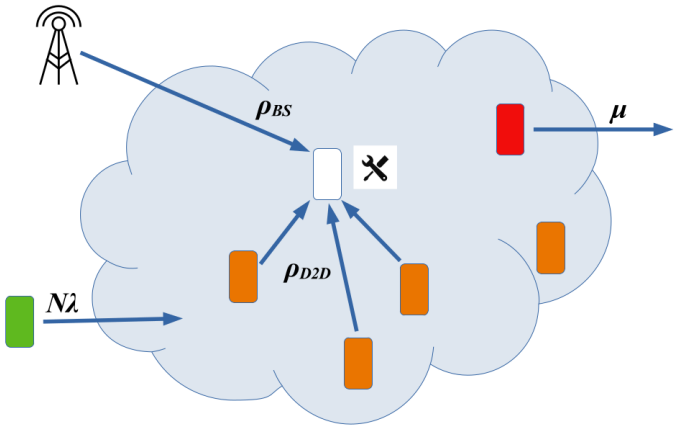
Bir simülasyon ortamında yardımcı düğümlerin yanı sıra Bİ'den de indirilen sembol sayıları açısından ortalamada teorik sınırlar belirtilmiştir. Ayrıca, LDPC tabanlı dağıtık önbelleklemede minimum bant genişliği maliyeti (minimum bandwidth cost) açısından optimum onarım stratejisi ile önerdiğimiz algoritma arasındaki performans farkı da gösterilmiştir.

4.3.1 Sistem modeli

Sistem modeli için bir hücresel ağ içerisinde depolama düğümleri ve boş düğümlerden oluşan toplamda N düğüm düşünülmüştür ve bu düğümlerin baz istasyonu (Bİ) yardımı olmadan birbirleriyle doğrudan veri alışverişi yapabildikleri (D2D) varsayılmıştır. Ayrıca bu hücresel ağ içerisinde ihtiyaç halinde kullanılmak üzere bir adet Bİ olduğu varsayılmıştır. Önbellekleme için F sembol (bayt) büyüklüğünde bir dosya kullanılmıştır ve bu dosya kodlanarak genişletildikten sonra (encoding)
$m \leq N$ bağıntısını koruyacak şekilde m adet depolama düğümüne dağıtılmıştır. Bu genişletme sürecinde dosya aşamalı olarak başlangıçta k eşit boyutlu veri parçalarına bölünür ve sonrasında $n > k$ bağıntısına göre n adet sembole genişletilir. Bu n adet sembol de m adet depolama düğümüne eşit şekilde dağıtılır. Bir diğer deyişle, her depolama düğümü eşit sayıda olacak şekilde her iterasyonda n / m kadar sembol alır.

Şekil 1'de sistem modeline dair örnek bir onarım gösterimi verilmiştir. Hücresel ağ içerisine yeni gelen düğümler herhangi ön belleklerinde veri bulundurmazlar ve boş düğüm statüsünde

değerlendirilirler. Burada ρ_{D2D} yardımcı bir düğümden bir sembolün indirilme maliyetini, ρ_{BS} ise baz istasyonundan (Bİ) bir sembolün indirilme maliyetini ifade etmektedir. N başlangıçta hücresel ağ içerisindeki toplam düğüm sayısını ifade ederken λ bir düğümün hücresel ağ içerisine girme olasılığını ifade etmektedir. Diğer yandan μ ise bir düğümün hücresel ağ içerisinden ayrılma oranını ifade eder. Hücresel ağdaki toplam düğüm sayısını N ortalamasında tutmak amacıyla λ ve μ değerleri eşit alınmıştır. Sistem modelimizde kaybolan içeriği hemen onarmak yerine (instantaneous repair) tembel onarım (lazy repair) olarak bilinen periyodik Δ zaman aralıkları sonunda onarım işlemleri başlatılmıştır.

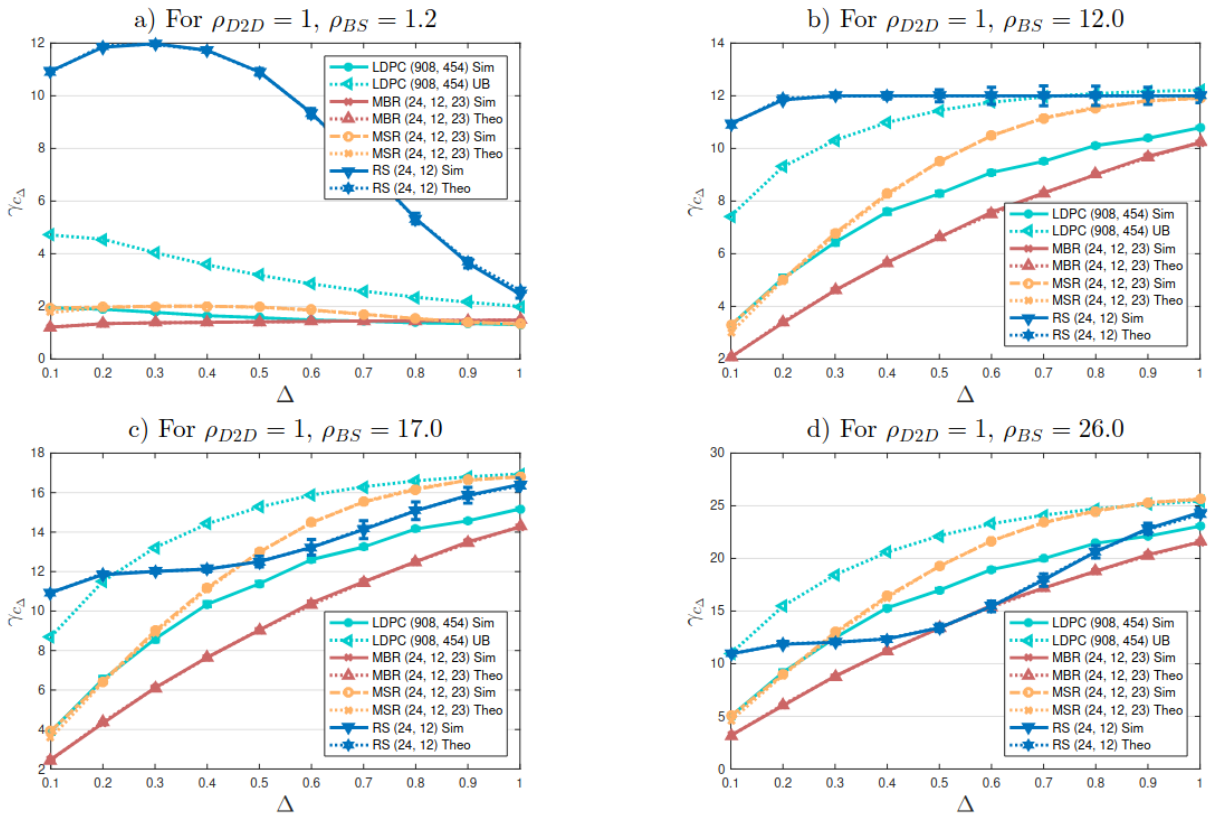


Şekil 16: Dağıtık depolamanın mobil cihazlar (düğümler) üzerinden gerçekleştirildiği bir hücresel ağ.

4.3.2 Simülasyon sonuçları

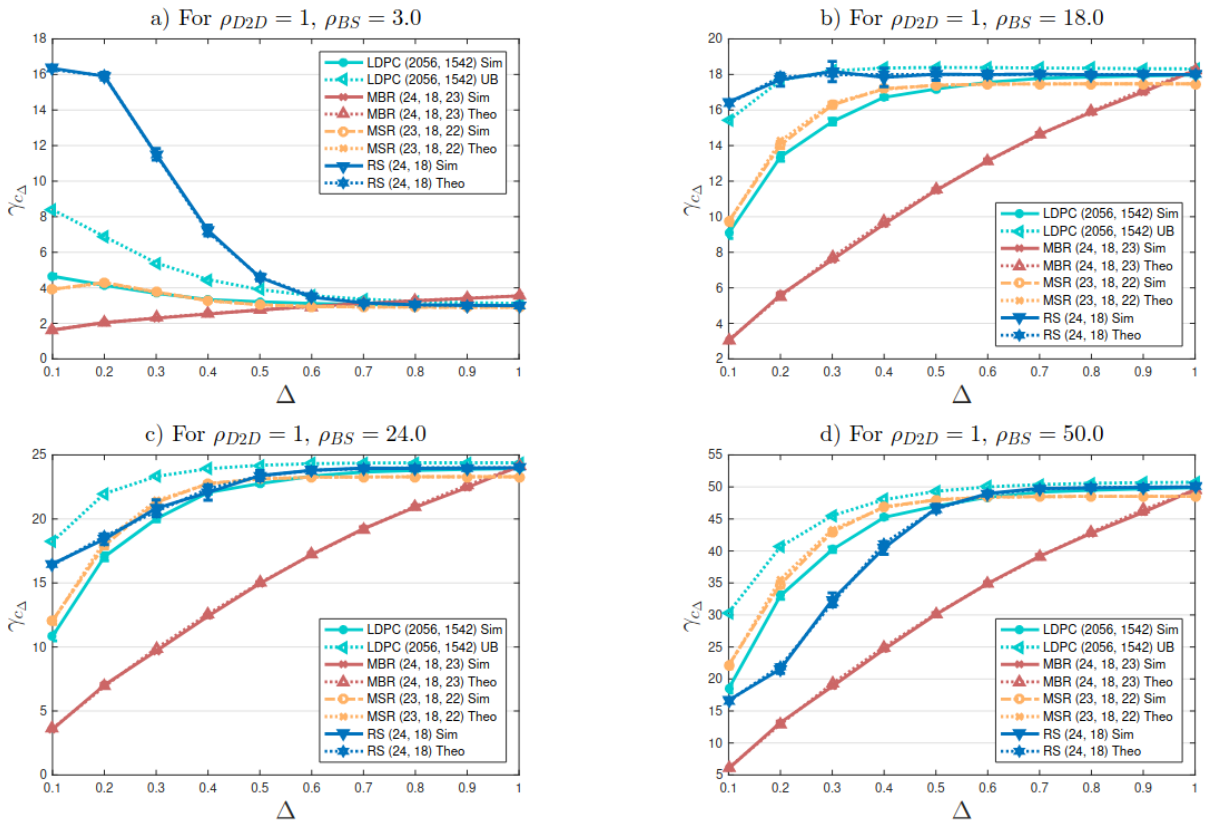
Simülasyon sonuçlarının alınması sürecinde 128 KB büyüklüğünde bir dosya kullanılmıştır. Bu dosya kodlanarak genişletilmiş ve kodlanmış hali hücresel ağ içerisindeki düğümlere eşit olarak dağıtılmıştır. Düğüm onarımları Δ zaman aralıklarında tembel onarım yöntemi kullanılarak gerçekleştirilmiştir. RS, MBR, MSR ve LDPC kodları için aynı kod oranları (k/n) gözetilmiştir. Sonuçlar daha sonra bir depolama düğümünün sakladığı veri miktarına göre $(\frac{F\,x\,n}{k\,x\,m})$ normalize de edilmiştir. Aynı zamanda sonuçlarımızı desteklemek adına teorik ortalama maliyetlerimizi (LDPC kodlar için üst sınır (upper bound) ve diğer silinti kodları için tam veriler) de dahil ettik.

Sonuçlarda kullanılan parametrelerde ise RS kodları ve $d = 23$ olacak şekilde MSR ve MBR kodları için $m = 24, n = 24, k = 12$ olarak alınmıştır. LDPC kodları için ayrıca $m = 24, n = 908, k = 454$ değerleri de kullanılmıştır. (ρ_{D2D}, ρ_{BS}) çifti için ise $(1, 1.2), (1, 1.12), (1, 1.17), (1, 1.26)$ değerleri kullanılmıştır.



Şekil 17: $1/2$ **kod oranı için onarım bant genişliği maliyeti (*repair bandwidth cost*).**

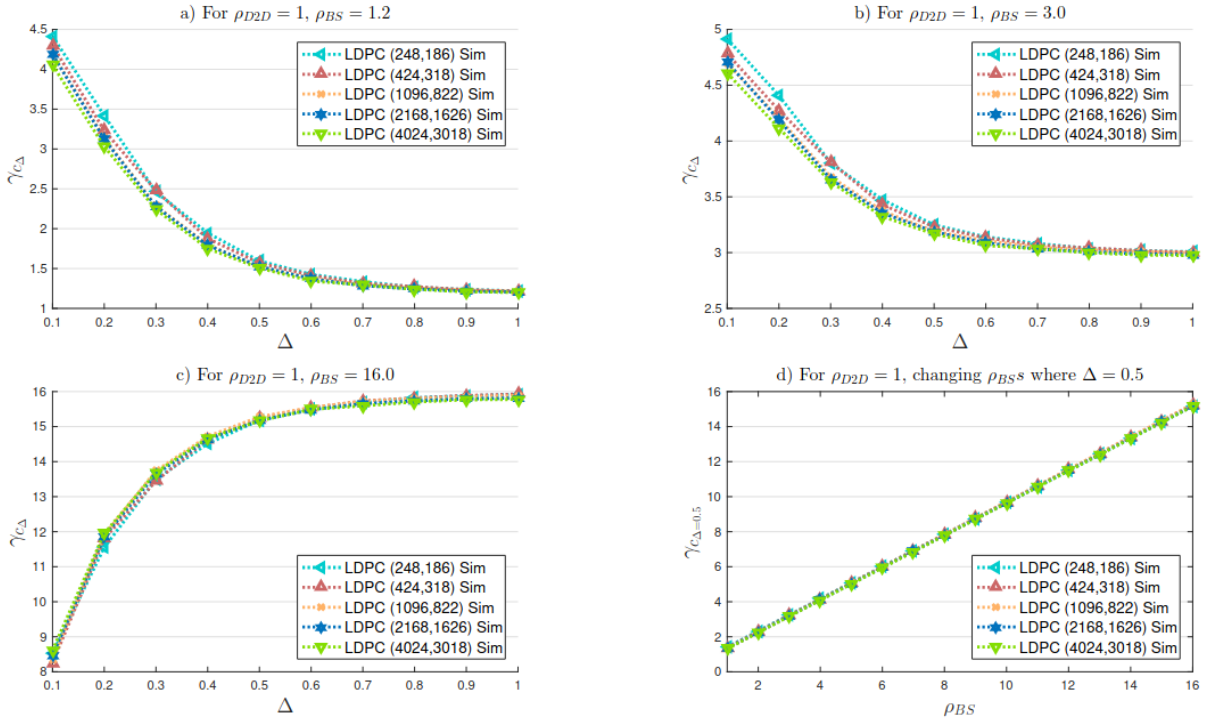
Şekil 17'de $1/2$ kod oranı için LDPC, RS, MBR, MSR silinti düzeltme kodlarına ait onarım bant genişliği maliyet sonuçları gösterilmektedir. (ρ_{D2D}, ρ_{BS}) çiftlerine göre 4 farklı karşılaştırma yapılmıştır. Bu sonuçlara göre $\rho_{BS} \leq 17$ ve $\Delta \leq 0.4$ iken, minimum maliyeti MBR kodları sağlamıştır. MBR kodlarını sırasıyla LDPC, MSR ve RS kodları takip etmiştir. $\rho_{BS} > 17$ iken RS kodları LDPC kodlarından daha iyi sonuçlar elde etmeye başlamıştır. $\rho_{BS} > 26$ olduğunda ise RS kodları bant genişliği maliyeti açısından MBR kodlarından daha iyi bir performans göstermiştir. Onarımlar esnasında yardımcı depolama düğümlerinden en çok sembol indiren RS kodları olmuştur. RS kodlarını MBR kodları izlemiştir. Ayrıca, LDPC, MSR ve MBR kodları, RS kodlarına kıyasla diğer depolama düğümlerinden sırasıyla %88, %88 ve %86 daha az sembol kullanır. Diğer yandan en yüksek Bİ kullanımı MSR kodlarının kullanımında gözlemlenmiştir. Minimum Bİ kullanımını ise RS kodları sağlamıştır. RS kodlarını MBR kodları takip etmiştir. Depolama alanı kullanımında MSR, RS ve LDPC kodları aynı depolama yüküne sahip iken MBR kodları en yüksek depolama alanı tüketimini göstermiştir. Ortalamada, RS kodları LDPC, MBR ve MSR kodlarından sırasıyla %61, %56 ve %66 daha az Bİ etkileşimine ihtiyaç duymuştur.



Şekil 18: $3/4$ **kod oranı için onarım bant genişliği maliyeti (repair bandwidth cost).**

Şekil 18'de MSR, MBR, RS ve LDPC kodlarını kullanan ve kod oranı 3/4 olan sistemlerin düğüm onarım maliyetleri gösterilmektedir. Sonuçlarda kullanılan parametrelerde RS kodları ve $d = 23$ olacak şekilde MBR kodları için $m = 24$, $n = 24$, $k = 18$ olarak alınmıştır. Simülasyonda aynı kod oranına sahip (3/4) silinti düzeltme kodları kullanılmıştır. Bu noktada, kullanılan MSR kodlarının 3/4 kod oranına, LDPC kodlarının da yine aynı şekilde 18/23 kod oranına tam olarak ulaşamadığını da belirtmek gerekir **(Wang vd., 2016)**. Bu yüzden MSR kodları için $m = 23$, $n = 23$, $k = 18$, $d = 22$ parametreleri kullanılmıştır. Ayrıca MSR kodlarının daha az depolama düğümüne sahip olması ve buna bağlı olarak daha az depolama yüküne (storage overhead) sahip olması nedeniyle MSR kodları $18/23$ kod oranına ölçeklendirilmiştir. LDPC kodları için $m = 24$, $n = 2056$, $k = 1542$ parametre değerleri kullanılmıştır. (ρ_{D2D}, ρ_{BS}) çifti için ise $(1, 3)$, $(1, 18)$, $(1, 24)$, $(1, 50)$ değerleri kullanılmıştır. $\rho_{BS} > 3$ iken MSR kodları, küçük Δ değerleri için LDPC kodlarından daha yüksek onarım maliyetli göstermiştir. $\rho_{BS} > 24$ iken RS kodları bazı Δ değerleri için LDPC ve MSR kodlarına göre daha az onarım maliyetine sahip olmuştur. $\Delta \geq 0.3$ iken MBR

kodlarının toplam Bİ iletişim miktarı RS kodlarından daha az olması nedeniyle RS kodları yüksek ρ_{BS} değerleri için bile MBR kodlarından daha iyi performans gösterememiştir. RS kodları, $\Delta < 0,6$ iken en yüksek D2D bant genişliğini talep etmiştir. $\Delta < 0,3$ olduğunda, diğer depolama düğümlerinden indirilen sembol sayısı bakımından RS kodlarını LDPC kodları ve LDPC kodlarını da MSR ve MBR kodları takip eder. $\Delta > 0,6$ için ise, MSR kodları diğer depolama düğümlerinden en az yardımı talep etmiştir. Bu sonuçlar, MBR kodlarının, en yüksek depolama yüküne sahip olmakla beraber, iletişim maliyeti açısından diğer kodlardan daha iyi performans gösterdiği gözlemlenmiştir. En yüksek Bİ iletişimi, $\Delta \leq 0,6$ için MSR kodlarıyla gerçekleşirken, $\Delta > 0,6$ için ise farklar çok da önemli değildir. MBR kodları, LDPC ve MSR kodlarına kıyasla sırasıyla %34 ve %35 daha az sembol indirmiştir. $\Delta > 0,2$ için MBR kodları, Bİ'den RS kodlarına göre %21 daha az sembol indirmiştir. En düşük Bİ iletişimi, 1/2 ve 3/4 kod oranları için RS ve MBR kodları kullanılarak elde edilmiştir. Diğer yandan, en düşük D2D iletişim $R = (3/4)^2$ için MSR ve LDPC kodları kullanıldığında elde edilmiştir.



Şekil 19: $3/4$ **kod oranı için farklı LDPC blok uzunluklarına göre onarım maliyeti.**

Şekil 19'da farklı blok uzunluklarına sahip LDPC kodlarının performansını karşılaştırmak amacıyla, 3/4 kod oranına göre beş farklı seçenek incelenmiştir. m=25 parametre değerine göre 25 depolama düğümü üzerinden (248, 186), (424, 318), (1096, 822), (2168, 1626) ve

(4024, 3018) LDPC kodları bu karşılaştırmada kullanılmıştır. (ρ_{D2D}, ρ_{BS}) parametre çifti için (1, 1.2), (1, 3), (1, 16) kullanılmıştır. Bu görsele göre pBS/pD2D oranının küçük değerleri için, küçük blok uzunluklarına sahip kodlar biraz daha yüksek maliyet göstermiştir. Ancak bu oran yükseldikçe farklı blok uzunluklarına sahip kodların maliyetleri birbirine çok yakın hale gelmiştir. Şekil 4-d'de Δ = 0.5 için $\gamma_{C\Delta}$, ρ_{BS}'nin bir fonksiyonu olarak sağlanmıştır. Δ arttığında, Bİ kullanılan onarımların sayısının artmaktadır çünkü bu durumda depolama düğümlerindeki yardımcı sembollerin kullanılabilirliği azalmaktadır. $|\bar{r}|$ kayıp bir sembolü onarmak için depolama düğümlerinden indirilen yardımcı sembollerin beklenen sayısıdır. $\rho_{BS}/\rho_{D2D} < |\bar{r}|$ ise onarım sürecindeki Bİ kullanımı, düğüm onarım bant genişliği maliyeti açısından avantajlıdır. Daha açık bir ifadeyle kayıp bir sembol Bİ aracılığıyla onarılırsa, yalnızca orijinal kayıp sembol indirilir. Ancak bu kayıp sembol yalnızca depolama düğümleri kullanılarak onarılırsa, kurtarma denklemindeki semboller indirilir. Sonuç olarak, ρ_{BS}/ρ_{D2D} $|\bar{r}|$'den küçük olduğunda, Şekil 14-a ve Şekil 14-b'de görülebileceği gibi, düğüm onarımlarındaki Bİ kullanımı artan Δ ile artar ve toplam düğüm onarım bant genişliği maliyeti Δ büyüdükçe azalır. Buna karşılık, ρ_{BS}/ρ_{D2D} $|\bar{r}|$'den yüksek olduğunda, Şekil 14-c'de görülebileceği gibi, artan Δ ile toplam düğüm onarım bant genişliği maliyeti artar.

Sonuç olarak bu çalışma kapsamında minimum maliyetle bir düğüm onarımı elde etmek için Greepair adlı yeni bir açgözlü algoritma önerilmiştir. Teorik ve simülasyon sonuçlarına göre, ρ_{BS} ve ρ_{D2D} arasındaki fark çok yüksek değilse, LDPC kodları düşük kodlama karmaşıklıklarının yanı sıra düşük depolama yükleri nedeniyle makul bir seçim olabilir.

4.4 Artık Veri ile İlgili Bulgular

Artık veri kavramı, depolama düğümlerinde düzensiz sayıda sembolün depolanmasıyla ilgilidir. Başka bir deyişle, kodlanmış (encoded) sembollerin depolama düğümlerine dağıtılması yöntemidir. Artık veri kullanımına dayalı bu sembol dağıtımı sayesinde, düğümlerde fazladan sembollerin depolanması önlenir. Bu dağıtım yönteminde onarım işlemlerinde zamandan tasarruf etmek yerine depolama alanından tasarruf edilmesi amaçlanmaktadır.

Artık verili sembol dağıtım yönteminin karşılaştırıldığı standart sembol dağıtım yönteminde ise kodlanan semboller depolama düğümlerine dağıtılırken amaç bu depolama düğümlerinin her birinde eşit sayıda sembol bulundurmaktadır. Böylece, sembol dağıtımından sonra her

depolama düğümünde eşit sayıda sembol depolanır. Bu amaçla gerekirse fazladan sahte (dummy) semboller saklanabilir.

Şekil 20'de yukarıda bahsedilen sembol dağıtım yöntemlerine dair bir örnek senaryo gösterilmektedir.

Symbol Index	1st Iteration	2nd Iteration	3rd Iteration
1	Node 1	Node 1	Node 1
2	Node 1	Node 1	Node 1
3	Node 1	Node 1	Node 1
4	Node 2	Node 2	Node 2
5	Node 2	Node 2	Node 2
6	Node 2	Node 2	Node 2
7	Node 3	Node 3	Node 3
8	Node 3	Node 3	Node 3
9	Node 3	Node 3	Node 3
10	Node 4	Node 4	Node 4
11	Node 4	Node 4	Node 4
12	Node 4	Node 4	Node 4

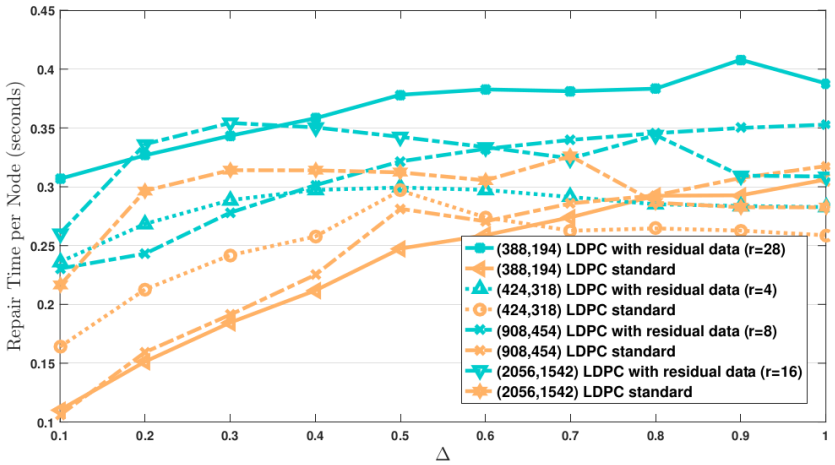
Symbol Index	1st Iteration	2nd Iteration	3rd Iteration
1	Node 1	Node 1	Node 1
2	Node 1	Node 1	Node 1
3	Node 2	Node 2	Node 2
4	Node 2	Node 2	Node 2
5	Node 3	Node 3	Node 3
6	Node 3	Node 3	Node 3
7	Node 4	Node 4	Node 4
8	Node 4	Node 4	Node 4
9	Node 1	Node 1	Node 1
10	Node 2	Node 2	Node 2
11	Node 3	Node 3	Node 3

Şekil 20: Standart sembol dağıtım yöntemi (Solda) ve Artık verili sembol dağıtım yöntemi (Sağda).

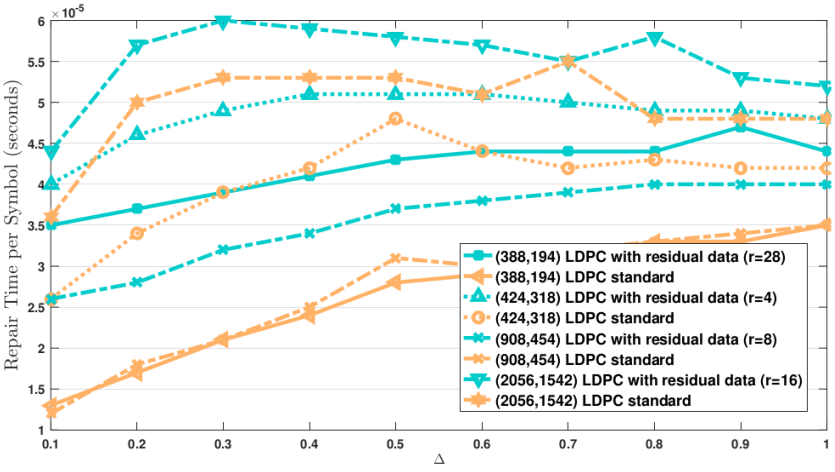
Bu senaryoda (Şekil 15), 11 kodlanmış sembol 4 depolama düğümüne dağıtılmak istenmektedir. Standart sembol dağıtım yönteminde her düğümde eşit sayıda kodlanmış sembol tutulmak istendiğinden, herhangi bir depolama düğümünde 3 sembol saklanır. Buna göre 12. sembol aslında kodlanmış bir sembol değildir ve istenilen eşitliği sağlamak için sıfır değeri verilir. 12. sembol indeksi ile ekstra depolama alanı kullanılmasına rağmen standart sembol dağıtım yönteminin temel motivasyonu onarım işlemlerinde zamandan tasarruf etmektir. Diğer yandan, artık verili sembol dağıtım örneğinde ise kodlanmış 11 sembolün ilk 8 sembolü çiftler halinde 4 depolama düğümüne dağıtılır. 9., 10. ve 11. sembol indeksleri artık verilerin sembol indeksleridir. 9. sembol indeksinden başlayarak, her bir sembol birinci depolama düğümünden sırayla bir düğüme gönderilir. Her döngüde aynı sembol indeksine karşılık gelen semboller, aynı depolama düğümünde saklanır. Önceki raporda bu aşamada artık veri sembol indeksleri sırasıyla her bir depolama düğümüne dağıtılırdı. Onarım zamanını iyileştirmek amacıyla böyle bir değişikliğe gidilmiştir. Bu kapsamda yapılan çalışmada, artık verilerin sembol indeksleri her döngüde her zaman aynı düğümlerle eşleştirilmiştir. Artık veri kullanımına dayalı bu dağıtım sayesinde, düğümlerde fazladan

sembollerin depolanması önlenir. Bu dağıtım yönteminde onarım işlemlerinde zamandan tasarruf etmek yerine depolama alanından tasarruf edilmesi amaçlanmaktadır.

Şekil 21'de standart ve artık verili sembol dağıtım yöntemleri sırasıyla (388,194), (424,318), (908,454) ve (2056,1542) LDPC kodları için bu İP kapsamında test edilmiştir. Bu yaklaşımlar için onarım süreleri aşağıdaki şekillerde görülebilir. Şekillerdeki yeşil çizgiler, artık veri dağıtımlı sembol dağıtım yöntemini kullanan kodları gösterirken, turuncu çizgiler standart sembol dağıtım yöntemini kullanan kodları temsil eder. Bu şekillerde r, her bir LDPC kodu için artık sembol indekslerinin sayısını ifade eder. Elde edilen sonuçlara göre, beklendiği gibi, artık veriye dayalı sembol dağıtım yöntemine sahip kodların onarım süreleri, standart dağıtım yöntemine dayalı kodların onarım sürelerine göre daha uzundur.

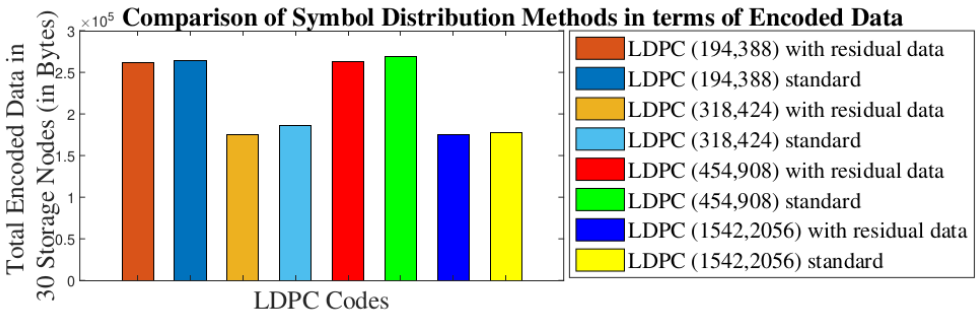


Şekil 21: Sembol dağıtım yöntemlerinin tek bir düğümün onarım süresi açısından karşılaştırılması.



Şekil 22: Sembol dağıtım yöntemlerinin tek bir sembolün onarım süresi açısından karşılaştırılması.

Şekil 23'te (388,194) LDPC kodu için, artık verili dağıtım yönteminin standart dağıtım yöntemine göre düğüm başına onarım aralıkları olan 0.1, 0.5 ve 0.9 Δ değerleri için sırasıyla %64.04, %34.57 ve %28.24 daha yüksek onarım sürelerine sahip olduğu görülmüştür.. (424,318) LDPC kodu için artık verilere dayalı dağıtım yöntemi, Δ değerlerinin sırasıyla 0.1, 0.5 ve 0.9 olduğu durumlarda standart dağıtım yöntemine göre düğüm başına %30,57, %0,86 ve %7,37 daha yüksek onarım sürelerine sahiptir. (908,454) LDPC kodu için artık verili dağıtım yöntemi, sırasıyla 0.1, 0.5 ve 0.9 Δ değerleri için standart dağıtım yöntemine göre düğüm başına %54.03, %12.53 ve %12.15 daha yüksek onarım sürelerine sahiptir. (2056,1542) LDPC kodu için ise artık verili dağıtım yöntemi, Δ



Şekil 23: 30 depolama düğümüne dağıtılan bayt cinsinden toplam sembol sayısı.

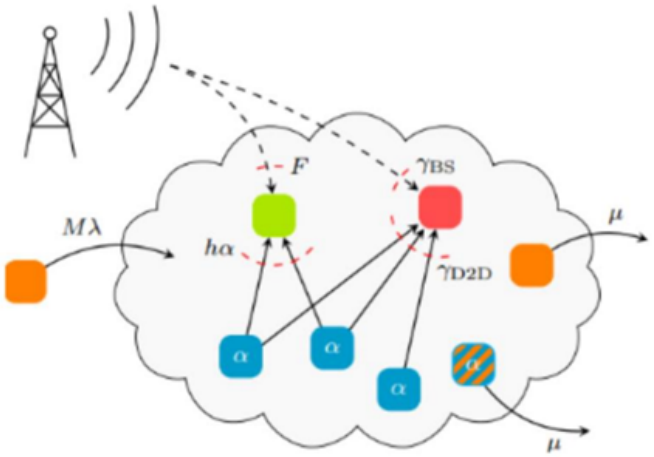
Diğer yandan yukarıdaki şekilde (Şekil 18) de görülebileceği gibi artık verili sembol dağıtım yöntemi saklanan kodlanmış veri miktarları açısından da karşılaştırmalara tabi tutulmuştur. Sonuçlardan da görüleceği üzere artık veri kullanımı (388,194) LDPC kodu için yaklaşık %0,51, (424,318) LDPC kodu için %5,78, (908,454) LDPC kodu için %2,37 ve LDPC kodu için %2,37 ve (2056,1542) LDPC kodu için %0,68 depolama alanı tasarrufu sağlamıştır. Artık sembol kullanımının depolama alanı açısından az da olsa tasarruf sağladığı gözlemlenmiştir.

4.5 Farklı Kuyruk Modelleri altında Performans Analizi ve Numerik Bulgular

4.5.1 Teorik altyapı

Bir adet baz istasyonu ve birçok düğümden oluşan bir sistemde her bir düğümün, baz istasyonunun kapsama alanına girme hızı λ ile ifade edilmektedir. Bu düğümlerin her birinin sistemden çıkma hızları ve sisteme girme hızları birbirine eşit kabul edilebilir. Sisteme girme

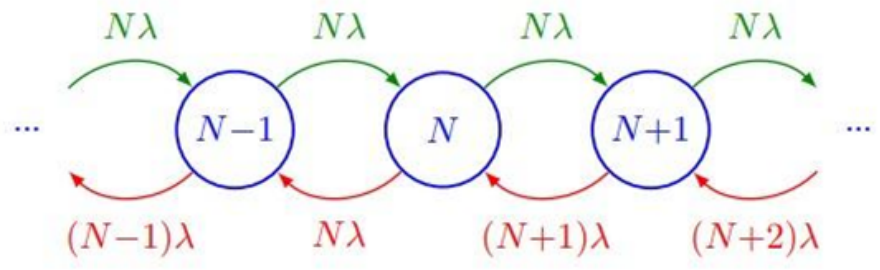
oranı ve çıkma oranı eşit olduğu zaman sistemde bulunan düğüm sayısı ortalama bir N değerinin etrafında değişkenlik gösterir. Sisteme toplam giriş hızı Nλ, toplam çıkış hızı da Nλ ile ifade edilebilir. Düğümlerin sisteme girişleri arasındaki süreler üstel (exponential) şekilde dağılım gösteren sistemlerin sayma süreçleri (count process) Poisson sürecine göre gerçekleşir. Bir düğümün sistem içerisinde geçirdiği beklenen süreye T, yani beklenen düğüm yaşam süresi denmektedir. Sistemde beklenen ortalama düğüm sayısı (popülasyon) ise N ile temsil edilmektedir.



Şekil 24: Simülasyonlarımızda kullandığımız düğümlerin sisteme giriş çıkışlarının sembolize edildiği sistem görseli (Paakkonen vd., 2013).

Fermat'ın "Little" yasasına göre sabit bir sistemdeki uzun vadeli ortalama müşteri sayısı L, uzun vadeli ortalama efektif varış oranı λ ile bir müşterinin sistemde geçirdiği ortalama sürenin W çarpımına eşit olduğu bilinmektedir, yani L=λW. Burada L yerine N, W yerine T yazarsak, λ, yani düğümlerin sisteme varış hızı N/T, art arda gelen iki varış arasında geçen süre ise T/N 'e eşit olur **(Paakkonen v.d., 2013)**. Şekil 24'de düğümlerin giriş-çıkışlarını sembolize eden sistem görseli verilmiştir.

Bu durumda düğüm sayısı anlık olarak değişmekte olup, beklenen düğüm sayısı *N*' nin etrafında değişecektir. Bu değişim Markov'un M/M/∞ kuyruk teorisine göre belirlenir **(Paakkonen vd., 2013)**. M/M/∞ kuyruk teorisindeki sonsuz, sistemdeki sunucu sayısını temsil etmektedir. Bu durumda düğümler tarafından gönderilen isteklerin (request) tamamı anlık olarak beklemeksizin karşılanabilmektedir. Yine aynı sistemdeki M'ler ise hem sistem girişleri hem de servis süreleri için üstel dağılımın kullanıldığını belirtmektedir.



Şekil 25: Yerel düğümlerin sayısı için M/M/1 Markov zincir durum diyagramı (Paakkonen vd., 2013).

Şekil 25' de Markov zincir durum görseli (state diagram) yer almaktadır. Mavi çemberler anlık düğüm sayısını temsil etmektedir. Düğümlerin gelme hızları Nλ olarak sabitken, çıkma hızları sistemde bulunan lokal düğüm sayısı ile doğru orantılıdır. Burada beklenen düğüm sayısı N ve λ=1/T 'dir. Sistem içerisinde baz istasyonlarının dosyaları tuttukları, düğümlerin baz istasyonuna dosya indirme ya da veri isteğinde bulunabildikleri varsayılmıştır. Aynı şekilde sistemdeki düğümler de kendi lokal depolama alanlarına sahiptirler. Böylece bir düğüm diğer bir düğümden dosya indirebilmektedir.

Düğümler sistemde kaldığı süre boyunca *w* oranı ile orantılı şekilde veri isteği yollarlar. Eğer mümkün ise dosya, D2D haberleşmesi kullanılarak depolama düğümlerinden indirilir. Ancak depolama düğümlerinin sayısı yeterli değilse, dosya baz istasyonundan indirilir. Baz istasyonu iletişimi maliyeti yüksek olduğundan, mümkün olduğu anlarda D2D haberleşmesi tercih edilmektedir Düğümlerin sonsuz depolama alanına sahip olduğu varsayılmaktadır Düğümler arasında veri iletmek, bir baz istasyonundan bir kullanıcıya veri iletmekten daha az maliyetlidir.

Sistemde, yani baz istasyonunun kapsama alanında bulunan düğümler sisteme girip çıkabilmekte olup, bu akış veri tutan düğüm sayısında bir eksilmeye sebep olabilir. Böyle bir durum gerçekleştiğinde ise kaybolan düğümün yerine yenisi getirilmelidir. Bu problemi çözmeye çalışan kuyruk sistemlerinin incelemesi ve de güç tüketimi (cost) açısından karşılaştırması bu araştırmanın asıl konusudur. Dağıtık sistemlerde güvenilir (doğru, değiştirilmemiş) veriye ulaşabilmek amacıyla "veri fazlalığı" (redundancy) dediğimiz fazlalık kısım veriye eklenir. Yani dağıtık sistemlerde reliability-redundancy arasında bir ödünleşim (tradeoff) bulunmaktadır.

Simüle edilecek sistemin ön kabulleri aşağıda belirtilmiştir:

(1) Sistemde bulunan düğümlerin sonsuz depolama kapasiteleri bulunmaktadır.
(2) Simülasyonda sadece tek bir dosya bulunduğu var sayılacaktır.
(3) Bu dosyanın tek bir düğüm tarafından istenme oranı w ile sembolize edilecektir.
(4) Dosya boyutu 1 olacak şekilde tüm maliyetler normalize edilmiştir.
(5) Lokal bir düğümden karşılanan bir isteğin maliyeti 1 joule olarak kabul edilmiştir.
(6) Baz istasyonundan karşılanan bir isteğin maliyeti R' dir. (R>1 olduğu kabul edilecektir.)

Karşılaştırması yapılacak olan caching metodları Basit önbellekleme (Simple Caching) ve fazlalıklı önbellekleme (Redundant Caching) olmak üzere ikiye ayrılır:

Simple Caching: İstenen dosyanın sistemde bulunan bir lokal düğüme parçalanmadan veya kodlanmadan tamamen kopyasının kaydedilmesi ve sonraki isteklerin de artık bu lokal (caching) düğüm üzerinden karşılanması bir basit önbellekleme uygulamasıdır. Ancak istenen dosya sistemde bulunan herhangi bir lokal düğümde bulunmuyorsa bu istek artık baz istasyonundan karşılanır. İsteği yapan düğüm ise dosyayı ön belleğine kaydederek sonraki istekleri karşılayabilir.

Redundant Caching: Birden fazla (fazlalıklı) lokal düğümün dosyanın kopyasını ya da bir parçasını sakladığı caching metodudur. Bu durumda bir istek geldiğinde dosyanın parçaları isteği yapan düğümde birleştirilir. Fazlalıklı önbellekleme metodunun en basit ve efektif uygulaması 2-replication olarak adlandırılan kopyalama yöntemidir. Bu yöntemde simple caching'den farklı olarak dosyanın iki adet kopyası bulundurmaya çalışılır.Daha önceki çalışmalarda bulunduğu üzere (Paakkonen vd., 2013), simple caching metodunun teorik iletim maliyeti aşağıdaki denklemle hesaplanabilir:

$$C_{\text{sc}}(R,N,\omega,T)=\frac{N\omega T+R}{T+\frac{1}{N\omega}}=\frac{N^2\omega^2T+RN\omega}{1+N\omega T}$$

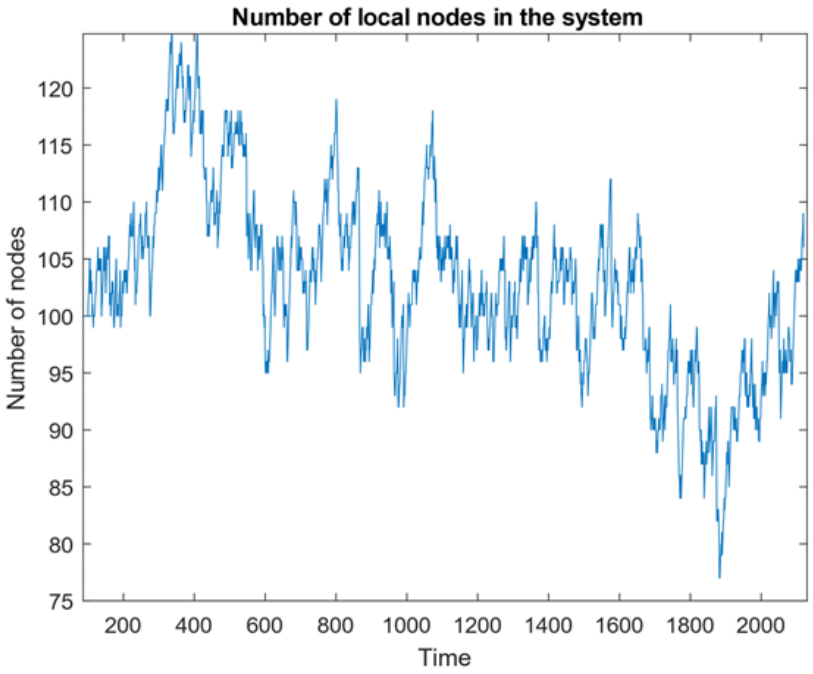
2-replication için kullanılacak olan teorik iletim maliyeti aşağıdaki denklemle hesaplanmaktadır:

$$C_{\text{2-rep}}(N,\omega,T)=N\omega+\frac{2}{T}.$$

4.5.1 Simulasyon bulguları

Simülasyon ortamının oluşturulmasında **Matlab simülasyon** ortamları kullanılmıştır. İlk olarak bir N sayısı etrafında değişkenli k gösteren düğüm sayısına sahip bir sistem

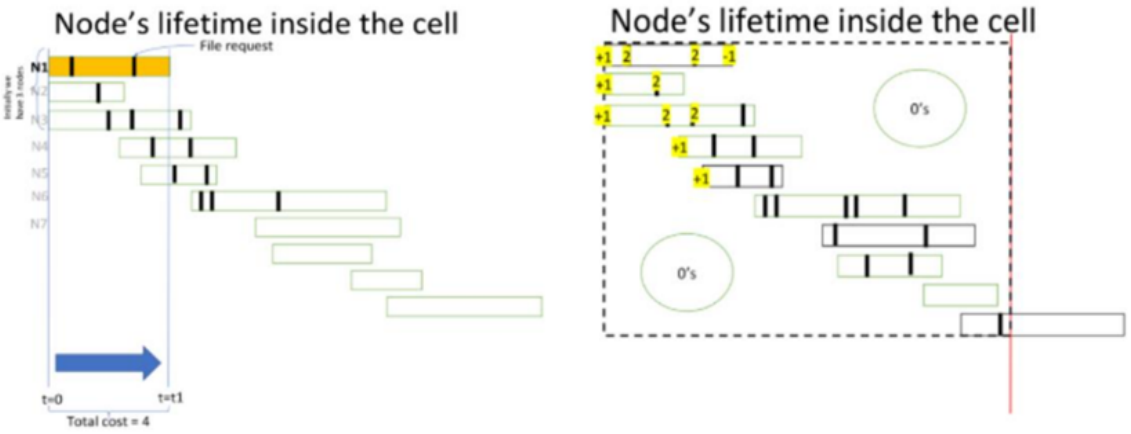
oluşturulmuştur. Düğümler arası sisteme ulaşma zamanı ve sistemden ayrılma zamanları öncelikli olarak Poisson dağılımına göre düzenlenmiştir.



Şekil 26: Düğüm sayısının zamana göre değişim grafiği (N=100).

Şekil 26'daki grafikte düğüm sayısının zamana göre değişimi N=100 değeri için gösterilmektedir. Görüldüğü üzere gerçek sayı bu değer etrafında gezinerek ortalamayı tutturmaktadır.

Sonraki adımda düğümlerin w oranındaki dosya indirme istekleri Poisson dağılımı ile belirlenen zaman adımlarına yerleştirilmiştir. Sonraki adımda ise simple caching algoritmasının implementasyonu için her bir düğümün bireysel olarak takibinin sağlanması gerektiği anlaşılmıştır. Bunun sebebi bir düğüm dosya indirme isteği gönderdiği durumda aynı zamanda o dosyayı ön belleğinde bulundurabilmesidir. Bu durumda dosya indirme maliyeti olmayacaktır. Bu gibi durumların takibi de ancak sisteme giren her bir dosyanın indekslenerek bir matrise kaydedilmesiyle mümkündür.



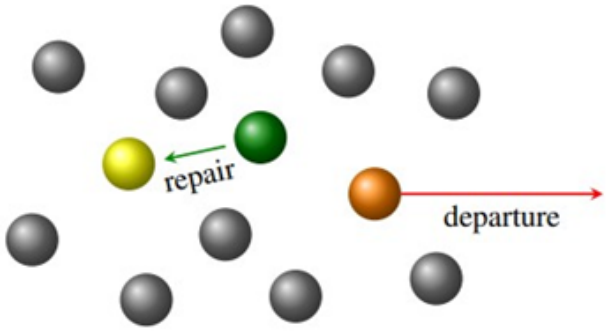
Şekil 27: Önbellekleme algoritmalarının implementasyonu için kurgulanmış bir senaryonun görseli.

Şekil 27'de verilen şemada önbellekleme algoritmalarının gerçeklenmesini sağlayan mantık bir örnek anlatımı ile kısaca özetlenmiştir. İki boyutlu bir matrisin satırları sisteme giren düğümlerin indeksini, sütunları ise zaman adımlarını temsil etmektedir. Bir düğüm sisteme girdiği zaman adımında matrise 1, sistemden çıktığında -1, dosya indirme isteği gönderdiğinde ise 2 yazılmaktadır. Simülasyon başlangıcında (simple caching için) dosya kaydeden düğümün 1. indeksteki düğüm olduğu varsayılmaktadır. Dikdörtgenler düğümlerin yaşam süresini ifade etmektedir. Yaşam süresi içerisinde bulunan siyah çizgiler ise dosya istemlerini (request) temsil etmektedir. Bir zaman adımı içerisinde harcanan enerji maliyetinin hesaplanması, dosyayı kaydeden düğümün dışında bulunan tüm düğümlerdeki dosya indirme istekleri, toplam enerji maliyetine eklenmektedir. Yine aynı şekilde sağdaki resimde ise düğümlerin sisteme varışı, dosya indirme istekleri ve sistemden ayrılışlarının matriste nasıl temsil edildiği gösterilmektedir. Bu matrisin oluşturulması, düğümlerin takibinin sağlanmasını kolaylaştırmaktadır. Bu matris, oldukça büyük boyutlara sahip olacağından dolayı bellekten tasarruf etmek amacıyla ayrık (sparse) yapılı matrisler olarak tanımlanmakta ve bellekte tutulmaktadır. Daha açık şekilde ifade edecek olursak, sparse matris, sıfır olmayan elemanları indeksleri ile birlikte hafızasında tutmasıyla bellekten tasarruf etmektedir.

Simülasyon maliyetinin hesaplanmasında ise aşağıdaki denklem kullanılmıştır:

(1) Sistemdeki dosya indirme isteklerinin toplam sayısı = A
(2) Baz istasyonuna gitme sayısı = B
(3) Toplam enerji maliyeti => $A + BR$

Redundancy caching algoritmasının gerçeklenmesi de yine Simple Caching algoritması için anlatılan temellere dayanmaktadır. 2-replication metodunda dosyanın kopyasını tutan düğüm sayısı 2'nin altına düşerse, gelecek bir sonraki istek kalan diğer düğümden; düğüm sayısı 0'a düşerse de baz istasyonundan karşılanmaktadır.

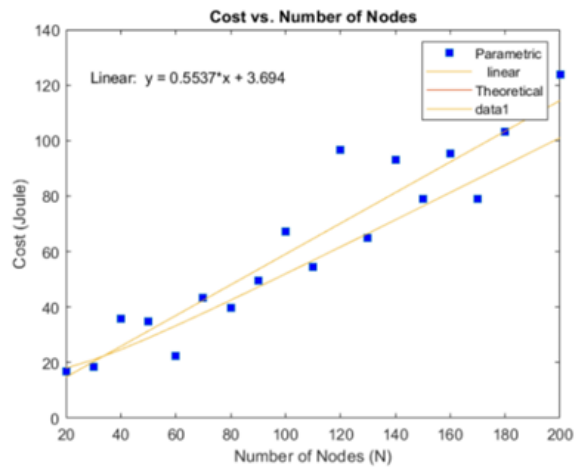
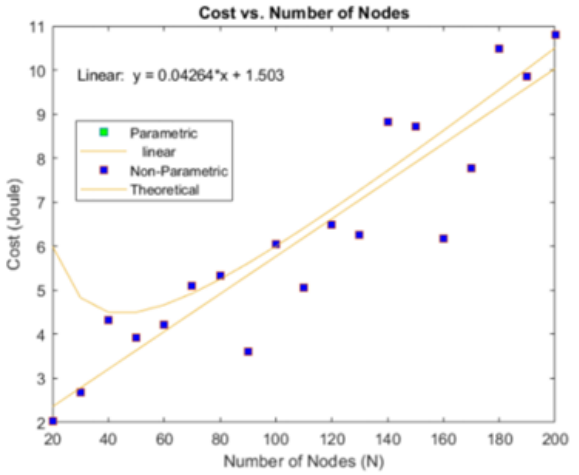
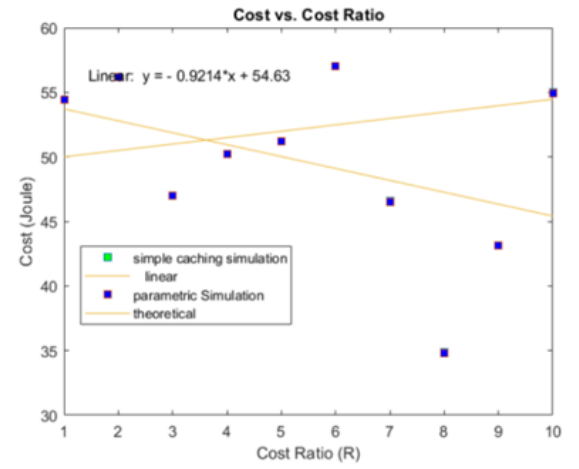
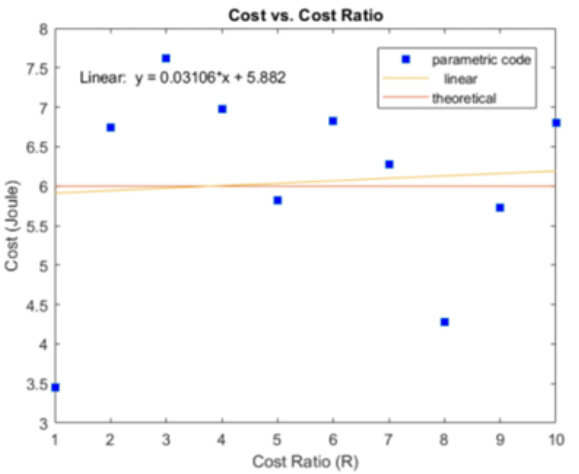


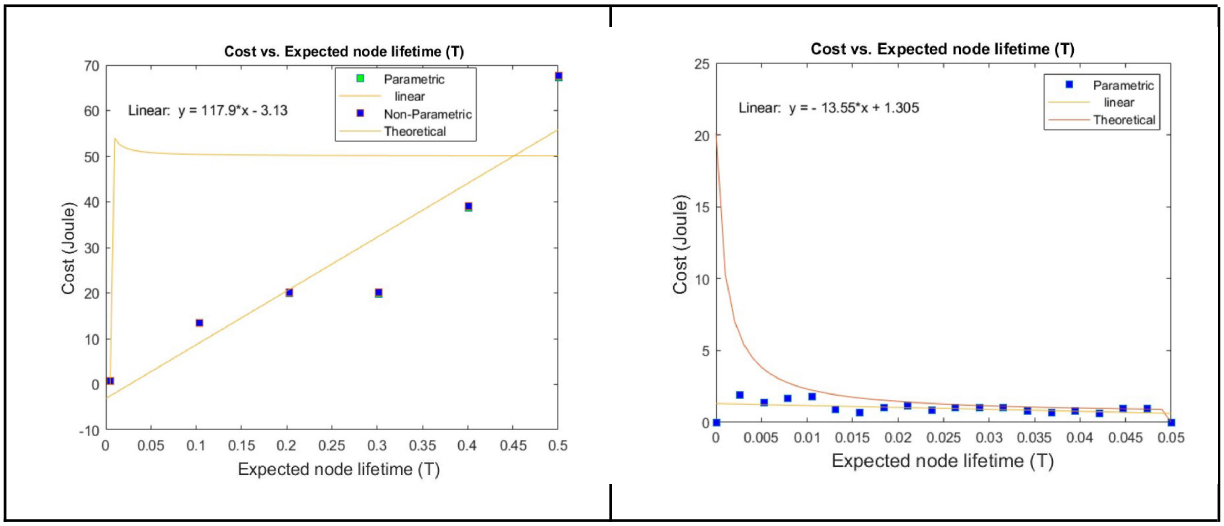
Şekil 28: 2-replikasyon onarımı senaryosu (Paakkonen vd., 2013).

Şekil 28'de görüldüğü üzere sistemde bulunan yeşil ve kahverengi düğümler dosyanın birer kopyasını bulundurmaktadırlar. Kahverengi düğüm sistemden ayrılırsa, bu ayrılmadan sonra gelen istek yeşil düğümden karşılanacaktır. Bu isteği gönderen sarı düğüm de yeni dosya tutan düğüm olmuş olur. Böylece istemde bulunan düğüm aslında aynı zamanda verinin ikinci önbelleklemesi için de sorumluluk almış olacaktır. Sonuçlarımız, simple ve redundant caching karşılaştırması özelinde, iletim maliyeti üzerinden yapılacaktır. Simple caching metodunda simülasyonun başlangıcında sistemde bir adet dosya tutan (caching) düğüm olduğu varsayılacaktır. Redundant caching de ise kaç kopya tutuluyor ise başlangıçta o adette kopyanın olduğu düşünülmektedir.

Simple caching ve redundancy caching algoritmaları için elde edilen simülasyon sonuçlarımızı Şekil 29' de toplam altı grafik aracılığı ile raporlamaktayız. İlk simülasyonlar genel Poisson sayma (üstel düğüm giriş çıkış zamanları) süreçleri ile üretilmiştir. İlk sırada masrafın (cost-joule cinsinden) sistem içerisindeki ortalama toplam düğüm sayısı cinsinden fonksiyonel olarak değişimi her iki caching için gösterilmiştir. Redundant caching için basitlik açısından 2-kopya olarak düşünülmüştür. Daha iyi görsellik açısından teorik sonuçlar üzerine lineer regresyon sonuçları da eklenmiştir. İkinci sırada masrafın (cost-joule cinsinden) baz istasyonuna ulaşımda maruz kalınan ekstra masraf (R) cinsinden ilişkisi analiz edilmiş ve görselleştirişmiştir. Görüleceği üzere R arttıkça masrafta artmaktadır. Son olarak son sırada masrafın (cost-joule cinsinden) bir düğümün hücresel sistemden ayrılmadan evvel geçirdiği zamana göre değişimi gösterilmiştir. Görüleceği üzere kalınan zaman arttıkça masraf da

artmaktadır. İlginç olan 2-kopyalı durumda ise bu masraf gözle görülür herhangi bir şekilde artış gözlemlenmemiştir.

Simple caching Algoritması	Redundancy caching Algoritması
	
	



Şekil 29: Masrafın (cost-joule cinsinden) sistem içerisindeki ortalama toplam düğüm sayısı, R ve ortalama düğüm hayatı cinsinden değişimi gösterilmiştir. İki türlü gerçekleme gerçekleştirmiş durumdayız:Non-Parametric: Bu gerçeklemede sistem sadece bir (ya da iki) kopya tutmak üzere sabit şekilde yazılmış, Parametric: Bu gerçeklemede ise kopya sayısı parametrik olarak belirlenmiş ve dolayısı ile birden fazla kopya tutabilmeye izin verilmiştir. Görüleceği üzere bu iki gerçeklemenin tek ve iki kopyalı senaryolarda birbirinin aynısı sonuçlar çıkardığı görülmüştür.

Proje önerimizde ve bu iş paketi kapsamında belirtildiği üzere, farklı kuyruk modelleri ile bu sonuçların nasıl değişeceğini araştırmış durumdayız. Burada yaklaşımımız düğümlerin giriş ve çıkış zamanlarını *üstel* dağılımdan *Weibull* dağılımı ile değiştirerek, dolaylı yoldan sayma (count process) sürecini de değiştirmek olmuştur. Böylece M/M/1 gibi (Poisson sayme süreci) bilinen ve literatürde sıkça kullanılan bir sistemden G/G/1 veya G/G/n kuyruk modellerine simülasyon ortamımızı dönüştürdük. Buradaki genel dağılımı ifade eden "G" için spesifik bir dağılımlar varsaydık. Spesifik dağılımlar seçmemizin iki temel nedeni şu şekilde anlatılabilir: (1) Weibull dağılımının birçok bulut ve hücresel sistemlerde toplanan veri için modelleme sağlayabildiği daha önceki çalışmalarda gösterilmiş durumdadır **(Satyanarayanan, 1981)**, (2) Weibull dağılımı üstel fonksiyonun genelleştirilmiş halidir. Dağılımı hatırlayacak olursak,

$$f(x;\lambda,k) = \left\{ \frac{k}{\lambda}\left(\frac{k}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k},\ x \geq 0 \right\}$$

$k > 0$ "shape" parametresi, $\lambda > 0$ "scale" parametresi olma üzere şeklinde ifade edilir. Görüleceği üzere "shape" parametresinin 1'e eşit olduğu durumda ($k = 1$), "scale" parametresi ile oynanarak bir üstel fonksiyon elde edilip, bu fonksiyonun tek parametresi olan "oranı" değiştirilebilir.

(McShane vd., 2008) çalışmasında da görüleceği üzere zaman dağılımı Weibull olan sayma süreçlerinin kapalı formda yazılması oldukça zor ve (kapalı formun aksine) toplamlar halinde

ifade edilmektedir. Bu da sistemin içerisinde bulunan düğümler ile ilgili performans ölçülerinin teorik/analitik olarak hesabını neredeyse imkansız hale getirmektedir. Bu nedenle simülatör ortamlarının oluşturulması ile, masraf performansının düğüm sayısı, R ve düğüm hayatı ile nasıl değiştiğini bulma fırsatımız olmuştur. Sonuçlarımız simple caching ve redundancy caching için ayrı ayrı gösterilmiştir. Bu sonuçlardan öğrendiklerimizi şu şekilde özetleyebiliriz:

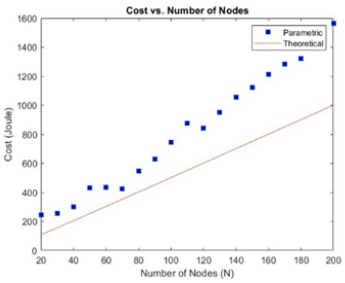
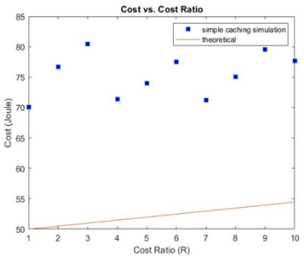
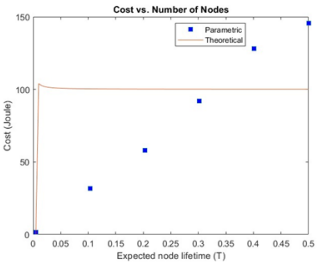
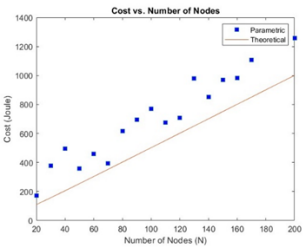
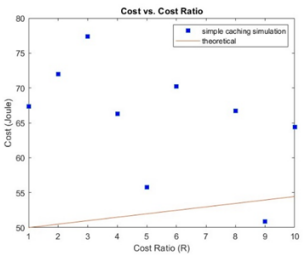
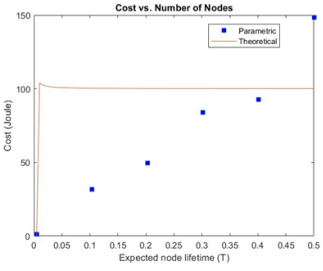
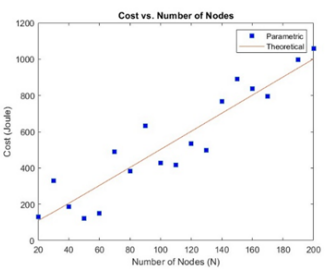
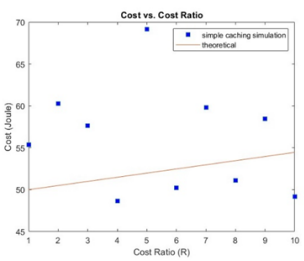
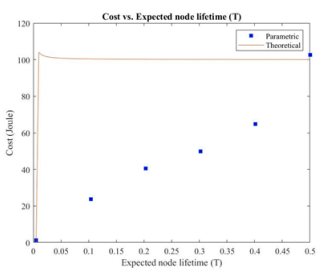
(1) Shape parametresi arttıkça masraf konvansiyonel üstel dağılımlı sisteme göre artış göstermektedir.

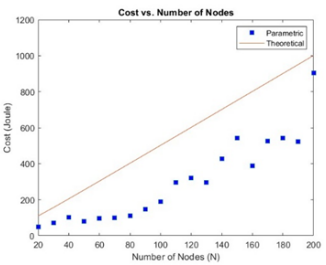
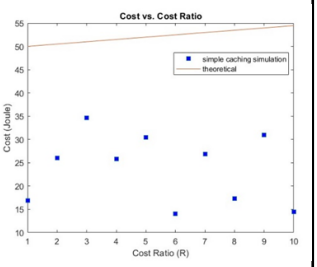
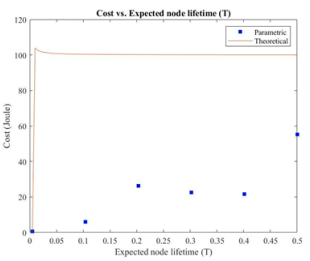
(2) Bu artış düğüm sayısı ve R parametresi ile doğru orantılıdır.

(3) hücre içerisinde kaldıkça üretilen masraf yine shape parametresinin artışı ile artış göstermektedir.

Bu sonuçlar aslında beklenildiği gibidir lakin shape parametresinin artması ile beraber dağılım daha Gauss tipi olacak ve hücreye giren düğümler eğer hemen ayrılmıyorlar ise belli bir süre düğüm içerisinde zaman geçirecekler demektir. Bu da dolayısı ile masrafı artıracaktır.

Simple Caching

Shape=5			
Shape=2			
Shape=1			

Shape=0.5			

Şekil 30: Simple caching Weibull dağılımı (farklı shape parametreleri için) ile düğüm giriş-çıkışı olduğundaki performansı.

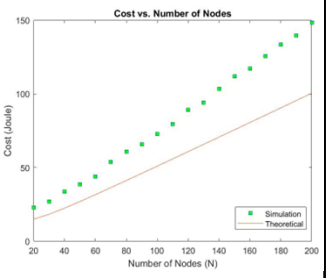
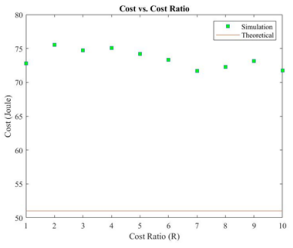
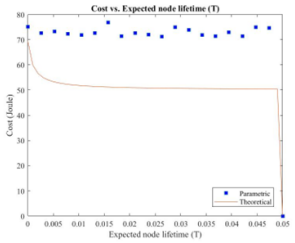
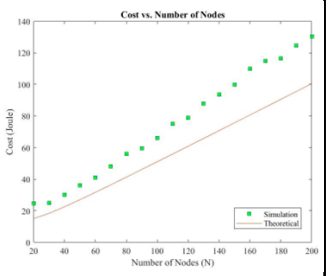
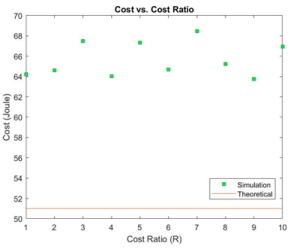
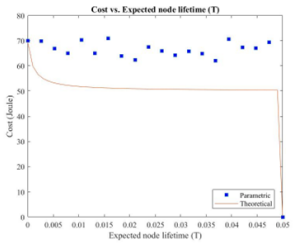
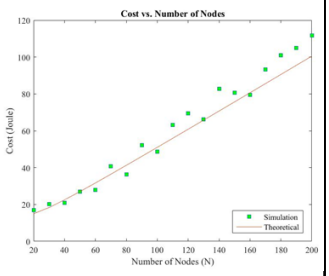
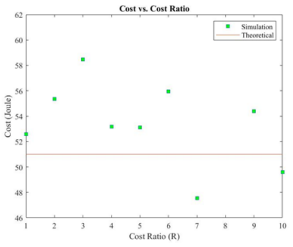
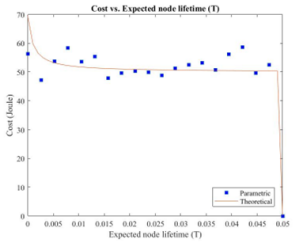
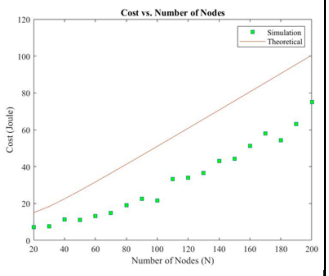
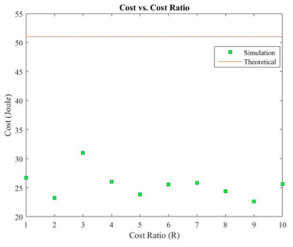
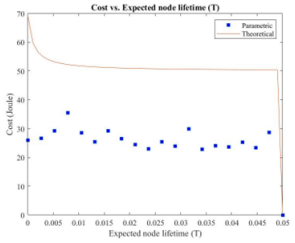
Bu sistemimizin yanında redundant caching yapan (2-copy) sistem içinde benzer simülasyonlar çalıştırılmış ve aşağıdaki sonuçlar elde edilmiştir. Yine sonuçlarımızı şu şekilde özetleyebiliriz:

(1) Benzer bir sonuç burada da gözlemlenmiştir: Shape parametresi arttıkça masraf konvansiyonel üstel dağılımlı sisteme göre artış göstermektedir. Bu artış düğüm sayısı ve R parametresi ile doğru orantılıdır. hücre içerisinde kaldıkça üretilen masraf yine shape parametresinin artışı ile artış göstermektedir.

(2) Bunlara ek olarak elde ettiğimiz verinin değişkenliği (varyansı) daha düşüktür ve dolayısı ile daha düzgün bir dizilim sergilemiştir.

(3) Masrafların simple caching'e göre gözlemlenebilir şekilde düştüğü sonucuna varılmıştır.

Varyansın düşük olmasının bir nedeni daha fazla kopyanın bulunması ile verideki değişkenliğin azaltılması başarılmış olmasıdır. Bunun yanında Redundant caching önceki Poisson süreçli sistemleri düşünen çalışmalarda bulunduğu üzere Weibull dağılımlı giriş-çıkış zamanlı bir sistem içinde masrafı oldukça düşürmektedir. Simple caching kullanarak Tek-kopyalı üstel sistem için elde edilen düğüm hayatı ve masraf arasındaki ilişki iki-kopyalı redundant caching kullanarak shape=0.5 için elde edilmiştir. **Dolayısı ile bu çalışmamızla kopya sayısı artırılarak üretilen ekstra masraf, protokol ve giriş-çıkış süreleri değiştirilerek azaltılabileceği, dolayısı ile sistemin masraf performansını belli bir seviyede tutarak birden fazla kopya tutulması ve dolayısı ile daha güvenilir ve uygun (reliable and available) sistem tasarımları yapabilir olduğunu göstermiş durumdayız.** Sonuçlarımızı lokal bir konferans aracılığı ile literatür ile paylaşmayı planlamaktayız.

Shape=5			
Shape=2			
Shape=1			
Shape=0.5			

Şekil 31: Simple caching Weibull dağılımı (farklı shape parametreleri için) ile düğüm giriş-çıkışı olduğundaki performansı.

4.6 ns-3 ile İlgili Bulgular

Bu başlık ns-3 tabanlı greepair düğüm tamiri yazılımı farklı Δ değerlerine göre tembel tamir etme işleminden geçirilmiş ve elde edilen bulgular Tablo 6'da listelenmiştir. (n=424,k=318) LDPC kodlar kullanılmış olup; hücre içerisinde Elde edilen bulgular arasında; çoklu ya da tekli düğüm tamiri süresince tamir edilen bir sembol başına UE düğümler ile gerçekleştirilen iletişimin sembol cinsinden miktarı, baz istasyonu ile gerçekleştirilen iletişim miktarı, ayrıca

düğümler ile gerçekleştirilen iletişimin verimi, benzer şekilde UE düğümler ile gerçekleştirilen iletişimin verimi ve son olarak yerel benzetim ile elde edemediğimiz bu iletişimlerin sırasıyla ne kadar sürdüğü belirlenmiştir.

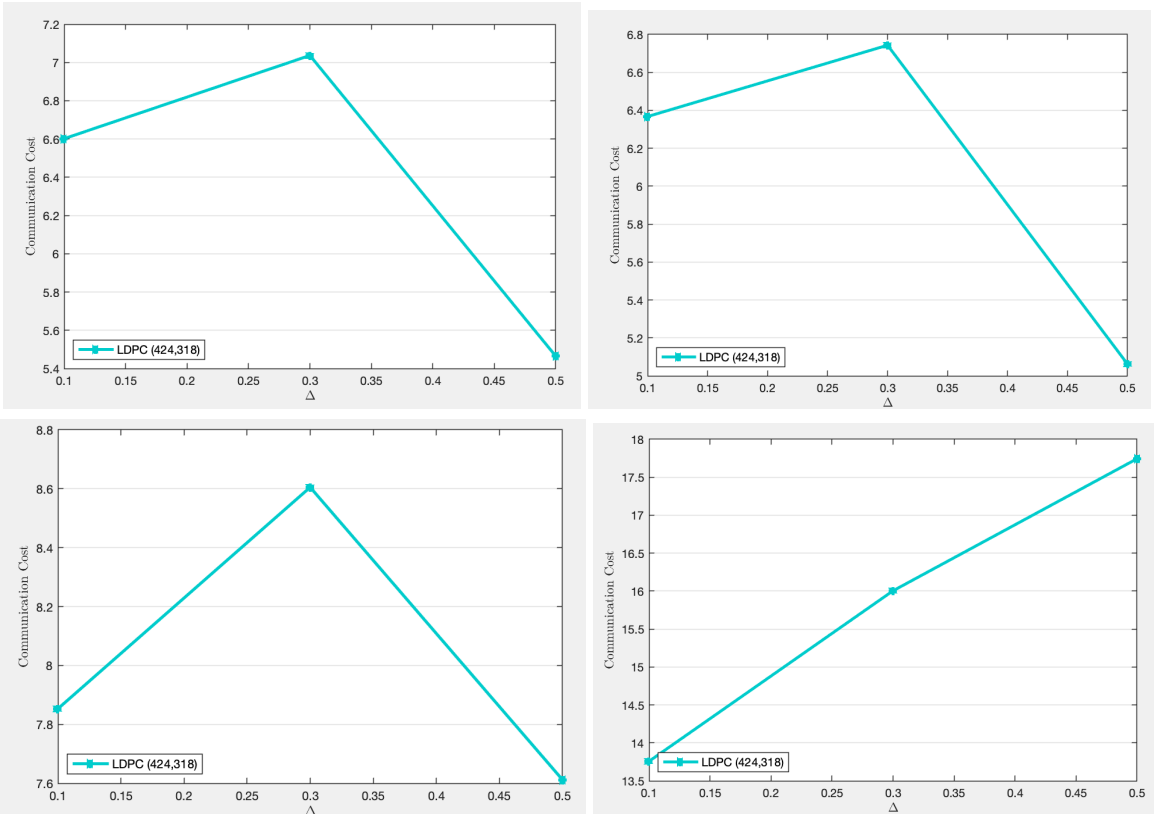
Tablo 6: Ağ benzetimi sonucunda Greepair düğüm tamiri için elde edilen veriler.

Δ	Düğümler aracılığıyla harcanan bant genişliği	Baz istasyonu aracılığıyla harcanan bant genişliği	Düğüm iletişimi verimi (Kbps)	Baz İstasyonu iletişimi verimi (Kbps)	Düğüm iletişimi Süresi (sn)	Baz istasyonu iletişim Süresi (sn)
ns-3 Ağ benzetim ortamın						
0.1	5.898	0.391	29.49843	31.59399	0.0028	0.0221
0.3	6.155	0.490	28.94078	41.77717	0.0029	0.02
0.5	4.257	0.671	29.05770	43.12532	0.0029	0.022
Yerel benzetim ortamında						
0.1	5.649	0.308	-	-	-	-
0.3	5.146	0.465	-	-	-	-
0.5	3.928	0.603	-	-	-	-

Tablo 3'te görüldüğü gibi tek bir sembolün onarımı için indilien sembol sayısı ağ benzetim ortamındaki daha yüksek çıkmakta ama sonuçlar arasındaki farklılıklar D2D (düğümler arası iletişimde) için Δ=0.1, 0.3 ve 0.5 için sırası ile 0.0441, 0.1961 ve 0.0838 kat daha fazla iken baz istasyonu iletişimi yerel ortamdan 0.2695, 0.0538 ve 0.1128 kat daha fazla olmaktadır. Burada düğümler arası ve baz istasyonu haberleşmesinde yerel benzetimlerde hesaba katılmayan paket başlıkları (headerlar) ve kablosuz haberleşmeden kaynaklanan paket kayıpları nedeni ile yeniden gönderimler ağ benzetim ortamında hesaba katılmıştır.

Sonuç olarak baz istasyonu ile gerçekleştirilen iletişimde D2D'ye göre daha çok yeniden gönderme yaşandığı gözlemlenmiştir.

Aşağıdaki Şekil 32'de farklı ρ_{D2D}/ρ_{BS} oranları için kayıp bir sembolün tamiri için harcanan iletişim maliyeti verilmiştir. Görüldüğü üzere ρ_{D2D}/ρ_{BS} oranı düşük değerlere sahipken yani baz istasyonu ile iletişim çok yüksek değilse baz istasyonu kullanımı daha düşük bir iletişim maliyeti oluştururken; ρ_{D2D}/ρ_{BS} değeri 5'ten büyük olduğunda iletişim ile ilgili maliyet eğrisi değişmekte ve Δ = 0.3 de tepe noktasını oluşturuken, son şekilde olduğu gibi daha yüksek ρ_{D2D}/ρ_{BS} değerleri için eğri bu parametreye göre lineer bir artış göstermekte, D2D iletişimin bu anlarda daha avantajlı olduğu görülmektedir.



Şekil 32: ρ_{D2D}/ρ_{BS} değeri sırası ile 1.2, 1.8, 5 ve 20.1 için ağ benzetimi ile elde edilen düğüm tamiri iletişim maliyeti.

5. Tartışma

Proje çalışması boyunca temelde dağıtık depolama ve dağıtık önbellekleme ile ilgili konular değişik açılardan; hem teorik hem pratik açıdan incelenmiştir. Bu bağlamda teorik olarak yapılan katkıları özetlemek gerekirse; çok katmanlı yani yerel düğümlerin yanında bazi

istasyonu, uydu gibi iletişim maliyetleri daha farklı olan sistemlerde; bant-genişliği açısından minimum maliyete sahip kodlar teorik olarak tasarlanmıştır.

Diğer taraftan, çok katmanlı sistemlere sahip bir yapıda iletişim maliyetini tahmin etmek için seçilen veri kaybı kaynağının nasıl etkileyeceği de incelenmiştir. Bu çalışma sonuçları daha genel bir çatı altında genişletilerek "tahmin maliyeti" (cost of guessing) olarak adlandırılan bir kavram geliştirilmiştir. Bu teorik altyapı ve bu kavramın ağ sistemlerinde veri onarımına nasıl uygulanabileceği, literatürde bir ilk niteliği taşımaktadır. Bu çalışma, çok katmanlı sistemlerin ve veri onarımının iletişim maliyeti üzerindeki etkisini anlamak için önemli bir adımdır ve ilgili alanda yeni bir bakış açısı sunmaktadır.

Ayrıca, seyrek çizge kodlarından LDPC kodlarının dağıtık depolama sistemleri için tasarlanmasında minimum düğüm tamir maliyeti ve maksimum geriçatım olasılığının sağlayacak kodların değişken ve kontrol düğümü derece dağılımları için pareto front elde edilmesini sağlayacak evrimsel algoritma tasarlanmıştır. Elde edilen pareto front noktalarını sağlayan dağılımlar ile elde edilen düğüm tamir maliyeti ve geriçatım olasılığı değerleri hem teorik hesaplamalarla hem de benzetim yolu ile alınarak; literatürde öne sürülenin aksine eğer minimum dereceye sahip kontrol düğümlerini seçme stratejisine sahip olunabilirse **düzenli kodlar değil de düzensiz LDPC kodlarının optimal sonuçları elde edebileceği gösterilmiştir.** Bu sonuç, LDPC kodlarının tarihsel olarak sadece veri geri çatımı açısından incelenmiş olduğunu ancak daha geniş tasarımların mümkün olacağını göstermiş ve nihayetinde bunların temel bilimsel limitlerini ortaya koyması açısından büyük önem taşımaktadır. **Bu sonuçların 6G ve ötesi sistemlerde kullanılacak olan kodların seçiminde büyük önem teşkil edeceğini düşünmekteyiz.**

Proje kapsamında daha pratik katkıları bulunan çalışmaları özetleyecek olursak; çizge kodlarının dağıtık veri önbellekleme kapsamında kullanılması literatüre önerilmiş olup bununla ilgili gerekli algoritmalar sunularak bu algoritmaların sonuçlarının performans katkıları yenileme kodları ve RS kodları ile kıyaslanarak analiz edilmiştir. **Bu karşılaştırmalı çalışma literatürde LDPC kodlarının veri önbellekleme konusundaki performansı hakkında incelenen ilk çalışma olması açısından önem taşımaktadır.** Ayrıca bu kodların dağıtık veri ön-bellekleme için kullanıldığı takdirde düğümlerin içerisine verilerin farklı bir yöntem ile dağıtılması, depolama alanının daha verimli kullanılmasını sağlayacak bir yöntem geliştirilmiş olup bu yöntemin klasik depolama yöntemleri ile kıyaslanarak depolama kullanımı açısından getirdiği iyileştirmenin yanında düğüm tamiri işleminin süresini artırdığı gösterilmiştir.

Projenin başka bir önemli katkısı, farklı kuyruk modellerinin ağ ortamlarında düğümlerin hücresel ağa bağlanma ve ayrılma süreçlerini modellemesidir. Bu çalışmada, literatürdeki genel yaklaşımdan farklı olarak, ayrılış ve bağlanış zamanlarının Weibull dağılımına tabi olduğu, daha gerçekçi sayma süreçlerinin düşünüldüğü görülmüştür. Ayrıca, düğümlerin sınırlı kapasitelerine de dikkat edilmiştir. Bu bağlamda, analitik sonuçların elde edilmesi zor olduğundan, simülasyon ortamları oluşturulmuş ve sayısal veriler elde edilmiştir. Bu sayısal veriler, **yorumlanabilir sonuçlar** sunmuştur.

Son olarak, çizge kodlarından LDPC kodları ile ilgili önerilen algoritma ns-3 ağ benzetim ortamında gerçekleştirilmiştir. Gerçeklemelerimiz sadece kod tarafında değil aynı zamanda gerçekçi sistem topolojileri ve kablosuz sistem tasarımları açısından da yenilikler içermiştir. **Böylelikle benzetim ortamında alınan değerlerin gerçeğe yakın ortamlarda da elde edilmesi gerçekleştirilmiş, ve uygulanabilir, verimli çalışan sistemler üretildiğini göstermiştir.**

Projenin genel olarak **gelecek çalışmalar kapsamındaki bir katkısı ise** bursiyerimiz Araş. Gör. Erdi KAYA'nın gerçekleştireceği tez olacaktır. Bu bağlamda bahsedilen dağıtık sistemin sabit düğümler değilde İHA'lar gibi hızlı bir şekilde yer değiştiren araçlar için daha farklı bir şekilde ele alınıp bu yeni sisteme göre uyarlanması planlanmış ve bu doğrultuda tez önerisi verilmiştir.

6. Sonuç ve Öneriler

Proje bünyesinde teorik çalışmalardan "cost of guessing" çatısı kapsamında iki adet IEEE ISIT bildirisi sunulmuş (**Arslan ve Haytaoglu a, 2020), (Arslan ve Haytaoglu, 2022)** ve bir adet de **IEEE Transactions on Information Theory** dergisinde incelemede bulunan makale çalışması bulunmaktadır (revizyonu hazırlanıp tekrar gönderilmiştir). Bu çalışmanın ön-baskı hali (**Arslan ve Haytaoglu, 2020 b.**)'den incelenebilinir. Bu çalışma Bölüm 5'te de belirtildiği üzere daha genelleştirilerek seçimlerin dinamik maliyetlere sahip olduğu yani her seçim sonrası diğer seçimlerin maliyetinin buna bağlı olarak değiştiği bir şekilde daha da geliştirilebilir durumdadır. Yine diğer bir uzantısı olan tahmin işleminin belli bir kritere göre sonlandırılması senaryosu da çalışılmış fakat henüz makale haline dönüştürülmüş ve literatür ile paylaşılmış durumda değildir.

Ayrıca çok katmanlı yani yerel düğümlerin yanında bazi istasyonu uydu gibi iletişim maliyetleri daha farklı olan sistemlerde bant-genişliği açısından minimum maliyete sahip kodlar kapsamında gerçekleştirilen çalışma IEEE SIU **(Pourmandi vd, 2021)** ve daha sonra IEEE ISIT **(Arslan vd., 2022)** gibi konferanslarda sunulmuştur. Yine teorik kapsamda

gerçekleştirilen ve onarım bant aralığını da işin içine katan LDPC kod tasarımları ile ilgili olan çalışmada **IEEE Communication Letters'da basılmıştır (Pourmandi vd., 2023)**. Bahsedilen son iki çalışma proje kapsamında bursiyer olan Boğaziçi Üniversitesi Elektrik-Elektronik Mühendisliği doktora öğrencisi Massoud Pourmandi tarafından doktora tez çalışmaları kapsamında gerçekleştirilmiş ve böylelikle proje çalışmaları sonucunda bir adet doktora çalışması tamamlanmıştır.

Özellikle **(Arslan vd., 2022)** tarafından gerçekleştirilen çalışma, veri depolama kapasitesi ile onarım bant aralığı arasındaki ödünleşim uzayının baz istasyonlu sistemler için karakterizasyonunu yapması açısından büyük önem taşımaktadır. Bu çalışma, önceki araştırmalardan farklı olarak, herhangi bir baz istasyonu gibi bir yedekleme varsayımı olmadan gerçekleştirildiği için, önceki çalışmaların sonuçları doğrudan uygulanamamaktadır. Bu çalışmada, sistem için kullanılacak kodların performansı için bilgi teorisi temelli sınırlar belirlenmiştir ve bu sınırlar, özellikle 5G ve sonrası ağlarda dağıtık depolama için büyük önem taşımaktadır. Minimum depolama ve minimum onarım bant aralığı noktalarında, bu çalışma sayesinde kapalı formda ifadeler elde edilmiş ve önerilen kodlama sistemleri pratik olarak gerçeklenebilir ve kullanılabilir durumdadır. Bu çalışma, baz istasyonlu sistemlerde veri depolama ve onarım bant aralığı arasında doğru bir denge sağlama konusunda önemli bir adımdır. Bu yeni bilgiler, gelecekteki kablosuz ağlarda daha verimli ve güvenilir veri iletişimi için stratejilerin geliştirilmesine yardımcı olacak ve araştırmacılara değerli bir kaynak sunacaktır.

Proje kapsamında daha pratik katkıları bulunan çalışmalardan olan; çizge kodlarının dağıtık veri ön-bellekleme kapsamında kullanılması literatüre önerilmiş olup bununla ilgili gerekli algoritmalar sunularak bu algoritmaların sonuçlarının performans katkıları yenileme kodları ve RS kodları ile kıyaslanarak analiz edilmiştir. Anlatılan çalışmalar **IEEE Transactions on Communications** dergisinde kabul edilmiş/basılmış olup **(Haytaoglu vd., 2022)'**dan da incelenebilir. Bu çalışma ise daha farklı çizge kodları için de değişik düğüm tamiri yöntemleri düşünülerek geliştirilebilir. İfade edilmelidir ki kullanılan LDPC kod tipleri 5G veri kanalında kullanılan LDPC kodları ile karşılaştırılabilir aynı karmaşıklıkta (yani düşük) kodlardır.

Ayrıca artık veri konusu ile ilgili olan çalışma da **(Erdi vd., 2022)'de IEEE SIU'da sunulmuştur.** Bu çalışma kapsamında elde edilen sonuçları doğrultusunda zaman konusunda büyük ölçüde bir kötüleşme elde edildiğinden daha da genişletilmesi gerektiği görülmüştür.

Son olarak ağ benzetim ortamında gerçekleştirilen çalışma kapsamında klasik benzetim ortamında elde edilen sonuç doğrultusunda veriler elde edilerek önceki çalışmada elde edilen klasik benzetim ortamının sonucu doğrulanmıştır. Ek olarak oluşturulan ağ benzetim ortamı **(Kaynak-Kodu, 2023)'**da açık olarak verilerek bu konuda çalışan gruplara bir kaynak olarak sunulmaktadır.

Projemizle ilintili ve proje yürütücüsünün kendisinin ya da ağındaki diğer araştırmacılarla ürettiği yayınlar da bulunmaktadır. Bu çalışmalarda bu projede elde edilen know-how'ın katkısı büyük olduğundan, proje desteği yayınlarda belirtilmiştir. Bu çalışmalara **(Arslan, 2021)** ve **(Arslan ve Zeydan, 2021)**'den ulaşılabilir.

Sonuç olarak, projemizin elde ettiği sonuçlar veri depolama iletişimi açısından 5G ve ötesi sistemler için birçok yeni sistem ve algoritma önerileri getirmiş, gerçekçi simülasyon ortamları ile verimliliği artırabileceği ispatlanmıştır. Profesyonel araştırmacı yetiştirilmesine katkılarda bulunmuş, TUBITAK STAR programı aracılığı ile lisans seviyesindeki öğrencilerimize araştırma bilinci kazandırmaya çalışmıştır.

Kaynaklar

1. Asadi, A., Wang, Q., & Mancuso, V. (2014). A survey on device-to-device communication in cellular networks. *IEEE Communications Surveys & Tutorials*, *16*(4), 1801-1819.
2. Maddah-Ali, M. A., & Niesen, U. (2014). Fundamental limits of caching. *IEEE Transactions on Information Theory*, *60*(5), 2856-2867.
3. Shanmugam, K., Golrezaei, N., Dimakis, A. G., Molisch, A. F., & Caire, G. (2013). Femtocaching: Wireless content delivery through distributed caching helpers. *IEEE Transactions on Information Theory*, *59*(12), 8402-8413.
4. Pääkkönen, J., Hollanti, C., & Tirkkonen, O. (2015, September). Device-to-device data storage with regenerating codes. In *International Workshop on Multiple Access Communications* (pp. 57-69). Springer, Cham.
5. Pedersen, J., i Amat, A. G., Andriyanova, I., & Brännström, F. (2016). Distributed storage in mobile wireless networks with device-to-device communication. *IEEE Transactions on Communications*, *64*(11), 4862-4878.
6. Pedersen, J., i Amat, A. G., Andriyanova, I., & Brännström, F. (2018). Optimizing MDS coded caching in wireless networks with device-to-device communication. *IEEE Transactions on Wireless Communications*, *18*(1), 286-295.
7. Piemontese, A., & i Amat, A. G. (2018). MDS-coded distributed caching for low delay wireless content delivery. *IEEE Transactions on Communications*, *67*(2), 1600-1612.
8. Calis, G., & Koyluoglu, O. O. (2016). On the maintenance of distributed storage systems with backup node for repair. In *2016 International Symposium on Information Theory and Its Applications (ISITA)* (pp. 46-50). IEEE.
9. Gallager, R. (1962). Low-density parity-check codes. *IRE Transactions on information theory*, *8*(1), 21-28.
10. Yongmei, W., Fengmin, C., & Cher, L. K. (2015). Large LDPC codes for big data storage. In *Proceedings of the ASE BigData & SocialInformatics 2015* (pp. 1-6).
11. Golrezaei, N., Molisch, A. F., Dimakis, A. G., & Caire, G. (2013). Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution. *IEEE Communications Magazine*, *51*(4), 142-149.
12. Dimakis, A. G., Godfrey, P. B., Wu, Y., Wainwright, M. J., & Ramchandran, K. (2010). Network coding for distributed storage systems. *IEEE transactions on information theory*, *56*(9), 4539-4551.
13. Reed, I. S., & Solomon, G. (1960). Polynomial codes over certain finite fields. *Journal of the society for industrial and applied mathematics*, *8*(2), 300-304.
14. Pääkkönen, J., Hollanti, C., & Tirkkonen, O. (2013, December). Device-to-device data storage for mobile cellular systems. In 2013 IEEE Globecom Workshops (GC Wkshps) (pp. 671-676). IEEE.
15. Pääkkönen, J., Barreal, A., Hollanti, C., & Tirkkonen, O. (2018). Coded caching clusters with device-to-device communications. IEEE Transactions on Mobile Computing, 18(2), 264-275.
16. Li, J., Gu, S., Wang, Y., & Zhang, Q. (2018, October). Double replication MDS codes for wireless D2D distributed storage networks. In *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)* (pp. 1-6). IEEE.
17. Wei, Y., Foo, Y. W., Lim, K. C., & Chen, F. (2014, December). The auto-configurable LDPC codes for distributed storage. In *2014 IEEE 17th International Conference on Computational Science and Engineering* (pp. 1332-1338). IEEE.
18. Papailiopoulos, D. S., & Dimakis, A. G. (2014). Locally repairable codes. *IEEE Transactions on Information Theory*, *60*(10), 5843-5855.
19. H. Park, D. Lee, and J. Moon, "LDPC code design for distributed storage: Balancing repair bandwidth, reliability, and storage overhead," IEEE Trans. on Communications, vol. 66, no. 2, pp. 507–520, 2017.
20. Sohn, J. Y., Choi, B., Yoon, S. W., & Moon, J. (2018). Capacity of clustered distributed storage. *IEEE Transactions on Information Theory*, *65*(1), 81-107.
21. Sohn, J. Y., Choi, B., & Moon, J. (2018, June). A class of MSR codes for clustered distributed storage. In *2018 IEEE International Symposium on Information Theory (ISIT)* (pp. 2366-2370). IEEE.
22. Chen, Z., & Barg, A. (2019). Explicit constructions of MSR codes for clustered distributed storage: The rack-aware storage model. *IEEE Transactions on Information Theory*, *66*(2), 886-899.

23. Tebbi, A., Chan, T. H., & Sung, C. W. (2019). Multi-rack distributed data storage networks. *IEEE Transactions on Information Theory*, *65*(10), 6072-6088.
24. Hou, H., Lee, P. P., Shum, K. W., & Hu, Y. (2019). Rack-aware regenerating codes for data centers. *IEEE Transactions on Information Theory*, *65*(8), 4730-4745.
25. Wang, J., Wang, T., Luo, Y., & Shum, K. W. (2019). Capacity of distributed storage systems with clusters and separate nodes. *arXiv preprint arXiv:1901.03000*.
26. Prakash, N., Abdrashitov, V., & Médard, M. (2018). The storage versus repair-bandwidth trade-off for clustered storage systems. *IEEE Transactions on Information Theory*, *64*(8), 5783-5805.
27. Yu, Q., Zeng, X., Liao, Y., & Ai, Q. (2019, August). Storage-Repair Tradeoff for Hierarchical Distributed Storage Systems. In *2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)* (pp. 1650-1654). IEEE.
28. Liu, S., Shum, K. W., & Li, C. (2020). Exact-Repair Codes with Partial Collaboration in Distributed Storage Systems. *IEEE Transactions on Communications*.
29. Zorgui, M., & Wang, Z. (2019). Centralized multi-node repair regenerating codes. *IEEE Transactions on Information Theory*, *65*(7), 4180-4206.31.
30. Bhagwan, R., Tati, K., Cheng, Y., Savage, S., & Voelker, G. M. (2004, March). Total Recall: System Support for Automated Availability Management. In Nsdi (Vol. 4, pp. 25-25).
31. Eleftheriou, E., & Olcer, S. (2002, April). Low-density parity-check codes for digital subscriber lines. In 2002 IEEE International Conference on Communications. Conference Proceedings. ICC 2002 (Cat. No. 02CH37333) (Vol. 3, pp. 1752-1757). IEEE.
32. Massey, J. L. (1994, June). Guessing and entropy. In *Proceedings of 1994 IEEE International Symposium on Information Theory* (p. 204). IEEE.
33. Arikan, E. (1996). An inequality on guessing and its application to sequential decoding. *IEEE Transactions on Information Theory*, *42*(1), 99-105.
34. Arslan, S. S., & Haytaoglu, E. (2020, June). Cost of guessing: applications to data repair. In *2020 IEEE International Symposium on Information Theory (ISIT)* (pp. 2194-2198). IEEE.
35. Shum, K. W. (2011, June). Cooperative regenerating codes for distributed storage systems. In 2011 IEEE International Conference on Communications (ICC) (pp. 1-5). IEEE.
36. R. Ahlswede, Ning Cai, S. . R. Li, and R. W. Yeung, “Network information flow,” IEEE Transactions on Information Theory, vol. 46,no. 4, pp. 1204–1216, 2000.
37. W. Shum and Y. Hu, “Cooperative regenerating codes,” IEEE Transactions on Information Theory, vol. 59, no. 11, pp. 7229–7258, 2013.
38. Wang, Z., Tamo, I., & Bruck, J. (2016, April). Explicit minimum storage regenerating codes. IEEE Transactions on Information Theory, (Vol 62(8), 4466-4480).
39. Arslan, S. S., & Haytaoglu, E. a) (2022, June), Improved Bounds on the Moments of Guessing Cost. In 2022 *IEEE International Symposium on Information Theory (ISIT)*, Espoo, Finland, (pp. 3351-3356.)
40. Arslan S. S., Pourmandi M. & Haytaoglu E. (2022, June), Base Station-Assisted Cooperative Network Coding for Cellular Systems with Link Constraints, IEEE International Symposium on Information Theory (ISIT), Espoo, Finland, 2022, pp. 2815-2820.
41. Arslan, S. S., & Haytaoglu, E. b)(2020). Cost of Guessing: Applications To Distributed Data Storage And Repair. arXiv preprint arXiv:2005.06666.
42. Pourmandi M, Pusane, A. E., Arslan, S. S. (2023, Feb) & Haytaoglu, E., Minimum Repair Bandwidth LDPC Codes for Distributed Storage Systems, in IEEE Communications Letters,27(2), pp. 428-432,
43. Kaya, E., Pourmandi, M., Haytaoglu, E., & Arslan, S. S. (15-18 May 2022). Residual Data Usage in LDPC Codes. In 2022 30th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4). IEEE. Safranbolu, Turkey.
44. Pourmandi, M., Arslan, S. S., Pusane, A. E., Haytaoglu, E., & Tengiz, A. C. (2021, June). Bİ-Yardimli Dağitik Depolama Ağlannda Ortalama Bant Genişliği Maliyeti ve Depolama Ödunleşimi Average Bandwidth-Cost vs. Storage Trade-off for BS-assisted Distributed Storage Networks. In 2021 29th Signal Processing and Communications Applications Conference (SIU) (pp. 1-4).
45. L. L. Campbell, “A coding theorem and Rényi’s entropy." Information and control 8.4: 423-429, 1965.

46. I. Sason and S. Verdú, “Improved Bounds on Lossless Source Coding and Guessing Moments via Rényi Measures," in IEEE Transactions on Information Theory, vol. 64, no. 6, pp. 4323-4346, June 2018.
47. T. Richardson and R. Urbanke, Modern Coding Theory. Cambridge University Press, 2008.
48. T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” IEEE Trans. on Information Theory, vol. 47, no. 2, pp. 619–637, 2001.
49. M. Pourmandi, A. E. Pusane, S. S. Arslan and E. Haytaoglu, "Minimum Repair Bandwidth LDPC Codes for Distributed Storage Systems," in *IEEE Communications Letters*, vol. 27, no. 2, pp. 428-432, Feb. 2023.
50. A. Shokrollahi and R. Storn, “Design of efficient erasure codes with differential evolution,” in Proc. IEEE International Symposium on Information Theory, 2000, pp. 1–5.
51. K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, 2001.
52. X.-S. Yang, Nature-inspired optimization algorithms. Academic Press, 2020.
53. W. Deng, S. Shang, X. Cai, H. Zhao, Y. Song, and J. Xu, “An improved differential evolution algorithm and its application in optimization problem,” Soft Computing, vol. 25, no. 7, pp. 5277–5298, 2021.
54. M. Satyanarayanan, "A study of file sizes and functional lifetimes." ACM SIGOPS Operating Systems Review 15.5 (1981): 96-108.
55. B. McShane et al. "Count models based on Weibull interarrival times." Journal of Business & Economic Statistics 26.3 (2008): 369-378.
56. S. S. Arslan, “Founsure 1.0: An erasure code library with efficient repair and update features,” SoftwareX, Vol. 13, pp. 1-12, January, 2021.
57. S. S. Arslan and E. Zeydan, "On the Distribution Modeling of Heavy-Tailed Disk Failure Lifetime in Big Data Centers," in IEEE Transactions on Reliability, vol. 70, no. 2, pp. 507-524, June 2021.
58. Kaynak-Kodu,2023, https://github.com/erdik1/Data-Repair-with-LDPC Son erişim zamanı (31/01/2023)