EGE ÜNİVERSİTESİ

EÜ FEN BİLİMLERİ ENSTİTÜSÜ

DOKTORA TEZİ

DAĞITIK DEPOLAMA SİSTEMLERİ İÇİN TAMİR VE YAPILANDIRIM ÜZERİNE BİR ÇALIŞMA

Elif HAYTAOĞLU

Tez Danışmanı: Prof.Dr. Mehmet Emin DALKILIÇ

Uluslararası Bilgisayar Anabilim Dalı

Sunuş Tarihi: 14.12.2015

Bornova-İZMİR

2015

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

(DOKTORA TEZİ)

DAĞITIK DEPOLAMA SİSTEMLERİ İÇİN TAMİR VE YAPILANDIRIM ÜZERİNE BİR ÇALIŞMA

Elif HAYTAOĞLU

Tez Danışmanı : Prof.Dr. Mehmet Emin DALKILIÇ

Uluslararası Bilgisayar Anabilim Dalı

Sunuş Tarihi : 14.12.2015

Bornova-İZMİR
2015

Elif HAYTAOĞLU tarafından Doktora tezi olarak sunulan "Dağıtık Depolama Sistemleri için Tamir ve Yapılandırım Üzerine bir Çalışma" başlıklı bu çalışma EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliği ile EÜ Fen Bilimleri Enstitüsü Eğitim ve Öğretim Yönergesi'nin ilgili hükümleri uyarınca tarafımızdan değerlendirilerek savunmaya değer bulunmuş ve 14.12.2015 tarihinde yapılan tez savunma sınavında aday oybirliği/oyçokluğu ile başarılı bulunmuştur.

Jüri Üyeleri :		**İmza**
Jüri Başkanı	**:** Prof.Dr. Mehmet Emin DALKILIÇ	.....
Raportör Üye	**:** Doç.Dr. Aydın KIZILKAYA	.....
Üye	**:** Doç.Dr. Orhan DAĞDEVİREN	.....
Üye	**:** Prof.Dr. Bahar KARAOĞLAN	.....
Üye	**:** Prof .Dr. Turhan TUNALI	.....

EGE ÜNİVERSİTESİ FEN BİLİMLERİ ENSTİTÜSÜ

ETİK KURALLARA UYGUNLUK BEYANI

EÜ Lisansüstü Eğitim ve Öğretim Yönetmeliğinin ilgili hükümleri uyarınca Doktora Tezi olarak sunduğum "Dağıtık Depolama Sistemleri için Tamir ve Yapılandırım Üzerine bir Çalışma" başlıklı bu tezin kendi çalışmam olduğunu, sunduğum tüm sonuç, doküman, bilgi ve belgeleri bizzat ve bu tez çalışması kapsamında elde ettiğimi, bu tez çalışmasıyla elde edilmeyen bütün bilgi ve yorumlara atıf yaptığımı ve bunları kaynaklar listesinde usulüne uygun olarak verdiğimi, tez çalışması ve yazımı sırasında patent ve telif haklarını ihlal edici bir davranışımın olmadığını, bu tezin herhangi bir bölümünü bu üniversite veya diğer bir üniversitede başka bir tez çalışması içinde sunmadığımı, bu tezin planlanmasından yazımına kadar bütün safhalarda bilimsel etik kurallarına uygun olarak davrandığımı ve aksinin ortaya çıkması durumunda her türlü yasal sonucu kabul edeceğimi beyan ederim.

14.12.2015

Elif HAYTAOĞLU

ÖZET

DAĞITIK DEPOLAMA SİSTEMLERİ İÇİN TAMİR VE YAPILANDIRIM ÜZERİNE BİR ÇALIŞMA

HAYTAOĞLU, Elif

Doktora Tezi, Uluslararası Bilgisayar Anabilim Dalı
Tez Danışmanı: Prof.Dr. Mehmet Emin DALKILIÇ
Aralık 2015, 176 sayfa

Bu tezde, verinin hata toleranslı bir şekilde depolanmasını sağlamak için kullanılan kaynaklardan başta bant genişliği ve toplam işlem süresi olmak üzere çeşitli kaynakların maliyetinin düşürülmesini sağlayacak çözümler önerilmiştir. Bu tez kapsamındaki çalışmalar beş bölüme ayrılmıştır. Bunlardan ilki MDS silinti kodlama kullanan dağıtık depolama sistemlerinde düğüm tamirinin ve verinin geri çatılmasının başlatımı ve veri güncellemesi işlemlerinde kullanılan süre ve bant genişliği maliyetini düşürmeyi hedefleyen topoloji farkındalıklı bir çalışmadır. İkinci çalışmada ise yeni bir melez kodlama şeması olan Homomorfik Minimum Bant Genişliği Tamir (HMBR) kodları geliştirilmiştir. HMBR kodlama şeması düğüm tamiri ve verinin geri çatılması işlemlerinde sırasıyla bant genişliği ve işlem süresi maliyetlerini iyileştiren iki farklı yöntem sağlamaktadır. Ayrıca önerilen kodlama şeması düğüm tamiri için bütünlük kontrolü mekanizmasına sahiptir. Üçüncü çalışmada da yeni bir melez kodlama şeması olan Homomorfik Minimum Depolama Tamir (HMSR) kodları geliştirilmiştir. HMSR kodları düğüm başına minimum depolama maliyeti gerektirirken bir yandan da sırasıyla bant genişliği ve işlem süresi maliyetlerini azaltan iki farklı düğüm tamiri yöntemi sağlamaktadır. Dördüncü çalışmada ise farklı düğümlerin farklı kodları kullanabildiği kümeleme tabanlı dağıtık bir depolama sistemi tasarlanmıştır. Son çalışmada ise silinti kodlama kullanan dağıtık depolama sistemlerinde veri geri çatma işlemi gerçekleştirilirken ağda bir tıkanma oluştuğunda, TCP soketlerinin yönetilmesi yoluyla bu işlemin toplam süresini düşüren bir algoritma önerilmiştir.

Anahtar sözcükler: Dağıtık Depolama Sistemleri, yenileme kodları, kendinden tamir kodları, topoloji, sonlu cisimler.

ABSTRACT

A STUDY ON NODE REPAIR AND DATA RECONSTRUCTION FOR DISTRIBUTED STORAGE SYSTEMS

HAYTAOĞLU, Elif

PhD in International Computer Department
Supervisor: Prof.Dr. Mehmet Emin DALKILIÇ
December 2015, 176 pages

In this thesis, solutions for reducing the cost of the sources -mainly bandwidth and total processing time- used to store data in fault tolerant way are proposed. The work in this thesis is divided into five parts. The first of these is a topology aware solution proposed for reducing time and bandwidth used for data update and the initiation of node repair and data reconstruction processes in distributed storage systems using MDS erasure codes. The second one is a new hybrid coding scheme: Homomorphic Minimum Bandwidth Repairing (HMBR) codes. HMBR codes provide two different node repair methods as well as two different data reconstruction methods for reducing bandwidth usage and the processing time. In addition, this new coding scheme presents an integrity checking mechanism for repaired node's content. In the third part of our work, another hybrid coding scheme called Homomorphic Minimum Storage Repairing Codes (HMSR) which minimizes storage cost on the nodes is designed. Moreover, these codes present two different node repair methods reducing either bandwith usage or elapsed time in node repair. In the fourth part of the thesis, the cluster based storage system is proposed in which different nodes can use different coding schemes. As the last part of our work, an algorithm is proposed for reducing the time elapsed in data reconstruction process of distributed storage systems using erasure codes in case of network congestion.

Keywords: Distributed Storage Systems, regenerating codes, self repairing codes, topology, finite fields.

TEŞEKKÜR

Bu doktora tezi süresince danışmanlığımı yapmış olan, deneyimleri, bilgileri ve önerileriyle araştırma ve geliştirmeyi yönlendirmesinden ötürü sayın Prof. Dr. Mehmet Emin DALKILIÇ'a (Uluslararası Bilgisayar Enstitüsü, Ege Üniversitesi, Türkiye) teşekkürü bir borç bilirim. Ayrıca, tezimin izleme toplantılarında jüri olarak görev alan Doç. Dr. Aydın KIZILKAYA'ya (Elektrik Elektronik Mühendisliği, Pamukkale Üniversitesi, Türkiye) ve Doç. Dr. Orhan DAĞDEVİREN'e (Uluslararası Bilgisayar Enstitüsü, Ege Üniversitesi, Türkiye) değerli önerileri ve çalışmaya olan katkılarından ötürü teşekkürlerimi sunarım.

Tez çalışması boyunca bilgilerinden faydalandığım Doç. Dr. Frédérique OGGIER (Matematiksel Bilimler Departmanı, Nanyang Teknoloji Üniversitesi, Singapur) ve Yrd. Doç. Dr. Alex DIMAKIS'e (Elektrik Bilgisayar Mühendisliği, Texas Üniversitesi, ABD) teşekkür ederim.

05-DPT-003/35 numaralı ÖYP tez projesine desteklerinden dolayı TC Kalkınma Bakanlığı'na ve Ege Üniversitesi Bilimsel Araştırma Projeleri Şubesi'ne teşekkür ederim.

Tez boyunca desteğini esirgemeyen biricik babama, biricik anneme, ablalarıma ve eşime teşekkürü bir borç bilirim.

Ege Üniversitesi Uluslararası Bilgisayar Enstitüsü'nde görevli arkadaşlarım Sinem ASLAN, Gül Boztok ALGIN, Sercan DEMİRCİ, Cihat ÇETİNKAYA, Kaya OĞUZ, Cemre CANDERMİR, Serkan ERGUN ve Moharram CHALLENGER'a, değerli hocalarıma ve idari personele teşekkür ederim.

İÇİNDEKİLER

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

İÇİNDEKİLER (devam)

Sayfa

ŞEKİLLER DİZİNİ

ŞEKİLLER DİZİNİ (devam)

ŞEKİLLER DİZİNİ (devam)

ŞEKİLLER DİZİNİ (devam)

ŞEKİLLER DİZİNİ (devam)

ŞEKİLLER DİZİNİ (devam)

ŞEKİLLER DİZİNİ (devam)

ÇİZELGELER DİZİNİ

SİMGELER VE KISALTMALAR DİZİNİ

Simgeler	Açıklama
α	Kodlanmış bir dosya için depolama düğümü tarafından depolanması gereken sembol sayısı
β	Bir düğüm arızalandığında yardımcı düğümlerin her birinden indirilmesi gereken sembol sayısı
$\mathbb{F}$	Sonlu Cisim
Δ	MDS kodlarında güncellenmek istenen verinin şu andaki hali ile yeni hali arasındaki fark
τ	HMBR kodlarında $ASRC_{NR}$ yolu ile düğüm tamirinde kullanılan yardımcı düğüm sayısı
Ψ	e-MBR ve e-MSR kodlarında kodlama matrisi
Φ	e-MBR, e-MSR, HMBR, HMSR, kodlarında bu kodlama şemalarının kodlama matrisinin bir alt matrisi
ϕ_i^t	e-MSR ve HMSR kodlarında Φ matrisinin i'inci satırı
Λ	e-MSR ve HMSR kodlarının kodlama matrisinde kullanılan köşegen matris
$\gamma(x, z, r)$	Depolama sisteminde f düğümünün arızalandığı ve çalışan x düğümün olduğu bir durumda, bu x düğümün polinom girdilerinin kertesinin r olduğu ve bu girdilerin $w_{[f]}$'ye doğrusal bağımsız olduğu permütasyonların sayısını veren fonksiyon

SİMGELER VE KISALTMALAR DİZİNİ (devam)

Simgeler	Açıklama
$\varphi(x,z,r)$	Depolama sisteminde f düğümünün arızalandığı ve çalışan x tane düğümün olduğu bir durumda kertesi r olan ve bu girdilerin $w_{[f]}$'ye doğrusal bağımlı olduğu x düğümlük permütasyonların sayısını veren fonksiyon
$\tilde{\varphi}(x,z,r)$	Depolama sisteminde f düğümünün arızalandığı ve çalışan x tane düğümün olduğu bir durumda kertesi r olan ve bu girdilerin $w_{[f]}$'ye doğrusal bağımlı olduğu ancak bu x düğümün arasında $ASRC_{NR}$ yöntemi ile f düğümünü tamir edebilen iki tane düğümün bulunmadığı x düğümlük permütasyonların sayısını veren fonksiyon
p_{tamir}	Düğüm tamirinin başarım olasılığı
p_{yapi}	Verinin geri çatılması işleminin başarım olasılığı
e_i^t	HMSR kodlarında kodlama matrisinin i'inci satırı
E_{VT}	HMSR kodlarında veri-toplayıcı düğüme veri gönderen düğümlerin ilgili E matrisi satırlarından oluşan matris
^{t}	Vektörün devriğini alma işlemi
^{T}	Matrisin devriğini alma işlemi
n	Toplam depolama düğümü sayısı
d	Düğüm tamiri için kullanılacak yardımcı düğüm sayısı
k	Verinin geri çatılması için gerekli düğüm sayısı.
E	HMBR ve HMSR kodlarında kodlama matrisi
M	e-MBR, e-MSR, HMBR, HMSR kodlarında mesaj matrisi.

SİMGELER VE KISALTMALAR DİZİNİ (devam)

Kısaltmalar	Açıklama
$AMBR_{NR}$	HMBR kodlarında bir düğüm tamiri yöntemi
$ASRC_{NR}$	HMBR kodlarında bir düğüm tamiri yöntemi
$AMBR_{DR}$	HMBR kodlarında bir verinin geri çatılması yöntemi
$ASRC_{DR}$	HMBR kodlarında bir verinin geri çatılması yöntemi
MBR	Minimum Bant Genişliği Yenileme Kodları (*Minimum Bandwidth Regenerating Codes*)
MSR	Minimum Depolama Yenileme Kodları (*Minimum Storage Regenerating Codes*)
HMBR	Homomorfik Minimum Bant Genişliği Tamir Kodları (*Homomorphic Minimum Bandwidth Repairing Codes*)
HMSR	Homomorfik Minimum Depolama Tamir Kodları (*Homomorphic Minimum Storage Repairing Codes*)
MDS	*Maximum Distance Seperable*, Bir kodlama şemasında kodlanarak n düğüme bölüştürülmüş bir dosyanın herhangi k düğümden tekrar geri çatılabilmesi özelliği
NP	Polinom zamanlı bir çözümü olmayan (*non-polynomial*)

1. GİRİŞ

Dünyada değişik profillere sahip İnternet kullanıcıları çeşitli alanlarda dijital veri üretmektedir. Buna örnek olarak *Amazon*, *eBay* vb. gibi çevrimiçi alış veriş sitelerinin kullanıcı bilgileri ya da ürün bilgilerine yenilerinin eklenmesi; hava durumu tahminlemesi için sensörlerin ölçtüğü sıcaklık, basınç gibi gerçek zamanlı veriler ya da paylaşıldıktan sonraki 12 saat içerisinde 26 milyon defa görüntülenen Ellen DeGeneres'in Oscar özçekimi *tweeti* verilebilir (Turner et al., 2014). International Data Corporation (IDC)'ın yayınladığı bir rapora göre dijital veri boyutunun 2005 yılından 2020 yılına kadar geçen süre içerisinde 300 kat artacağı tahmin edilmektedir (Gantz and Reinsel, 2012). Ayrıca 2020 yılında dünyadaki dijital veri hacminin 44 Zettabayt ya da diğer bir deyişle 44 trilyon Gigabayt olması beklenmektedir. Bu kadar hızla büyüyen veri hacmine karşı, alt yapısında İnternet kullanan sistemlerin (sosyal ağ uygulamalarının, video paylaşım sistemlerinin, e-posta hizmeti veren firmaların, hasta kayıtlarını kullanan hastanelerin vb.) arşivsel ve halihazırda kullanılan verilerini depolamak için kapasitelerini düşük maliyetlerle arttırabilmeleri, buna paralel olarak hacmi gittikçe büyüyen bu verilerin bakım maliyetlerini de düşük tutmaları önem arz etmektedir.

Depolama işlemine katılan depolama sunucuları çeşitli nedenlerden dolayı arızalanma olasılığına sahiptir. Hatta bu sunucuların zaman zaman arızalanması olağan bir durum olarak kabul edilmektedir (Ghemawat et al., 2003). Bu yüzden dağıtık depolama sistemleri verilerini bu arızalanmalardan koruyacak önlemler almalıdır. RAID (Redundant Array of Independent Disks) sistemleri, düğüm arızalanmalarına karşı önlem alan depolama sistemlerinin en bilinen örnekleri arasındadır. Ancak büyük boyutlara sahip veriler için RAID artık verimsiz kalmaktadır. Çünkü RAID eşlik tabanlıdır (*parity based*) ve eş zamanlı iki hatayı tolere edememektedir (Cleversafe, 2015). RAID sistemlerinde, verinin yeniden yapılandırılabilme başarım olasılığının sabit tutulabilmesi için; veri boyutu belli bir büyüklüklüğü geçtiğinde, basit çoklu kopyalama yöntemi sisteme eklenmektedir. Bu işlem depolama alanı açısından oldukça maliyetli olmaktadır.

RAID sistemlerinin yerine veri dağıtımı yaklaşımının (Rabin, 1989) kullanılması ile, arızalanmalara karşı daha yüksek veri dayanılıklılığı, daha az miktarda ek veri depolama yükü ile sağlanabilmektedir. Bu yöntemde veriler kodlanarak n parçaya bölünmekte, ardından bu kodlanmış n parça ağ üzerindeki depolama sunucularına dağıtılmaktadır. Bir kullanıcının sistemde depolanmış bir veriye ulaşabilmesi için sabit sayıda sunucuya erişerek depolanmış verileri sunuculardan çekmesi; ardından orijinal veriyi yeniden yapılandırması gerekmektedir. Veri dağıtımı yakla-

şımı kullanılarak düğüm yedekleme sayısı istenildiği gibi genişletilebilecek, eş zamanlı sunucu arızalanmaları tolere edilebilecektir. Ayrıca verinin dayanıklılığının sabit tutulması için veri boyutu arttıkça RAID'deki kadar yüksek miktarda depolama alanına ihtiyaç duyulmayacaktır. Veri dağıtımı yaklaşımı üçüncü bölümde detaylı bir şekilde anlatılacak olan MDS silinti (*erasure*) kodlama tekniği yardımıyla gerçekleştirilebilmektedir. Ancak MDS silinti kodları da her sistemin ihtiyacını tam olarak karşılayamamaktadır. Örneğin, bir düğüm bozulduğunda o düğümde depolanan verinin tamir edilebilmesi için MDS silinti kodlamada orijinal verinin boyutu kadar bant genişliği harcanması gerekmektedir. Oysa bu bedel, salt kopyalama tabanlı depolama sistemlerinde kayıp düğümde depolanan verinin boyutu kadardır. Bu nedenle literatürde silinti kodlama yönteminin depolama maliyeti düşüklüğü ve salt kopyalama tabanlı sistemlerin düğüm tamirindeki az bant genişliği kullanımından yararlanılabilmesi için tasarlanmış çeşitli melez kodlama teknikleri geliştirilmiştir. Bu melez kodlama tekniklerine örnek olarak *Hierarchical* kodları (Duminuco and Biersack, 2008), *Pyramide* kodları (Huang et al., 2007) ve *Double* kodları (Araujo et al., 2011) verilebilir. Ayrıca MDS *erasure* kodlarının düğüm tamirindeki verimsizliğini iyileştiren yenileme (*regenerating*) kodları (Dimakis et al., 2007), da önerilmiştir. Yenileme kodları, MDS silinti kodlarına kıyasla düğüm başında depolanması gereken veri miktarını arttırarak, düğüm tamirinde kullanılması gereken bant genişliğini azaltmayı başarmıştır. Bu yenileme kodları iki farklı uç noktayı sağlayabilmektedir. Bu uç noktalardan ilki düğüm başına minimum depolamayı sağlarken düğüm tamirinde gerekli bant genişliği harcamasını da belirli bir noktaya indirgeyebilmektedir. Bu tip yenileme kodlarına minimum depolama yenileme kodları (MSR) denilmektedir. İkinci uç noktayı sağlayan yenileme kodu ise düğüm tamirinde harcanan bant genişliği harcamasını en iyileştirirken, düğüm başına düşen depolama gereksinimini de belirli bir noktada tutmaktadır. Bu tür yenileme kodlarına ise minimum bant genişliği yenileme kodları denilmektedir.

Hızla gelişen teknolojiyle birlikte uygulama gereksinimleri de değişmiştir. Bu değişim depolama sistemlerinin performans parametrelerinin göreceli önceliklerine de yansımıştır. Örneğin bulut sistemleri için düğüm tamirindeki darboğazın bant genişliği değil I/O kullanımı olduğu tespit edilmiş ve bunun için bant genişliğinden biraz ödün veren ve düğüm tamirinde daha az sayıda düğüm kullanan *Localy Repairable* kodları (Gopalan et al., 2012; Rawat and Vishwanath, 2012; Oggier and Datta, 2011b; Papailiopoulos and Dimakis, 2012) geliştirilmiştir. Bu çalışmanın ardından düğüm tamirinde I/O kullanımını azaltan çeşitli başka kodlama şemaları da geliştirilmiştir. Bu kodlama şemaları, Datta and Oggier (2013) tarafından gerçekleştirilen literatür taramasından incelenebilir.

Her ne kadar klasik silinti kodlamayı baz alan birçok değişik kodlama şeması geliştirilmiş olsa da bazı bilinen sistemler (ör.: Facebook, Windows Azure) halen geleneksel silinti kodlama şemasını kullanmaktadır. Silinti kodlama tekniği kullanılan depolama sistemlerinde verinin geri çatılması, bozulan bir düğümün tamiri ve verinin güncellenmesi gibi temel işlemlerde, verinin dağıtıldığı topolojinin dikkate alınması depolama sisteminin performansını doğrudan ve önemli ölçüde etkilemektedir.

Gerçekleştirilen doktora tez çalışmasında, geleceğin dijital sistemlerinde kaçınılmaz görünen depolama ihtiyacı artışını karşılayan ve veri dağıtımı yaklaşımını destekleyen yedekli depolama sistemlerini çeşitli performans kriterleri açısından iyileştirebilmek ve bu sistemlerin ihtiyaçlarına yönelik servislerin tasarlanması hedeflenmiştir.

1.1 Tezin Katkıları

Bu tez kapsamındaki çalışmalar beş bölümde incelenebilir. Bunlardan ilki MDS silinti kodlama kullanan dağıtık depolama sistemlerinde düğüm tamirinin ve verinin geri çatılması işlemlerinin başlatımı ve veri güncellemesi işlemleri ile ilgili topoloji farkındalıklı bir çalışmadır. Bu çalışmada mevcut sistemlerdeki düğümlerin bulunduğu topoloji göz önünde bulundurularak belirtilen işlemlerde gerçekleşen bant genişliği harcamasının düşürülmesi ve kullanılan sürenin azaltılması hedeflenmiştir. Bunun için düğüm tamiri ve verinin geri çatılması işlemlerinin başlatımı ve veri güncellemesi işlemlerinde iki farklı iletişim yöntemi kullanılmıştır: Steiner Ağaç yöntemi ve çoklu en kısa yol yöntemi. Bu yöntemlerin bu işlemlerde kullandığı süre ve bant genişliğinin kuramsal analizi gerçekleştirilmiş olup bu sonuçlar benzetim sonuçları ile desteklenmiştir. Elde edilen sonuçlara göre düğüm tamiri ve verinin geri çatılması işlemlerinin başlatımı için kullanılan süre açısından çoklu en kısa yol yöntemi daha iyi performans gösterirken Steiner Ağaç yöntemi bant genişliği harcaması açısından çoklu en kısa yol yöntemine göre daha iyi sonuçlar vermiştir. Güncelleme işlemi için ise hem kullanılan süre hem bant genişliği harcaması açısından Steiner Ağaç yöntemi daha iyi bir performans göstermiştir. Bu çalışmanın sonucunda genellikle tam bağlı bir çizgenin üzerinde MDS silinti kodlama şemasının kullanıldığını varsayan mevcut çalışmalara bir alternatif olarak ağ topolojisinin bu sistemler üzerindeki etkisi incelenmiş olup düğüm tamiri, verinin geri çatılması işlemlerinin başlatımı ve veri güncellemesi işlemlerinde kullanılabilecek iletişim yöntemleri incelenmiştir.

İkinci çalışmada ise yeni bir melez kodlama şeması geliştirilmiştir. Bu yeni

kodlama şeması Homomorfik Minimum Bant Genişliği Tamir Kodları (HMBR) olarak adlandırılmıştır. Bu kodlama şeması yenileme kodlarındaki minimum bant genişliği noktasını sağlamaktadır. HMBR kodlama şeması düğüm tamirinde iki farklı düğüm tamiri yöntemi sunmaktadır. İlk yöntem diğerine göre daha az bant genişliği harcaması gerektirirken, ikinci yöntem daha az düğümle ve daha az hesapsal karmaşıklık gereksinimi ile düğüm tamirini gerçekleştirebilmektedir. HMBR kodları aynı zamanda verinin geri çatılması işlemi için iki farklı yöntem sağlamaktadır. Bu yöntemlerden ilki diğerine göre daha az hesapsal karmaşıklık oluştururken diğeri daha çok hesapsal karmaşıklık ve daha az bant genişiliği harcaması gerektirmektedir. Aynı zamanda bu kodlama şeması düğüm tamiri için bütünlük kontrolü mekanizmasına da sahiptir. Önerilen bu melez kodlama şeması, zaman içerisinde önceliği değişen kaynaklara sahip sistemler için farklı düğüm tamiri ve verinin geri çatımı yöntemleri sunarak, bu sistemlerin bu işlemler için toplamda harcadığı maliyeti düşürebilmektedir. Önerilen yeni kodlama şemasının düğüm tamiri ve verinin geri çatımı işlemlerindeki başarım olasılığının kuramsal analizi ve benzetimleri gerçekleştirilmiştir. Elde edilen sonuçlara göre birbirine alternatif olan düğüm tamiri yöntemlerinin toplam düğüm tamiri başarım olasığını da yükselttiği gösterilmiştir. Buna göre ilk düğüm tamiri yöntemi diğer yöntemin başarılı olmadığı durumlarda %12- 24'e varan bir olasılıkla başarım sağlayabiliyorken, ikinci düğüm tamiri yöntemi ilk düğüm tamiri yönteminin gerçekleştirilemediği durumlarda %8-10 arasına varan bir başarım olasılığı getirmektedir.

Üçüncü çalışmada da ikinci çalışmaya benzer şekilde yeni bir melez kodlama şeması geliştirilmiştir. Bu yeni kodlama şeması ise Homomorfik Minimum Depolama Tamir Kodları (HMSR) olarak adlandırılmıştır. Bu kodlar minimum depolama noktasını sağlamaktadır. Bu kodlama şeması da iki farklı düğüm tamiri işlemi bulundurmaktadır. Bunlardan birincisi ikincisine göre daha çok hesapsal karmaşıklık oluşturmakta ve daha az bant genişliği kullanımı gerektirmekte, diğer tamir etme yöntemi ise daha çok bant genişliği harcaması ve daha az hesapsal karmaşıklık gerektirmektedir. Önerilen bu kodlama şeması düşük depolama maliyeti gerektiren ve zaman içerisinde CPU, süre ve bant genişliği parametrelerine verilen önceliklerin değiştiği sistemler için bu parametrelerin toplamda oluşturduğu maliyeti azaltabilecektir. Ayrıca önerilen kodlama şemasındaki düğüm tamiri, verinin geri çatımı işlemlerinin başarım olasılığı incelenmiştir. Sonuç olarak, önerilen bu kodlama şemasında, rakip kodlama şemalarının başarım olasılıklarına yakın sonuçlar elde edilmiştir.

Dördüncü çalışmada, kümeleme tabanlı depolama sistemi çalışması gerçekleştirilmiştir. Bu çalışmanın temelini bir dağıtık depolama sistemini oluşturan dü-

ğümlerin farklı kısıtlı kaynaklara sahip olabilmesi oluşturmaktadır. Bir dağıtık depolama sistemini oluşturan düğümlerin kaynakları ve/veya iş yükleri eş yapılı bir şekilde dağılmamışsa bir kodlama şeması bir düğüm için verimli olurken diğeri için istenilen kaynak kullanımını sağlamayabilir. Bu yüzden depolama sistemini oluşturan düğümlerin farklı kaynak kullanımlarını iyileştirmeyi hedefleyen farklı kodlama şemalarını kullanması sistemin toplam performansını iyileştirebilmektedir. Bu güdüleme ile dördüncü çalışmada bir sistemi oluşturan düğümlerin kendi kaynak kısıtlarına göre kümelendirilerek her bir kümenin kendine uygun olan mevcut bir kodlama şemasını kullandığı bir dağıtık depolama sistemi ns-3 ortamında gerçekleştirilmiştir. Kodlama şemaları olarak e-MBR (Rashmi et al., 2011b), HSRC(Oggier and Datta, 2011b) ve HMBR (Haytaoglu and Dalkilic, 2013a) kodlama şemaları kullanılmıştır. Bu yeni kümeleme tabanlı depolama sisteminin düğüm tamiri ve verinin geri çatılmasındaki performansı kümeleme tabanlı olmayan depolama sistemlerinin performansları ile karşılaştırılmıştır. Elde edilen benzetim sonuçları kullanılarak kümeleme tabanlı olan ve olmayan dağıtık depolama sistemlerinin düğüm tamiri ve verinin geri çatılması işlemlerindeki maliyetleri, fonksiyon halinde oluşturularak, örnek maliyet faktörleri ile maliyet analizleri gerçekleştirilmiştir. Maliyet faktörleri olarak düğüm tamiri ve verinin geri çatımı işlemlerindeki CPU kullanımı, oluşan ağ trafiği ve kullanılan toplam süre incelenmiştir. Bu maliyet fonksiyonları, sistem tasarımcılarına kendi ortamları için en uygun sistemi seçmelerinde bir kaynak oluşturabilecek niteliktedir.

Son çalışmada ise, MDS silinti kodlama kullanan dağıtık depolama sistemlerinde verinin geri çatılma işleminde ağ tıkanıklığı problemi yaşanması durumunda, veri geri çatma işlemini hızlandıracak, ağ trafiğini düşürecek; dolayısıyla ağ tıkanıklığı problemini azaltacak yeni bir algoritma önerilmiştir. Bu algoritma ns-3 (ns-3, 2015) benzetim ortamında gerçekleştirilmiş olup benzetim sonuçlarında oluşan ağ trafiğinin azaldığı ve verinin geri çatılması işleminin daha kısa sürede tamamlandığı gözlemlenmiştir. Bu çalışmada önerilen algoritma ile MDS kodlama şeması kullanan mevcut depolama sistemlerinde verinin geri çatımı işleminde bir ağ tıkanıklığı meydana geldiğinde bu ağda kullanılması gereken toplam süre veri düğümü sayısı 30'u geçtiğinde % 16-19 arasında düşmüştür. Kullanılan mesaj sayısındaki azalma ise daha düşük olup, örnek olarak aradaki fark $k = 40$ ve 4 MB dosya parçası için 6643'tür.

İkinci bölümde, tez boyunca gerçekleştirilen beş farklı çalışma ile ilgili literatürde bulunan çalışmalar sırasıyla anlatılmıştır. Üçüncü bölümde, gerçekleştirilen çalışmaların kapsamı ve bu çalışmalarda önerilen yöntemler sırasıyla açıklanmıştır. Dördüncü bölümde her bir çalışma için önerilen yöntemler sonucunda elde edilen

bulgular verilmiştir. Son olarak beşinci bölümde tez kapsamında gerçekleştirilen çalışmalar ile ilgili tartışma, sonuç ve öneriler bulunmaktadır.

2. ÖNCEKİ ÇALIŞMALAR

Bu doktora tezi kapsamında gerçekleştirilen çalışmalar beş farklı başlık altında toplanmıştır. Bunlardan ilki MDS silinti kodlarını kullanan dağıtık depolama sistemleri için topoloji farkındalıklı bir çalışmadır. İkincisi melez bir kodlama yöntemi olan homomorfik minimum bant genişliği tamir kodları çalışmasıdır. Üçüncü çalışma ise yine melez bir kodlama yöntemi olan homomorfik minimum depolama tamir kodları çalışmasıdır. Dördüncü çalışma kümeleme tabanlı bir depolama sisteminin geliştirilmesi ile ilgilidir. Son çalışma ise silinti kodlama kullanan dağıtık depolama sistemlerinde geri çatma işleminde meydana gelen bir ağ tıkanıklığı durumunda geri çatma işleminin hızlandırılması ile ilgilidir. Bu tez kapsamındaki literatür özeti bu çalışmalar için beş ayrı alt bölümde anlatılmıştır.

2.1 MDS Silinti Kodlama Kullanan Dağıtık Depolama Sistemleri İçin Topoloji Farkındalıklı Bir Çalışma İle İlgili Literatür Özeti

Literatürde, silinti kodlarını kullanan dağıtık depolama sistemlerinde düğüm tamiri işleminin iletişim maliyetini azaltmayı hedefleyen çeşitli çalışmalar bulunmaktadır. Li et al. (2009) ağaç tabanlı bir iletişim yöntemi önererek düğüm tamiri maliyetini azaltmış ve bu yöntemin kullanılması durumunda düğüm tamiri başarım oranını incelemişlerdir. Bu çalışmada, kodlama şeması olarak (n, k) doğrusal ağ kodlama şeması kullanılmıştır. Önerilen ağaç tabanlı iletişim yönteminde yardımcı düğümler kendine ait kodlanmış veri parçalarını diğer yardımcı düğümlerin parçalarıyla birlikte kodlayarak yeni gelen düğüme iletmektedir. Yine Li et al. (2010) ağ topolojisini göz önünde bulundurarak düğüm tamirini daha hızlı gerçekleştiren bir yöntem önermişlerdir. Bu çalışmada, tamir etme işleminde kullanılmak üzere düğümler arasındaki iletişim yollarını belirleyen dört farklı topoloji karşılaştırılmıştır. Bunlar; yıldız topolojisi, *Rcstar* topolojisi (yenileme kodu kullanan yıldız topolojisi), ağaç topolojisi ve *Rctree* topolojisi (yenileme kodu kullanan ağaç topolojisi) dir. Matematiksel analizlere ve benzetim sonuçlarına göre topolojilerden en iyi sonucu, tamir işlemi boyunca düğüm kayıplarını da tolere edebilen *Rctree* topolojisi vermiştir. Bu çalışmada temel olarak yüksek bant genişliğine sahip hatların kullanılmasına öncelik verilerek düğüm tamiri iletişiminin daha hızlı gerçekleştirilmesine odaklanılmıştır. Güncelleme işleminin ya da verinin geri çatılmasına ait kontrol mesajlarının iletişim maliyeti gibi konular incelenmemiştir. Ayrıca iyileştirilmesi düşünülen iletişim modeli içe gönderim (*Incast*) iletişim modelidir.

Gerami et al. (2011) ise, silinti kodlama kullanan dağıtık depolama sistemlerinde düğüm tamir etme işleminin iletişim maliyetini, sistemin topolojisini dikkate alarak azaltmaya çalışmışlardır. Bu çalışmada, en düşük maliyet gerektiren bir çoğa gönderim iletişim modelini tasarlama probleminin Minimum Steiner Ağaç Problemine denk geldiği belirtilmiştir. Ayrıca bu yöntemi ağ kodlama kullanmadan Steiner ağaç yöntemiyle çözmeye çalışmanın NP-zor (*NP-hard*) olduğu belirtilmiştir ve yenileme kodları kulanılan sistemlerde çoğa gönderim problemi Simpleks yöntemiyle çözülebilecek bir doğrusal programlama problemi formuna dönüştürülmüştür. Burada da yardımcı düğümlerin diğer yardımcı düğümlerin paketlerini kodlaması düşünülmüştür. Ancak, önerilen çözümler *tandem*, yıldız ve *grid* ağlarında uygulanmış olup, İnternet'in yapısına benzer bir topolojide uygulanmamıştır. Ayrıca veri güncellemesi ya da verinin geri çatılması işlemleri bu çalışma kapsamında incelenmemiştir.

Literatürde, MDS silinti kodlarının güncelleme işlemleri ile ilgili de çeşitli çalışmalar bulunmaktadır. Aguilera et al. (2005) doğrusal sistematik kodlarda eş zamanlı güncelleme işlemi üzerinde çalışmışlar ve eş zamanlı güncellemeler altında kodlanmış verinin geri çatılabilirliğinin sürdürülebilmesi için matematiksel sınırlar belirlemişlerdir. Ancak bu çalışmada düğümlerin bulunduğu ağ topolojisinin etkisi incelenmemiştir.

Anthapadmanabhan et al. (2010) dağıtık depolama sistemlerinde çok sık güncellenen verilere odaklanmışlardır. Bu kapsamda, kodlama şeması olarak logaritmik hesapsal karmaşıklığa sahip olan rastgele tipte bir kod sınıfı önermişlerdir. Ancak bu çalışmada da ağ topolojisinin bu kodlar üzerindeki etkisi üzerinde durulmamıştır.

Reed-Solomon kodları (Reed and Solomon, 1960) kullanılarak verilerin düğüm arızalanmalarına karşı dayanıklı bir şekilde nasıl depolanabildiğini açıklayan bir rapor yayınlamıştır (Plank, 1997). Plank verinin nasıl kodlandığını, nasıl güncellendiğini ve bazı düğüm arızalanmaları yaşandığında sistemin nasıl ilk haline getirilebileceğini detaylı bir şekilde açıklamıştır. Ancak bu kodların ağ topolojisiyle etkileşimi bu çalışmada da açıklanmamıştır.

Minimum Steiner Ağaç Problemi, bilgisayar ağlarında birden çoğa gönderim iletişimlerinde toplam iletişim maliyetinin azaltılması amacıyla kullanılabilmektedir. Literatürde, Minimum Steiner Ağaç'ını oluşturmak için tasarlanmış çeşitli yaklaşım algoritmaları bulunmaktadır. Minimum Steiner Ağaç problemi için yaklaşık çözüm üreten algoritmalar (*approximation algorithms*) (Berman and Ramaiyer, 1994; Karpinski and Zelikovsky, 1995; Hougardy and Prömel, 1999; Dolagh and

Moazzami, 2011) çalışmalarında görülmektedir. Bu algoritmalardan diğerlerine nispeten daha kolay uygulanabilir olanı Dolagh ve Moazzami'nin 1,5 ile en iyi yaklaşım oranına sahip olan algoritma "*New Approximation Algorithm for Minimum Steniner Tree Problem*" makalesinde bulunmaktadır (Dolagh and Moazzami, 2011).

Chekuri et al. (2006) ise bir çoğa gönderim oturumunun veriminin, hedef düğümlerin birim zamanda aldığı veri miktarı ile doğru orantılı olduğunu ifade etmişlerdir. Bir çoğa gönderim işleminde ağ kodlamanın, verimi nasıl etkilediğini incelemişler, bunu yaparken Steiner Ağaç Problemi'nin doğrusal programlama formunu kullanmışlardır.

Şimdiye kadar yapılan çalışmaların bazılarında çoğa gönderim iletişim modeli olarak Minimum Steiner Ağaç yöntemi düşünülmüş olsa da, bu yöntem ya doğrusal programlama formuna dönüştürülmüş ya da NP-zor olduğu belirtilip bu yöntem kullanılmamıştır. Literatürde depolama sistemlerinde kullanılan silinti kodları ile ilgili Minimum Steiner Ağaç Probleminin yaklaşım algoritmasının kullanıldığı bir çalışmaya rastlanmamıştır. Literatürde silinti kodlarını kullanan dağıtık depolama sistemlerinde doğrudan kodlama işlemi ile ilgili iyileştirme çalışmaları yapılmıştır, ek kontrol mesajlarının maliyetini düşürmeyi amaçlayan bir çalışma ise bulunmamaktadır.

Tez kapsamında yapılan bu çalışmada, silinti kodlarını kullanan bir dağıtık depolama sisteminin iletişim performansının, bu sistemdeki depolama düğümleri arasında Steiner Ağaç yaklaşım algoritmasının kullanıldığı ve kullanılmadığı durumlarda nasıl etkilendiği incelenmiştir (Haytaoglu and Dalkilic, 2013b). Steiner Ağacının kullanıldığı durumda depolama düğümleri (verileri silinti kodlama yöntemiyle depolayan düğümler) arasında oluşan bir çoğa gönderim oturumuna istemci düğümleri (veri düğümlerini depolama sistemi olarak kullanan düğümler) de dahil edilerek, iletişimin bir ağaç üzerinden gerçekleştirilmesi durumu incelenmiştir. Ayrıca, silinti kodlama kullanan dağıtık depolama sistemini oluşturan veri düğümleri arasında gerçekleştirilen verinin geri çatılması, düğüm tamiri ve verinin güncellenmesi gibi işlemler için bu farklı iletişim modelleri kullanıldığında ortaya çıkan iletişim maliyetleri analiz edilmiştir.

2.2 HMBR Çalışması İle İlgili Literatür Özeti

Bu bölümde Homomorfik Minimum Bant Genişliği Tamir (HMBR) kodları ile ilgili çalışmalar incelenmiştir.

Düğüm tamir etme işlemine geleneksel silinti kodlarına kıyasla daha çok yardımcı düğümün dahil edilmesiyle, bu işlem için harcanan bant genişliğini azaltan yenileme kodları, Dimakis et al. (2007) tarafından önerilmiştir. Ancak bu kodlar düğüm tamiri için gerekli olan bant genişliği harcamasını azaltırken, bir düğümde depolanması gereken veri miktarını arttırmaktadır. *Yenileme kodları*, minimum bant genişliği kullanımı noktası ve minimum depolama kullanımı noktası arasında optimal bir ödünleşim sağlamaktadır. Bu kodların teorik olarak varlığı ispatlandıktan sonra, literatürde yenileme kodlarının uç noktaları için çeşitli MSR (Minimum Depolama Yenileme) ve MBR (Minimum Bant Genişliği Yenileme) kod tasarımları önerilmiştir. Örneğin matris çarpımına dayanan optimal kesin $[n, k, d \geq 2k - 2]$ MSR ve $[n, k, d]$ MBR kodlarının açık bir şekilde oluşturulması gösterilmiştir (Rashmi et al., 2011b) . Bu kesin MSR ve MBR kodları, literatürde bulunan kesin-MBR ve kesin-MSR [1] kodlarının n'ye bağlı olmadığı ilk örneğidir.

Yenileme kodlarında depolama ve bant genişliğindeki iki uç noktanın teorik sınırları aşağıdaki kesim sınırı (*Min Cut-Max Flow*) aracılığıyla bulunmuştur (Dimakis et al., 2007):

$$B \leq \sum_{i=0}^{k-1} min\{\alpha, (d-i)\beta\} \ . \tag{2.1}$$

Eşitlik (2.1)'deki B, kodlanan orijinal verideki toplam sembol sayısı, k verinin geri çatılması işleminde veri-toplayıcı düğümün bağlanması gereken düğüm sayısı, d ise düğüm tamirinde yeni-gelen düğüm tarafından bağlanılması gereken yardımcı düğüm sayısıdır. Eşitlik (2.1)'den elde edilen minimum depolama yenileme noktasında (MSR), bir düğümde depolanması gereken sembol sayısı α ve düğüm tamirinde yardımcı düğümlerin herhangi birinden indirilmesi gereken sembol sayısı (bant genişliği) β sırasıyla aşağıda verilmiştir (Dimakis et al., 2007):

$$\alpha = \frac{B}{k}. \tag{2.2}$$

$$\beta = \frac{B}{k(d-k+1)}. \tag{2.3}$$

Eşitlik (2.1)'den elde edilen minimum bant genişliği yenileme noktasında (MBR), bir düğümde depolanması gereken sembol sayısı α ve düğüm tamirinde yardımcı düğümlerin bir tanesinden indirilmesi gereken sembol sayısı (bant genişliği) β sırasıyla aşağıda verilmiştir (Dimakis et al., 2007):

$$\alpha = \frac{2dB}{k(2d-k+1)}. \tag{2.4}$$

[1] Buradaki kesin kelimesinin kattığı anlam düğüm tamirinde kayıp veri ile tamir sonucu elde edilen verinin birbiriyle aynı olmasıdır.

$$\beta = \frac{2B}{k(2d-k+1)}. \tag{2.5}$$

Gastón et al. (2011) neredeyse çevrimsel (*quasi-cyclic*) mimariye dayanan yeni bir MSR kodlama şeması önermişlerdir. Bu kodlar arızalanan bir düğümün tamiri için düşük hesaplama karmaşıklığı gerektirirken, ilave bir kısıt olarak bu işlem için gerekli olan yardımcı düğüm sayısı $k+1$'dir. Shum (2011) ise birden fazla düğüm arızalanması oluştuğunda, bir düğüm tamiri işlemi başına harcanan bant genişliğinin yardımcı düğümlerin birbirleriyle iletişimde bulunmasıyla azaltılabileceğini göstermişlerdir. Bunu gerçekleştirebilen kodlama şemalarına işbirlikçi kodlama şeması denilmektedir. (Shum, 2011)'de bu kodların optimal olarak oluşturulması da açık bir şekilde gösterilmiştir.

Yenileme kodları bazı uygulama alanları için çok verimli olamamaktadır. Örneğin bulut tabanlı sistemlerde kodlama şemalarının kullanımını zorlayan temel kısıtlardan biri I/O kullanımıdır. Bu yüzden düğüm tamirinde olabildiğince az sayıda yardımcı düğüme bağlanılması bulut tabanlı sistemlere avantaj sağlayacaktır. Bu amaçla, düğüm tamirinde gerekli olan yardımcı düğümlerin sayısını azaltan yeni kodlama şemaları geliştirilmiştir. Örneğin (Huang et al., 2012)'de, düğüm tamirinde gerekli olan yardımcı düğüm sayısını azaltırken bant genişliği harcamasını da düşük seviyede tutabilen *Yerel Yenileme* (*Local Reconstruction*) kodları önerilmiştir. Bu kodlar okuma gecikmesi süresinin kritik olduğu *Windows Azure*'de (Calder et al., 2011) kullanılmıştır. Ayrıca, düğüm tamirinde kullanılan yardımcı düğüm sayısını azaltan bir başka çalışma ise *Self-Repairing* kodları (SRC)'dır (Oggier and Datta, 2011b). SRC'de arızalanan bir düğüm az sayıda yardımcı düğüm ile tamir edilebilmektedir. Bu kodların bariz örneği olan Homomorfik *Self-Repairing* kodları (HSRC) ilgili çalışmada gösterilmiş olup HSRC zayıf doğrusallaştırılmış polinomların homomorfizm özelliğinin kullanılması ile oluşturulmaktadır.

Literatürde bulunan kodlama tekniklerini birleştiren bir takım çalışmalar da mevcuttur. Duminuco and Biersack (2008), salt kopyalama ve silinti kodlamanın istenilen özelliklerinin tek bir kodlama şeması altında birleştirildiği hiyerarşik kodları önermişlerdir. Bu kodların farklı şekillerde oluşturulması ile depolama yükü, dayanıklılık ya da düğüm tamirinde kullanılan yardımcı düğüm sayısı gibi parametrelerin maliyetleri ihtiyaca göre düzenlenebilmektedir. Bir parametrenin maliyetinin azaltılması diğer parametrelerin maliyetinin artmasına neden olabilmektedir. Bu kodlar özdeş (P2P) ağlar için uygundur. Başka bir melez kodlama ise önerilen *Double* kodlarıdır (Araujo et al., 2011). Bahsedilen çalışmada *Double* kodlarının, Reed-Solomon kodlarının ve melez kodlarının (Reed-Solomon kodları ile salt-kopyalama yaklaşımını birleştiren kodların) ve yenileme kodlarının dayanıklılık ve

depolama alanı açılarından birbirleriyle karşılaştırılması da bulunmaktadır (Araujo et al., 2011).

Pamies-Juarez et al. (2013b) arızalanan bir düğümün çeşitli yardımcı düğüm kümeleriyle tamir edilebildiği bir kodlama şeması üzerinde çalışmışlardır. Bu kodlar yerel kodlara göre daha yüksek tamir edilebilme olasılığına sahiptir. Bahsedilen çalışmada ayrıca bu kodların açık bir örneği olan pg-BLRC kodları da gösterilmiştir. Bozulan bir düğümü tamir edebilen yardımcı düğüm kümelerinin sayısını arttırmak ya da bu kümelerdeki düğüm sayısını azaltmak bu kodların düşük $\frac{k}{n}$ oranlarına sahip olmasına neden olmaktadır. Kamath et al. (2013) yerelliği sağlayan minimum bant genişliği yenileme kodlarını çalışmışlardır. Bu kodlar Gabudilun kodlamaya (Gabidulin, 1985) dayalı bir ön-kodlama işlemi gerektirmektedir. Ancak bu kodlarda ara-kodun farklı alt kümelerine ayrı ayrı MBR kodlama işlemi uygulanmaktadır. Rashmi et al. (2014) düğüm tamirinde geleneksel silinti kodlarına göre daha az bant genişliği gereksinimi olan ancak bu kodlara göre ek bir depolama gerektirmeyen yeni bir kodlama şeması önermişlerdir. Ancak bu kodlama şemasında bir birimin içerisindeki tüm baytların aynı kaderi paylaştığı varsayılmaktadır. Yani, bir birim içerisindeki bir bayt erişilemez durumdaysa aynı birim diğer tüm verilerin de erişilemez halde olduğu kabul edilmektedir.

Literatürde, kodlama şemalarının farklı yönleri üzerinde de çalışmalar gerçekleştirilmiştir. Pamies-Juarez et al. (2013a), dağıtık depolama sistemlerinde kodlama hızını incelemişlerdir. Kodlama süresinin azaltılması için boru-hattı stratejisini önermişlerdir. Bu stratejinin, tek bir dosyanın kodlama süresini azaltırken, birden çok dosyanın eş-zamanlı kodlama süresini de azalttığı ifade edilmiştir. Hu et al. (2013) ise yeni bir tür minimum depolama yenileme kodu tasarımı önermişlerdir. Bu yeni tip MSR kodlarında düğüm tamirinde kodlama işlemi gerekmemektedir, böylece düğüm tamirinde harcanan süre kısaltılmıştır.

Bu tez kapsamında geliştirilen HMBR kodları, literatürde iki farklı düğüm tamiri yöntemi ve iki farklı veri geri çatılması yöntemi sunan, aynı zamanda tamir edilen düğümün bütünlüğünün kontrol edilmesini sağlayan ilk kodlama şemasıdır. Bu yeni kodlama şeması ile yakından ilişkili iki tane kodlama şeması mevcuttur. Bu kodlama şemaları HSRC (Oggier and Datta, 2011b), e-MBR (Rashmi et al., 2011b) kodlama şemalarıdır. Aşağıdaki alt bölümlerde bu kodlama şemaları ayrıntılı bir şekilde incelenmiştir.

2.2.1 HSRC kodları

Oggier and Datta (2011b), *Self Repairing Codes* (SRC) ismini verdikleri yeni bir kodlama şeması sınıfı önermişlerdir. Aynı zamanda bu sınıfa giren yeni bir kodlama şeması olan *Homomorphic Self Repairing* Kodlarının (HSRC) bir örneğini göstermişlerdir. Bu türde bir kodlama şemasında düğüm tamiri az sayıda yardımcı düğüm ile gerçekleştirilebilirken (yerel tamir), verinin geri çatılması işlemi diğer MDS kodlarına göre daha düşük olasılıkla gerçekleştirilebilmektedir. Bu kodlar homomorfizm özelliğini kullanmak için zayıf doğrusal polinomlardan yararlanmaktadır.

Tanım 2.1. *$\mathbb{F}_q$ $(q = 2^m)$ üzerinde bir zayıf doğrusal polinom $p(x)$ şu forma sahiptir (Oggier and Datta, 2011b):*

$$p(x) = \sum_{i=0}^{k-1} p_i x^{2^i},\ p_i \in \mathbb{F}_q.$$

Ön Kuram 2.1. *$a, b \in \mathbb{F}_{2^m}$ olsun ve $p(x)$ bir zayıf doğrusal polinom olsun: $p(x) = \sum_{i=0}^{k-1} p_i x^{2^i}$. Bu polinom aşağıdaki özelliği sağlamaktadır:*

$$p(a+b) = p(a) + p(b)$$

İspat. *$p(X)$ polinomunun $a+b \in \mathbb{F}_{2^m}$ girdisiyle hesaplanması işlemi aşağıda gösterilmiştir (Oggier and Datta, 2011b):*

$$p(a+b) = \sum_{i=0}^{k-1} p_i (a+b)^{2^i} = \sum_{i=0}^{k-1} p_i (a^{2^i} + b^{2^i}) = p(a) + p(b)$$

HSRC kodlarında M boyutlu bir orijinal dosya öncelikle k tane eşit parçaya bölünür. Ardından dosya parçasını oluşturan değerler katsayı olarak kullanılarak bir zayıf doğrusal polinom oluşturulur. Diğer bir deyişle, p_i'nin değeri (i)'inci orijinal veri parçasına denk gelmektedir. Kodlama işlemi $p(X)$ polinomunun n tane farklı girdi ile hesaplanması sonucunda gerçekleştirilir. Sonuçta elde edilen n tane farklı değer kod kelimesine denk gelmektedir. HSRC kodları ile düğüm tamiri ve verinin geri çatılması işlemleri ilerleyen alt bölümlerde açıklanmıştır.

2.2.1.1 HSRC kodlama şemasında düğüm tamiri

Bozulan bir düğümün tamirinde; polinom girdilerinin $\mathbb{F}_2$ üzerinden doğrusal bir kombinasyonu bozulmuş düğümün polinom girdisini oluşturan düğümler kul-

lanılmaktadır. Örneğin $p(a)$ değerini depolayan düğüm bozulmuş ve $a = b + c \mid \{a, b, c\} \in \mathbb{F}_{2^{M/k}}$ olsun. Bu durumda, bozulan düğümün içeriği $p(b)$ ve $p(c)$ kod parçalarını depolayan düğümler tarafından tamir edilebilir. Eğer bu koşulu sağlayan herhangi ikili düğüm yoksa $p(a)$ değeri üç düğümden, o da yoksa k tane düğüme kadar bir alt kümeden tamir edilebilir.

2.2.1.2 HSRC kodlama şemasında verinin geri çatılması

HSRC şemasında kodlanmış bir dosyanın yeniden elde edilmesi için, polinom katsayılarının oluşturduğu vektörleri $\mathbb{F}_{2^{M/k}}$ üzerinden doğrusal bağımsız olan k tane düğüm kullanılmaktadır. Veri-toplayıcı düğüm bu koşulu sağlayan herhangi k düğüme bağlanır ve bu düğümlerde kayıtlı verileri aldıktan sonra Lagrange aradeğerleme ya da geleneksel Reed-Solomon (Reed and Solomon, 1960) tarzı verinin geri çatılması işlemi ile orijinal dosyanın içeriğini elde eder.

2.2.2 e-MBR kodları

Bu kodlama şemasında, arızalanan bir düğüm d farklı yardımcı düğüme bağlanılarak minimum bant genişliği kullanımı ile tamir edilir. Bu kodlama şemasında da kodlanmış bir dosyanın parçaları n tane farklı düğüme dağıtılır. Orijinal dosya herhangi k düğüme bağlanılarak toplamda kd sembol bant genişliği harcaması ile geri çatılır. Bir dağıtık depolama sisteminin $[n,\ k,\ d]$ e-MBR kodlama şemasını kullandığı ve kodlanacak dosyada $B = \binom{k+1}{2} + k(d-k)$ tane sembol olduğu varsayılsın. Bu kodlama şemasının $d \times d$ boyutlu simetrik mesaj matrisi M orijinal veri sembolleri ile doldurulmaktadır. M matrisi dört tane farklı alt matris yapısından oluşmaktadır. Bunlardan ilki olan $k \times k$'lık simetrik S matrisinin üst yarım kısmı kodlanacak orijinal verinin ilk $\binom{k+1}{2}$'lik sembolü ile doldurulmaktadır. S matrisinin geriye kalan elemanları ise üst kısmın simetriği olacak şekilde alt kısma kopyalanır. İkinci matris yapısı T, M matrisinin sağ üst $k \times (d-k)$'lık alt matrisini oluşturmaktadır ve dosyanın sonraki $k(d-k)$ sembolü ile doldurulur. Üçüncü matris T matrisinin devriğidir ve M matrisinin $k+1.$ ve $d.$ satırlarının arasında yer alır. M matrisinin geriye kalan elemanları ise 0 sembolü ile doldurulmaktadır.

Bu kodlama şemasındaki kodlama matrisi ise $n \times d$ boyutuna sahip Ψ matrisidir. Bu matris kendi içerisinde $(n \times k)$'lık Φ ve $n \times (d-k)$'lık Δ matrislerini ihtiva eder. Ψ matrisi Cauchy (Bernstein, 2005) ya da Vandermonde (Turner, 1966) matrisi olarak oluşturulabilir. e-MBR kodlama şemasında düğüm tamiri ve verinin geri çatılması işlemleri aşağıda açıklanmıştır.

2.2.2.1 e-MBR kodlama şemasında düğüm tamiri

Yeni-gelen düğüm, bozulan bir düğümü tamir etmek için d tane yardımcı düğüme bağlanır. Bu d yardımcı düğümün her biri kendi depoladığı sembol satırı ile bozulmuş düğüme karşılık gelen kodlama matrisi satırının devriğini çarpar ve sonucu yeni-gelen düğüme gönderir. Yeni-gelen düğüm aldığı değerlerle yardımcı düğümlerin belirteçlerinin sırasını da gözönünde bulundurarak bir matris hazırlar. Yeni-gelen düğüm ayrıca yardımcı düğümlerin kodlama matrisi satırlarından bir alt kodlama matrisi üretir. Ardından bu matrisin tersini alır ve elde ettiği matrisi, yardımcı düğümlerden aldığı değerlerle hazırladığı matrisle çarparak bozulmuş düğümün içeriğini elde eder.

2.2.2.2 e-MBR kodlama şemasında verinin geri çatılması

Veri-toplayıcı düğüm herhangi k düğüme bağlanır ve bu düğümlerden orijinal veriyle ilgili kodlanmış tüm sembolleri indirir. Veri-toplayıcı düğüm aynı zamanda sadece veri topladığı düğümlerin ilgili kodlama satırlarından Φ'nin bir alt matrisini yani Φ_{VT}'yi oluşturur. Φ_{VT} yapısı gereği tersinir olduğu için; veri-toplayıcı düğüm önce T'yi açığa çıkartır. Ardından da T'yi kullanarak S'yi yeniden oluşturur. S ve T orijinal veriden oluşturulan matrisler olduğu için orijinal veri elde edilmiş olur.

Önerilen HMBR kodları, yukarıda açıklanan bu iki kodlama şemasının; HSRC ve e-MBR kodlarının istenilen özelliklerinin bir kodlama şeması içerisinde bulundurulabilmesi amacıyla geliştirilmiştir. Bu özellikler, e-MBR kodlarında düğüm tamirinin düşük bant genişliği ile gerçekleştirilebilmesi, HSRC kodlarında düğüm tamirinin düşük hesapsal karmaşıklık ve az sayıda yardımcı düğüm ile gerçekleştirilebilmesidir.

2.3 HMSR Çalışması İle İlgili Literatür Özeti

HMSR kodları; e-MSR (Rashmi et al., 2011b) ve HSRC (Oggier and Datta, 2011b) kodlarının avantajlarını kullanmak için tasarlanmış olan melez bir kodlama yöntemidir. e-MSR (Rashmi et al., 2011b) kodları, e-MBR kodlama yöntemine benzemekte ancak minimum depolama özelliğini sağlayan matris çarpımına dayalı bir yenileme kodlama (*regenerating code*) şemasıdır. HSRC (Oggier and Datta, 2011b) kodlama şeması ise Alt bölüm 2.2.1'de anlatılmış olan düğüm tamirinde yerelliği sağlayan bir kodlama şemasıdır.

HSRC kodlarını geliştiren (Oggier and Datta, 2011b) bu çalışmanın ardından

Self Repairing kodlarının bir örneği olarak yeni bir kodlama şeması olan *Projective Self Repairing* kodlarını (PSRC) (Oggier and Datta, 2011a) önermişlerdir. Bu kodlama şeması izdüşümsel geometriye dayanmaktadır. Bu kodlama şemasında e-MBR kodlama şemasında olduğu gibi tek bir düğümde tek bir sembol depolanması yerine birden fazla sembol depolanmaktadır. PSRC kodlarında düğüm tamiri iki veya daha çok yardımcı düğüm ile yapılabilmektedir. Verinin geri çatılması için ise PSRC kodlarında k düğüme bağlanılmaktadır. Ancak bağlanılan k düğüm izdüşümsel geometrinin özelliklerine göre seçilmektedir, yani verinin geri çatılması için bağlanılan düğümler herhangi k düğüm olamamaktadır. Önerdiğimiz HMSR kodları gibi PSRC kodları da MDS özelliğini taşımamaktadır. PSRC kodlama şeması depolama kısıtında minimum depolama noktasını sağlayabiliyorken, MSR kodlarının düğüm tamirinde harcadığı bant genişliği değerinin altında bant genişliği harcaması yapabilmektedir. Ancak, düğüm tamirinde kullanılabilen düğümler gelişigüzel seçilemeyip belirli kısıtları sağlamak zorundadır (Oggier and Datta, 2011a).

Rashmi et al. (2011a), kesin ve optimal yenileme kodlarının ilk açık örneklerini vermişlerdir. Bu çalışmada MSR kodlarında bir düğüm tamirinde bağlanılan yardımcı düğüm sayısı d, $k+1$'e, bir kodlama işleminde kodlanacak sembol sayısı B ise $2k$'ya eşit olmalıdır. Bu kodların kodlama işleminde $k \times 1$ boyutlu bir MDS kodlama vektörü ve bir de $k \times 1$ boyutlu elemanları sıfırdan farklı bir vektör kullanılmaktadır. Bu MSR kodlarında her düğüm bir kodlama işlemi için iki tane sembol depolamaktadır. Bu kodların düğüm tamirinde ise her yardımcı düğümün depoladığı sembollerin bir kombinasyonunu ağ üzerinden göndermesi gerekmektedir.

MSR kodlarının oluşturulma yöntemlerinden biri de neredeyse çevrimsel (*quasi-cyclic*) mimarinin kullanılmasıdır. Gastón et al. (2011) neredeyse çevrimsel bir MSR kodlama şeması önermişlerdir. Bu kodlama şeması düğüm tamirinde düşük hesapsal karmaşıklık gerektirdiği için bu kodlama şemasının gelişmiş hesaplama kaynağı bulunmayan basit sabit diskler için kullanılabileceği vurgulanmıştır. Ancak, bu kodlama şemasında, bir düğüm tamirinde kullanılan yardımcı düğüm sayısı d için $d = k + 1$ olma zorunluluğu bulunmaktadır.

Vignesh and Thangaraj (2013) kodlama ve geri çatma hesaplama karmaşıklığı düşük olan neredeyse çevrimsel MSR kodları üzerinde çalışılmışlardır. Bu çalışmada, sistematik düğüm sayısı üçü geçtiğinde sembol genişlemesi bulundurmayan çevrimsel kodların bulunmadığı teorik olarak ispatlanmıştır. Ayrıca bu çalışmada uygulama karmaşıklığını azaltan MSR noktasını kısmen sağlayan neredeyse çevrimsel kodlar gösterilmiştir.

Suh and Ramchandran (2011), $\frac{k}{n} \leq \frac{1}{2}$, $d \geq 2k-1$ ve $k \leq 3$ durumları için de kesin-MSR kodlarının varlığını göstermişlerdir. Önerdikleri kodlarda kablosuz ağlarda kullanılan *interference alignment* tekniği kullanılmaktadır. Bu çalışmada, *interference alignment* tekniğini kablosuz ağlarda kullanmaktansa depolama sistemleri için kullanmanın daha kolay olduğu çünkü kablosuz ağlarda kanal katsayılarının doğadan geldiği ancak veri depolama sistemi için bu katsayıların tasarımcı tarafından atanabildiği ifade edilmiştir. Ancak, bu çalışmada yardımcı düğüm sayısı $n-1$ olmak zorundadır, yani tolere edilebilen düğüm arızası sayısı sadece birdir. Bu kesin-MSR çalışmasının ardından, Chen and Shum (2013), Suh and Ramchandran (2011) tarafından önerilen kesin-MSR çalışmasını genişleterek *interference alignment* tekniğinin kullanıldığı bir kesin-MSR kodlamasında iki tane arızalanmış düğümün işbirlikçi tamir etme yöntemi kullanıldığında optimal bant genişliği harcamasıyla da tamir edilebildiğini göstermişlerdir.

Literatürde, geleneksel silinti kodlarının düşük depolama maliyeti avantajından yararlanan melez kodlama şemaları da mevcuttur. Bunlardan biri *Pyramid* kodlarıdır (Huang et al., 2007). *Pyramid* kodları geleneksel silinti kodlamaya yakın ek depolama miktarı gerektirirken bu kodlarda bozulan bir düğümün tamiri daha az düğüme bağlanılarak gerçekleştirebilmektedir. İki farklı türde *Pyramid* kodu mevcuttur: Temel *Pyramid* kodları ve Genelleştirilmiş *Pyramid* kodları. Temel *Pyramid* kodlarında orijinal veri parçaları kendi arasında gruplara ayrılır. Her bir gruptaki veri ayrı ayrı MDS kodlama ile kodlanır. Yani, tüm kodlama sisteminde yerel kod parçaları oluşturulur. Ayrıca, tüm veri de MDS kodlamasından geçer, ancak kodlanmış veri parçası sayısı yerel kodların varlığı sebebiyle orijinal MDS kodlamadan daha az sayıdadır. Genelleştirilmiş *Pyramid* kodlarında ise gruplar için ayrı ayrı kodlama gerçekleştirilirken (yatay ek veri) gruplar arasında da kodlama işlemi gerçekleştirilmektedir (dikey ek veri). Bu kodların kodlama matrisi, bir arızalanan düğümün minimum sayıda düğüme bağlanılarak tamir edilmesini sağlayacak şekilde oluşturulmaktadır. Genelleştirilmiş *Pyramid* kodları, Temel *Pyramid* kodlarının arızlanmış bir düğüm tamirini gerçekleştiremediği düğüm arızalanma senaryolarında bu düğümü tamir edebilir. Temel *Pyramid* kodları çeşitli MDS kodlarından oluşturulabilirken genelleştirilmiş *Pyramid* kodları tamamen yeni bir silinti kodlama çeşitidir.

Bu tezde önerilen HMSR kodları, HSRC (Oggier and Datta, 2011b) ve e-MSR (Rashmi et al., 2011b) kodlarının avantajlarını birleştirecek şekilde oluşturulan melez bir kodlama şemasıdır. HSRC kodlama şeması bir önceki bölümde (alt bölüm 2.2.1'de) detaylı bir şekilde anlatılmıştı. Aşağıdaki alt bölümlerde e-MSR kodlama şeması (Rashmi et al., 2011b), bu kodlama şemasında düğüm tamiri ve verinin geri çatılması işlemleri detaylı bir şekilde anlatılmaktadır.

2.3.1 e-MSR kodları

$[n,\ k,\ d]$ e-MSR (Rashmi et al., 2011b) kodlama şemasında, $d \geq 2k-2$ olma koşulu bulunmaktadır. Bu kodlama şemasının α ve β değerleri Eşitlik (2.2) ve Eşitlik (2.3)'ü sağlamaktadır.

$d = 2k-2$ için MSR noktasında (Eşitlik (2.2)) $\alpha = d-k+1 = k-1$'dir. Yani $d = 2\alpha$'dır. Toplamda kodlanacak sembol sayısı ise $B = k\alpha = \alpha(\alpha+1)$'dir. Ayrıca mesaj matrisi M ise aşağıdaki şekildedir:

$$M = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \tag{2.6}$$

Burada S_1 ve S_2 matrisleri $(\alpha \times \alpha)$ boyutunda simetrik matrislerdir. Bu matrislerin üst sağ üçgen tarafı $\binom{\alpha+1}{2}$ tane orijinal veri sembolü ile doldurulurken alt üçgen kısımları bu sembollerin yansıması ile doldurulmaktadır. Böylece $B = \alpha(\alpha+1)$'dir. Bu kodlama şemasının kodlama matrisi de $\Psi = \begin{bmatrix} \Phi & \Lambda\Phi \end{bmatrix}$'dır. Φ $(n \times \alpha)$'lık bir matris, Λ ise $(n \times n)$'lik bir köşegen matrisidir. Ψ'nin elemanları şu koşulları sağlamalıdır: Ψ'nin herhangi d satırı doğrusal bağımsız olmalıdır, Φ'nin herhangi α satırı doğrusal bağımsız olmalıdır ve Λ'nın köşegen elemanları farklı olmalıdır (Rashmi et al., 2011b).

e-MSR kodlarında kodlama işlemi $C = \Psi M$ şeklinde gerçekleştirilmektedir. C matrisinin her bir satırı farklı bir düğüme gönderilmektedir. Yani sonuç olarak her düğümde α sembol depolanmaktadır. Bu koşulları sağlayabilmesi için Ψ matrisi, elemanları özel bir kritere göre seçilen bir Vandermonde (Turner, 1966) matrisi olarak oluşturulabilir.

2.3.1.1 e-MSR kodlama şemasında düğüm tamiri

e-MSR kodlarında arızalanmış bir düğüm $d = 2k-2$ tane yardımcı düğüme bağlanılarak tamir edilebilmektedir. $\begin{bmatrix} \phi_f^t & \lambda_f\phi_f^t \end{bmatrix}$ satırı arızalanmış düğümün belirtecine denk gelen kodlama matrisi satırı olsun. Arızalanmış düğümün depoladığı kodlanmış veri de $\begin{bmatrix} \phi_f^t & \lambda_f\phi_f^t \end{bmatrix} M = \phi_f^t S_1 + \lambda_f\phi_f^t S_2$ 'dir (Rashmi et al., 2011b).

Arızalan f düğümünün yerine gelen düğüm, çalışır durumdaki herhangi d düğüme bağlanır. Bu yardımcı düğümler $\{h_j | j = 1, \ldots, d\}$ olsun. Yeni-gelen düğüm bu düğümlere bağlandığında bu düğümlerin her biri $\psi_{h_j} M\phi_f$'yi hesaplayarak yeni-gelen düğüme gönderir. Bu yardımcı düğümlerin kodlama matrisi satırlarından oluşturulan matris ψ_{tamir} olarak adlandırılmıştır:

$$\psi_{tamir} = \begin{bmatrix} \psi_{h_1} \\ \psi_{h_2} \\ \vdots \\ \psi_{h_d} \end{bmatrix} \tag{2.7}$$

Yeni-gelen düğüm her yardımcı düğümden ilgili çarpım sonucunu aldıktan sonra $\psi_{tamir} M\phi_f$'yi elde etmiş olur. ψ_{tamir} matrisi tersinir olduğu için yeni-gelen düğüm

$$M\phi_f = \begin{bmatrix} S_1\phi_f \\ S_2\phi_f \end{bmatrix} \tag{2.8}$$

matrisini elde edebilir. S_1 ve S_2 matrisleri simetrik olduğu için $\phi_f^t S_1$ ve $\phi_f^t S_2$ kolaylıkla iki alt matrisin devriğini alma yolu ile hesaplanır ve bunlar kullanılarak arızalanmış düğümün içeriği yani $\begin{bmatrix} \phi_f^t & \lambda_f^t \phi_f \end{bmatrix} M = \phi_f^t S_1 + \lambda_f \phi_f^t S_2$ hesaplanır.

2.3.1.2 e-MSR kodlama şemasında verinin geri çatılması

e-MSR kodlama şemasında (Rashmi et al., 2011b), çalışır durumdaki herhangi k düğüme bağlanılarak orijinal veri geri çatılabilir. $\Psi_{VT} = \begin{bmatrix} \Phi_{VT} & \Lambda_{VT}\Phi_{VT} \end{bmatrix}$ verinin geri çatılması için bağlanılmış olan düğümlerin ilgili kodlama matrisi satırları ile oluşturulmuş bir matris olsun. Veri-toplayıcı düğüm bağlandığı düğümlerin hepsinden tüm kodlanmış verilerini indirdiğinde aşağıdaki matrise sahip olur:

$$\Psi_{VT} M = \begin{bmatrix} \Phi_{VT} & \Lambda_{VT}\Phi_{VT} \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \begin{bmatrix} \Phi_{VT}S_1 + \Lambda_{VT}\Phi_{VT}S_2 \end{bmatrix} \tag{2.9}$$

Veri-toplayıcı düğüm (2.9) eşitliğini sağ taraftan Φ_{VT}^T ile çarparak aşağıdaki eşitliği elde eder.

$$\begin{bmatrix} \Phi_{VT}S_1 + \Lambda_{VT}\Phi_{VT}S_2 \end{bmatrix} \Phi_{VT}^T = \Phi_{VT}S_1\Phi_{VT}^T + \Lambda_{VT}\Phi_{VT}S_2\Phi_{VT}^T \tag{2.10}$$

P ve Q matrisleri aşağıdaki şekilde tanımlanmıştır.

$$P = \Phi_{VT}S_1\Phi_{VT}^T \tag{2.11}$$

$$Q = \Phi_{VT}S_2\Phi_{VT}^T \tag{2.12}$$

S_1 ve S_2 simetrik olduğu için, P ve Q matrisleri de simetriktir. Veri-toplayıcı düğümün sahip olduğu matris P ve Q cinsinden aşağıda ifade edilmiştir:

$$P + \Lambda_{VT} Q \tag{2.13}$$

Yukarıdaki matrisin $(i,\ j)$'inci elemanı $(1 \leq i, j \leq k)$

$$P_{ij} + \lambda_i Q_{ij} \tag{2.14}$$

dir. Bu matrisin $(j,\ i)$'inci elemanı ise

$$P_{ji} + \lambda_j Q_{ji} = P_{ij} + \lambda_j Q_{ij} \tag{2.15}$$

dir. λ_i'lerin hepsi e-MSR kodlama matrisinin yapısından dolayı birbirinden farklıdır. Böylelikle, P_{ij} ve Q_{ij}'ler iki bilinmeyenli iki denklemin çözülmesi yolu ile elde edilebilir ($i \neq j$ için). İlk olarak P'nin elde edilmesi düşünülürse, Φ_{VT} aşağıda verilmiştir:

$$\begin{bmatrix} \phi_1^t \\ \vdots \\ \phi_{\alpha+1}^t \end{bmatrix} \tag{2.16}$$

P'nin tüm köşegen olmayan elemanları çözülmüştür. Köşegen elemanlar için yani $i = j$ için $P_{ii} + \lambda_j Q_{ii} = P_{ii} + \lambda_j Q_{ii}$ olduğundan bu noktalarda sadece bir denklem bulunmaktadır. P'nin i'inci satırı aşağıda gösterilmiştir:

$$\phi_i^t S_1 \begin{bmatrix} \phi_1 & \dots & \phi_{i-1} & \phi_{i+1} & \dots & \phi_{\alpha+1} \end{bmatrix} \tag{2.17}$$

Yukarıdaki eşitlikte sağ taraftaki matris tersinir olduğu için, veri-toplayıcı düğüm aşağıdaki matrisi hesaplayabilir:

$$\{\phi_i^t S_1 | 1 \leq i \leq \alpha + 1\} \tag{2.18}$$

Yukarıdaki elemanların ilk α tanesini seçerek, veri-toplayıcı düğüm aşağıdaki matrise erişebilir:

$$\begin{bmatrix} \phi_1^t \\ \vdots \\ \phi_{\alpha}^t \end{bmatrix} S_1 \tag{2.19}$$

Yukarıda, sol taraftaki matris tersinir olduğu için veri-toplayıcı düğüm S_1 matrisini elde eder. Veri-toplayıcı düğüm benzer şekilde Q'nün elemanlarını çözerek S_2'yi hesaplar.

2.4 Kümeleme Tabanlı Dağıtık Depolama Sistemi İle İlgili Literatür Özeti

Bu alt bölümde kümeleme tabanlı dağıtık depolama sistemleri ile ilgili literatür özeti bulunmaktadır. Literatürde dağıtık depolama sistemleri için genel olarak tek bir kodlama şemasının işleyişi incelenmiştir. Ancak, tek bir kodlama şemasının kullanılması farklı özelliklere sahip düğümleri bulunan bir sistem için verimli olmayabilir. Bu nedenle, ağda farklı özelliklere sahip olan düğümlerin kümelendirilerek bu kümelere uygun farklı kodlama şemalarının kullanılması ile harcanan kaynakların maliyetinin ne şekilde değişeceği bu çalışmada incelenmiştir. Bu çalışma kapsamında kümelerde kullanılmak üzere üç farklı kodlama şeması belirlenmiştir. Bu kodlama şemaları: HSRC (Oggier and Datta, 2011b), e-MBR (Rashmi et al., 2011b) ve HMBR (Haytaoglu and Dalkilic, 2013a) kodlama şemalarıdır. HSRC kodlama şeması düğüm tamirinde az sayıda düğüme bağlandığından daha düşük I/O bekleme süresi gerektirmektedir. Ayrıca, bu kodlama şemasında yardımcı düğümlerin yerel olarak gerçekleştirmesi gereken bir matematiksel işlem bulunmamaktadır. Ancak, bu kodlama şemasında her yardımcı düğüm grubunun arızalanan bir düğümü tamir edebilme olanağı yoktur. e-MBR kodlarında ise düğüm tamiri işlemi için bağlanılması gereken düğüm sayısı HSRC kodlama şemasına göre daha fazla olup her bir düğümden indirilmesi gereken sembol sayısı azalmaktadır. Ancak bu kodlama şemasında depolanması gereken ek veri miktarı artmaktadır. HMBR kodlama şeması ise hem HSRC hem de e-MBR düğüm tamiri yöntemlerini birarada bulundurmaktadır. Bu kodlama şemaları sırasıyla, alt bölüm 2.2.1, alt bölüm 2.2.2 ve alt bölüm 3.2'de ayrıntılı bir şekilde anlatılmıştır.

Rashmi et al. (2011a) depolama sistemini oluşturan düğümlerin iki kümeye ayrıldığı ve bu düğümlerin hepsinin farklı bir kodlama şemasını kullandığı bir çatı önermişlerdir. Bu çatıda bir mesaj önce C_0 tipinde bir kodlama şeması ile kodlanır ve kodlanmış sonuçlar ilk kümedeki düğümlerde depolanır. Ardından bu mesajın devriği alınır ve bu hali C_1 tipinde bir kodlama şeması kullanılarak kodlanır. Bir düğümün tamiri diğer kümedeki düğümlere bağlanılarak gerçekleştirilir. Verinin geri çatılması işlemi ise aynı kümedeki düğümlerden kodlanmış sembollerin indirilmesiyle gerçekleştirilmektedir. Bu şekilde, önerilen çatı ile iki kodlama şemasının da güçlü yanlarından faydalanılabileceği vurgulanmıştır. Bu çalışma, tez çalışmamızdan farklı olarak aynı veriyi iki farklı kodlama şeması ile kodlamaktadır. Yani kodlanacak bir veri için iki farklı kodlama şeması kullanıldığından ek depolama alanı gerekmektedir. Tez kapsamında önerilen çalışmada ise farklı verilerin farklı kodlama şemaları ile kodlandığı durum incelenmiştir.

Huang et al. (2014) da farklı hata dayanıklılığı seviyeleri sağlamak için farklı uygulamalarda farklı kodlama şemalarının kullanılabileceğini önermişlerdir. Bunu sağlamak için *Kesişen Zigzag Kümeleri* kodları (Tamo et al., 2013) kullanılmıştır. Tez kapsamında gerçekleştirilen çalışmada düğüm tamirinde ve verinin geri çatılmasında kullanılan süre ve bant genişliği iyileştirilmeye çalışılırken, Huang et al. (2014) tarafından önerilen çalışmada farklı hata dayanıklılığı seviyeleri oluşturulmaya çalışılmıştır.

Literatürde kodlama şemalarının performans analizlerinin gerçek ya da benzetim ortamında gerçekleştirildiği az sayıda çalışma bulunmaktadır. Bu bölümün ilerleyen kısımlarında kümeleme tabanlı dağıtık depolama sistemine benzer şekilde kodlama şemalarının uygulamalarının gerçekleştirilerek performans analizinin yapıldığı çeşitli çalışmalar incelenmiştir.

Pamies-Juarez et al. (2012) tarafından gerçekleştirilen çalışmada, yenileme kodları (*Regenerating Codes*) (Dimakis et al., 2007) ve SRC (*Self Repairing Codes*) kodları (Oggier and Datta, 2011b) gibi yeni nesil kodlama şemalarıyla geleneksel silinti kodlama şemasının (Reed and Solomon, 1960) düğüm tamiri performansları karşılaştırılmıştır. Geleneksel silinti kodları, işbirlikçi yenileme kodları (Shum, 2011), SRC ve boru hattı (*pipeline*) yöntemini destekleyen SRC kodlarında, kodlanmış nesnelerin düğümlere dağıtılma stratejilerinin ve kodlanmış nesne büyüklüklerinin düğüm tamirinde bant genişiliği kullanımı ve düğüm tamiri işleminin hızı üzerindeki etkileri incelenmiştir. Düğüm tamiri performansı incelenirken tek bir düğüm tamiri ve çoklu düğüm tamiri işlemleri için farklı senaryolar incelenmiştir. Ayrıca nesnelerin düğümlere dağıtılması işleminde kümeleme yaklaşımı kullanılmıştır. Ancak, bu tez kapsamındaki çalışmadan farklı olarak kümeleme tek bir kodlama şeması kullanılarak gerçekleştirilmiştir. Bir ağda farklı kümeler bulunurken her küme birbiriyle aynı kodlama şemasını kullanmaktadır. Bu çalışmaya göre, tek bir düğümün tamirinde en az bant genişliği harcamasını yenileme kodları yapmaktadır. SRC kodları ise, tek bir düğüm tamiri için yenileme kodlarından daha çok ama geleneksel silinti kodlarından daha az bant genişliği harcaması gerçekleştirmektedir. Ancak çoklu düğüm arızalanmaları yaşandığında ve arızalanmış düğüm sayısı çok fazla olduğunda (tüm nesnenin geri çatılması söz konusu olduğunda) geleneksel silinti kodları diğer kodlama şemalarına göre daha iyi sonuç vermektedir. Düğüm tamiri süresinde ise en yüksek performansı sağlayan boru hattı yöntemi kullanılan SRC kodları olmuştur. Bu çalışmada bahsedilen, boru hattı kullanılan SRC kodları SRCp (SRC-*pipelined*) olarak adlandırılmıştır.

Pamies-Juarez et al. (2013a), orijinal verilerin salt kopyalama kullanılarak ye-

deklendiği bir sistemde, bu verilerin arşivsel veri haline dönüşmesi durumunda, bu verileri yedekleme yönteminin salt kopyalamadan silinti kodlamaya çevrilmesi işlemini incelenmişlerdir. Arşivsel veriye dönüşen bu verilerin silinti kodlama ile kodlama işleminde, birden fazla düğümde bulunan kopyalarından faydalanılarak boru hattı yaklaşımı ile kodlanması önerilmiştir. Bu kapsamda, boru hattı yaklaşımı ile kodlama işlemini gerçekleştiren *RapidRAID* kodları sunulmuştur. Bu kodların kodlama performansı geleneksel silinti kodları ile karşılaştırılmış ve yeterli ön bellek (*cache*) boyutlarında büyük ölçüde hızlanma sağladığı gösterilmiştir.

Rashmi et al. (2014), Reed-Solomon (Reed and Solomon, 1960) kodlama şemasında düğüm tamiri işlemi için yüksek ölçüde bant genişliği ve I/O bekleme süresi gereksinimi olduğunu belirtmişlerdir. Reed-Solomon kodlama şeması çatısı üzerine Piggybacking (Rashmi et al., 2013b) çatısı kullanılarak üç farklı depolama sistemi önermişlerdir. Bu yeni tasarlanmış sistemler düğüm tamirinde daha az bant genişliği ve daha az I/O kullanımı gerektirmektedir. Bu yeni depolama sistemlerinin bu işlemlerdeki performansı Facebook tarafından kullanılan iki farklı HDFS (Hadoop Distributed File System) (Shvachko et al., 2010) kümesi üzerinde test edilmiştir. Bu testlerde yeni sistemlerin düğüm tamiri işleminde kullandığı bant genişliği ve I/O kullanımının azaldığı görülürken, orijinal verinin kodlanması için gereken sürenin ise arttığı görülmüştür.

2.5 Silinti Kodlarında Dosya Geri Çatma İşlemi İçin TCP Bağlantılarının Yönetimi İle İlgili Literatür Özeti

TCP *Incast* problemi, çoktan bire modeline sahip bir ağ iletişiminde; birden fazla TCP bağlantısının paketlerini ileten bir ağ anahtarının (*switch*) tampon bellek kapasitesinin, hedefi aynı olan çoklu TCP bağlantılarının paketlerini tamponlayabileceği kapasiteden daha az olması durumunda meydana gelen, TCP bağlantılarındaki veri aktarım veriminin düşmesi problemidir. TCP *Incast* problemi için veri hattı (bağlantısı) katmanı, taşıma katmanı ve uygulama katmanı seviyesinde çeşitli çözüm önerileri bulunmaktadır.

Veri hattı katmanındaki çözüm önerileri genel olarak akış kontrolü ve tıkanıklık kontrolü olarak ikiye ayrılmıştır. Phanishayee et al. (2008) aşırı yüklenmiş ağ anahtarlarının tıkanıklık bulunan arayüzüne bağlı aygıtlardan gelen veri gönderimlerinin durdurulması için bu aygıtlara "durakla" mesajının gönderilmesini önermişlerdir. Böylece ağ anahtarlarının aşırı yüklenmiş kuyruklarının rahatlatılması hedeflenmiştir. Bu yöntem, tüm istemci ve sunucuların tek bir ağ anahtarına bağlı olduğu du-

rumlar için avantajlı olup birden çok ağ anahtarının bulunduğu durumlar için önemli bir avantaj sağlamamaktadır. Bu nedenle farklı aygıtlardan gelen veri gönderimlerinde aktarım hızlarını kısıtlayan bir Ethernet modeli tasarımı da ortaya konmuştur (Wadekar, 2007). Alizadeh et al. (2008) tarafından veri merkezi ağlarında veri hattı katmanında tıkanıklık kontrolünü sağlayan bir tıkanıklık belirleme algoritması önerilmiştir. Bu algoritma iki aşamadan oluşmaktadır. İlk aşamada tıkanıklık oluşumu ve tıkanıklığın boyutu belirlenir. İkinci aşamada, elde edilen veriye göre gönderici kendi gönderim hızını ayarlar. Shpiner et al. (2012), veri merkezlerinde TCP iletişim veriminin azalması durumunda ağ anahtarı seviyesinde kullanılması düşünülmüş özet fonksiyonu (*hash*) tabanlı ve öncelik kuyruklarının kullanıldığı bir algoritma önermişlerdir. Bu algoritma varolan TCP uygulamalarının üzerinde herhangi bir değişiklik yapmamakta böylelikle TCP bağlantılarının iki ucundaki düğümlerin fazladan bir işlem yapmasına gerek duyulmamaktadır. Ancak bu çalışmada da, veri merkezlerinde kullanılan hata toleransı yönteminin gerektirdiği özel durumlar incelenmemiştir.

Taşıma katmanındaki öneriler kendi içerisinde TCP parametrelerini değiştiren ya da bu parametreleri değiştirmeyen olmak üzere ikiye ayrılmaktadır. TCP parametrelerini değiştiren önerilerin biri yeniden gönderim süresini azaltmaktır. TCP yeniden gönderim zamanlayıcısı esasen WAN'lara uygun olarak 200 ms'ye ayarlanmıştır. Ancak bu değer veri merkezi ağları için yüksek bir değerdir. Yeniden gönderim süresinin yüksek olması büyük miktarlarda veri aktarımları yapan veri merkezleri için boşa harcanan bir zaman oluşturabilmektedir. Phanishayee et al. (2008) bu nedenle yeniden gönderim süresini düşürmüş ve gerçekleştirdikleri benzetimlerde bu süreyi düşürmenin veri aktarım verimini arttırdığını gözlemlemişlerdir. TCP'nin gecikmeli ACK paketini engellemek TCP *Incast* problemini azaltmak için önerilen diğer bir yaklaşımdır. Vasudevan et al. (2009) tarafından önerilen çalışmada gecikmeli ACK paketini engellemeyle elde edilen veri aktarım veriminin daha yüksek olduğu gösterilmiştir.

Veri merkezleri için yeni TCP protokolleri de önerilmiştir. Bunlardan bir tanesi Data Center TCP'dir (Alizadeh et al., 2010). Data Center TCP'nin amacı, küçük kapasiteli tampon belleğe sahip ağ anahtarları bulunan veri merkezleri için düşük gecikmeli, veri aktarım verimi yüksek ve bir anda yüksek boyutta oluşan ağ trafiğini kaldırabilecek bir taşıma kontrol protokolü sağlayabilmektir. Bu protokolde tıkanıklık sorununun boyutu tahminlenmeye çalışılır ve buna göre tıkanıklık penceresinin boyutu ayarlanır. TCP *Incast* tıkanıklık kontrolünde (Wu et al., 2013) ise veri aktarım veriminin arttırılması için alıcı tarafında TCP alıcı penceresi ayarlanarak paket kayıpları oluşmadan önce engellenmeye çalışılmıştır.

Incast problemi için uygulama katmanında önerilen çözümler arasında veri gönderiminin kısıtlanması da bulunmaktadır. Krevat et al. (2007) tarafından önerilen çalışmada TCP *Incast* problemi açıklanmış ve bu problemi gidermek için uygulama düzeyinde çözümler önerilmiştir. Önerilen çözümler arasında bir sunucudan istenen dosya parçalarının boyutlarını arttırma, bir dosya için eş zamanlı istekte bulunulan sunucu sayısını azaltma, birebir bağlantılar arasındaki veri akışını kısma, sunucuların istemci isteklerini yerine getirmeyi geciktirmeleri ve veri akışlarının merkezi bir otorite tarafından planlanması bulunmaktadır. Ancak bu problemin silinti kodlama işleminin toplam süresi üzerindeki etkisi ile ilgili bir yorum bulunmamaktadır. Phanishayee et al. (2008) her bir sunucudan istenen veri blok boyutunun arttırılmasını önermişlerdir. Bu şekilde bir istemci daha az sunucuya bağlanacak ve ağda daha az sayıda TCP bağlantısı olacaktır. Ancak, bir dosyanın sınırlı sayıda sunucuya dağıtılması hata toleranslı sistemler için uygun olmayabilir.

Bunun dışında, daha yüksek kapasiteye sahip tampon bellekli ağ anahtarı tasarımı (Phanishayee et al., 2008) ve olasılıksal yeniden gönderim yaklaşımı (Kulkarni and Agrawal, 2011) gibi öneriler de TCP *Incast* problemine yönelik çözümler arasında bulunmaktadır.

Cho and Aguilera (2012) ise farklı konumlarda bulunan veri merkezleri arasında tıkanıklık meydana gelmesi sonucunda oluşabilecek uzun ağ gecikmelerini önleyebilecek bir sistem önermişlerdir. Ancak bu sistemde veriler salt kopyalama ile depolanmaktadır ve farklı kopyaların tutarlı olması üzerinde çalışılmıştır.

3. MATERYAL VE YÖNTEM

Bu bölümde, doktora tezi kapsamında gerçekleştirilen çalışmaların içeriği ve önerilen yöntemler beş ayrı alt bölümde anlatılmıştır. Bu beş çalışmadan ilki MDS silinti kodlarını kullanan dağıtık depolama sistemlerinde verinin geri çatılması ve düğüm tamirinin başlatılması ve veri güncelleme işlemlerinde kullanılan ağ trafiğinin azaltılması ve sürenin düşürülmesi için önerilen topoloji farkındalıklı bir çalışmadır. İkincisi Homomorfik Minimum Bant Genişliği Tamir (HMBR) kodlarıdır. Üçüncü çalışma Homomorfik Minimum Depolama Tamir (HMSR) kodları üzerinedir. Dördüncü çalışmada ise bir kümeleme tabanlı dağıtık depolama sistemi tasarlanmıştır. Beşinci ve sonuncu çalışma ise silinti kodlarını kullanan dağıtık depolama sistemlerinde ağda tıkanıklık oluştuğunda verinin geri çatılması işleminin hızlandırılması için yeni bir yöntem önerilmesidir.

3.1 MDS Silinti Kodlama Kullanan Dağıtık Depolama Sistemleri İçin Topoloji Farkındalıklı Bir Çalışma

Bu alt bölümde MDS silinti kodlama kullanan dağıtık depolama sistemleri için topoloji farkındalıklı bir çalışma ile ilgili yapılan ön hazırlık ve bu konuda tez kapsamında önerilen yaklaşımlar anlatılmıştır. Tanımlar ve sistem modeli bölümünün altında Minimum Steiner Ağaç probleminin tanımı ve silinti kodlarının çalışma mekanizması anlatılmıştır. Ayrıca önerilen yaklaşımların uygulanacağı sistemin modeli ayrıntılı bir şekilde açıklanmıştır. Önerilen yöntemler bölümünde verinin geri çatılmasının ve düğüm tamirinin başlatılması için önerilen iki farklı yaklaşım olan Steiner Ağaç Yaklaşımı ve Çoklu En Kısa Yol Yaklaşımı açıklanmıştır. Yine aynı bölümde verinin güncellenmesi işleminde iletişimi sağlamak için önerilen Steiner Ağaç Yaklaşımı ve Çoklu En Kısa Yol Yaklaşımı açıklanmıştır.

3.1.1 Tanımlar ve sistem modeli

Bu bölümde sırasıyla minimum Steiner Ağacı problemi, silinti kodlarının dağıtık depolama sistemlerinde nasıl kullanıldığı ve önerilen yaklaşımların uygulanacağı sistem modeli anlatılacaktır.

3.1.1.1 Minimum Steiner ağacı problemi

Çizge teorisinin bilinen problemlerinden biri de NP-tam olan Minimum Steiner Ağacı problemidir (Karp, 1972). Bu problem şu şekilde tanımlanmıştır: $G(V, E)$ bağlı bir çizge olsun ve S de bu çizgedeki köşelerin bir alt kümesi olsun yani $S \subseteq V$. Ayrıca bu çizgedeki her kenar için bir maliyet fonksiyonu $c(v_i, v_j)$ olsun ($v_i,\ v_j \in$

V). Verilen bir çizgenin maliyeti o çizgedeki tüm kenarların maliyetinin toplamı olarak belirlenmiştir. Bir $G(V, E)$ çizgesi için Steiner Ağaç Problemi S köşe kümesi için minimum maliyete sahip bir $T(V', E')$ ağacının bulunmasıdır, öyle ki bu ağacın köşeleri V', V'nin alt kümesidir ($V' \subseteq V$) ve $E' \subseteq E$'dir (Takahashi and Matsuyama, 1980).

3.1.1.2 Dağıtık depolama sistemleri için silinti kodları

Bu bölümde silinti kodlama kullanan tipik bir dağıtık depolama sistemi kısaca açıklanmıştır. Geleneksel (n, k) silinti kodlamada (Reed and Solomon, 1960), orijinal veri önce k eşit parçaya bölünür. Sonra bu parçalar $(n \times k)$'lık bir kodlama (üretici) matrisi ile çarpılır. Böylece matris çarpımı sonucunda $(n \times 1)$'lik bir matris oluşur. Böylece $(k \times 1)$'lik orijinal veri $(n \times 1)$'lik bir kod matrisine genişletilmiş olur. Kodlama matrisinin her bir satırı farklı bir düğüme gönderilir. Toplamda n tane düğüm bulunmaktadır. Bu şekilde oluşturulan bir dağıtık depolama sistemi toplamda $(n - k)$'ya kadar düğüm arızalanmalarını tolere edebilmektedir.

Orijinal verinin geri çatılması yani yeniden oluşturulması gerektiğinde, bu işlemi gerçekleştirecek düğümün, sistemdeki n düğümün herhangi k tanesinden depoladığı verileri indirmesi gerekmektedir. Bu düğüme veri-toplayıcı düğüm adı verilmektedir. Veri-toplayıcı bir düğüm bağlandığı düğümlerin ilgili kodlama matrisi satırlarından oluşan $(k \times k)$'lık bir alt matris oluşturur. Ardından veri-toplayıcı düğüm bu alt matrisin tersi ile diğer k düğümden indirdiği $(k \times 1)$'lik matrisi çarparak $(k \times 1)$'lik orijinal veri matrisini elde eder.

Bir dağıtık depolama sisteminde herhangi bir düğüm arızalandığında verinin geri çatılabilmesi olasılığının sabit tutulması için arızalanan düğümün tamir edilmesi gerekmektedir. Bir düğüm arızalandığı takdirde sisteme yeni bir düğüm eklenmekte ve bu düğüm verinin geri çatılması işleminde olduğu gibi, çalışır durumdaki herhangi k düğümden o düğümlerin depoladığı kod parçalarını indirmektedir. Bu işlemi yapan düğüme yeni-gelen düğüm denilmektedir. Yeni-gelen düğüm veri-toplayıcı düğümün yaptığı gibi $(k \times 1)$'lik orijinal veriyi geri çatar, geri çattığı orijinal veriyi yeniden kodlama matrisi ile kodlar. Ancak yeni-gelen düğüm sadece yerini aldığı arızalanmış düğümün depoladığı kod parçasını depolar.

Silinti kodları sistematik ya da fonksiyonel olarak sınıflandırılmaktadır. Kod eğer sistematik ise kodlama matrisinin ilk k satırı bir birim matrisi oluşturmaktadır. Yani kodlama sonucunda elde edilen genişletilmiş matrisin ilk k satırı orijinal veri matrisine denk gelmektedir. Kodlanmış veri matrisinin ilk k satırından herhangi

birini depolayan düğüme de sistematik düğüm denilmektedir. Sistematik kod kullanılan bir sistemde, eğer k sistematik düğümün hepsi çalışır durumda ise, verinin geri çatılması işlemi bu k sistematik düğümde depolanan verilerin indirilmesi ile gerçekleştirilebilir. Yani geri çatma işlemi için matris çarpımına ihtiyaç duyulmamaktadır. Fonksiyonel kodlarda ise kodlama matirisinin içerisinde birim matrisi bulunmamaktadır. Bu kodlarda verinin geri çatılması işleminde matris çarpımı işlemine ihtiyaç duyulmaktadır.

Dağıtık depolama sistemlerinde bozulan bir düğümün biran önce tamir edilmesi çalışkan strateji olarak adlandırılmaktadır. Ancak bazı sistemlerde, düğüm bozulma sayısının bir eşik değerine erişmesi beklenip sonrasında tamir işlemine başlanmaktadır. Buna tembel strateji denilmektedir.

MDS silinti kodlarında güncelleme işlemi için ilk önce güncellenecek orijinal veri matrisinin, yeni hali ile şimdiki durumu arasındaki fark hesaplanmalıdır. Bu fark Galois sonlu cisim aritmetiği ile hesaplanır ve bu fark Δ (delta) olarak adlandırılmaktadır. Ardından bu Δ değeri, kodlanmış veriyi tutan $(n-k)$ tane düğüme gönderilmektedir (Burada sistematik kodun kullanıldığı düşünülmüştür). Δ değerini alan düğümler, Δ'yı kendi kodlama matrisi satırının ilgili elemanı ile çarpar, sonra bu sonucu kendi depoladıkları kodlanmış veri matrisi parçası ile Galois cismi üzerinden toplar. Dağıtık depolama sisteminde güncelleme işlemi, $n-k$ düğüm bu işlemi gerçekleştirdiğinde tamamlanmaktadır.

3.1.1.3 Benzetim için sistem modeli

Dağıtık depolama sistemini oluşturan bir bilgisayar ağı, bilgisayarları temsil eden köşeleri ve bilgisayarlar arasındaki bağlantıları temsil edecek kenarlara sahip olan bir çizge olarak temsil edilebilir. Bu çizgedeki kenar ağırlıkları bilgisayarlar arasındaki iletişim maliyetlerini temsil etmektedir. Bu çizgede bazı düğümler depolama düğümleri, diğer düğümler ise istemci düğümler olarak atanmıştır. İstemci düğümler dağıtık depolama sistemlerinin depolama, veri çekme ve güncelleme gibi servislerini kullanmaktadır. Ağda n tane depolama düğümü bulunmaktadır ve bunlardan herhangi k tanesinde depolanan veriler kullanılarak orijinal veri geri çatılabilmektedir. Ağda toplamda N tane düğümün olduğu, yani toplamda $N-n$ tane istemci düğümün olduğu varsayılmaktadır. Bu sistemde, MDS silinti kodlarının özellikleri kullanılarak düğüm tamiri, verinin geri çatılması ve güncelleme işlemleri gerçekleştirilebilmektedir.

Görevdeş (P2P) ağlar günümüzde çeşitli uygulamalarda kullanılabilmektedir,

bunlardan bir tanesi ise veri depolama uygulamalarıdır (Gupta and Awasthi, 2011). Dağıtık depolama sistemlerini oluşturan ağlar, adanmış veri merkezleri ya da özdeş ağlar tarafından kullanılıyor olabilir. MDS kodlarında kullanılan n ve k parametreleri, üzerinde çalışılan ağın ve uygulamanın özelliklerine göre değişmektedir. Örneğin, özdeş ağlarda n ve k değerleri oldukça yüksek iken bu değerler adanmış veri merkezlerinde oldukça düşüktür ($10 - 20$ arası).

3.1.2 Önerilen yöntemler

Aşağıdaki iki alt bölümde verinin geri çatılmasının ve düğüm tamirinin başlatılması ve verinin güncellenmesi işlemleri için önerilen iki farklı iletişim yaklaşımı anlatılmaktadır.

3.1.2.1 Verinin geri çatılmasının ve düğüm tamirinin başlatımı

Verinin geri çatılmasının ve düğüm tamirinin başlatılması işlemlerinde kullanılması gereken birden çoğa iletişim modeli (*multicast*) için iki farklı yaklaşım denenmiştir. Bunlardan ilki Minimum Steiner Ağaç için tasarlanmış bir yaklaşım algoritmasını kullanmaktadır, ikincisi ise Dijkstra'nın en kısa yol algoritmasını kullanmaktadır. Verinin geri çatılması ve düğüm tamiri başlatım işlemlerinin iletişim gereksinimleri aynıdır. Bu yüzden, aşağıda önerilen yaklaşımlar bu işlemlerin ikisi için de aynıdır. Bu bölümde verilen şekillerde kullanılan sembollerin açıklaması Çizelge 3.1'de verilmiştir.

Çizelge 3.1. Şekil 3.1 ve Şekil 3.2'de Kullanılan Sembollerin Anlamları.

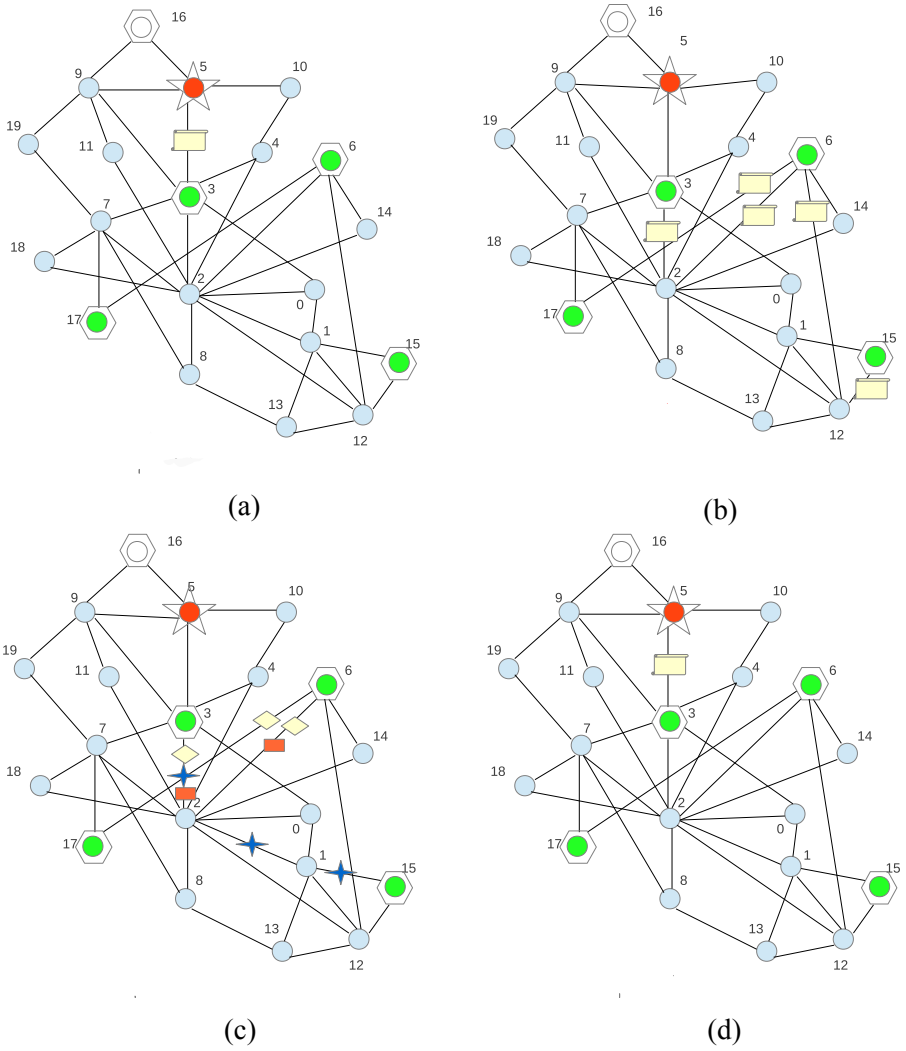
Sembol	**Anlamı**
	Depolama düğümü
	Verinin geri çatılmasını, düğüm tamirini ya da güncelleme işlemini tetikleyen istemci
	İstemci düğüm
	Mesaj
	Seçilen k düğümün içerisinde bulunmayan depolama düğümü
	3. ve 17. düğüm arasındaki mesaj
	3. ve 6. düğüm arasındaki mesaj
	3. ve 15. düğüm arasındaki mesaj

Steiner Ağaç Yaklaşımı (STA) Sistemdeki bilgisayarlar istemci düğüm ve depolama düğümü olmak üzere iki farklı tipe ayrıldıkları için birden çoğa iletişiminin maliyetini düşürme problemi Minimum Steiner Ağaç bulma problemi ile benzeşmektedir. Çünkü Minimum Steiner Ağacı Probleminde hedefte olmayan düğümler de iletişim sürecinde kullanılabilmektedir. STA'da ise depolama düğümleri arasındaki iletişim maliyetini düşürmek için istemci düğümler depolama düğümleri arasındaki iletişime dahil edilebilmektedir.

Bir depolama düğümü herhangi bir veri geri çatımı ya da düğüm tamiri isteği aldığında, bu düğüm bir vekil sunucu gibi işlemle ilgili istek mesajını yeterli sayıda depolama düğümüne iletir. Bu tipteki iletişim modeli birden çoğa iletişim modeli için çok uygundur, çünkü aynı mesajın birden çok düğüme gönderilmesi gerekmektedir. Ayrıca, birden çoğa iletişimde kullanılması gereken iletişim ağacı (Dolagh and Moazzami, 2011) tarafından geliştirilmiş olan Steiner Ağacı yaklaşım algoritması ile oluşturulabilir. Bu yaklaşıma Steiner Ağaç yaklaşımı (STA) denilmektedir.

STA'da, bir deplama düğümü herhangi bir istemci düğümden bir veri geri çatımı ya da düğüm tamiri isteği aldığında, bu depolama düğümü k tane farklı depolama düğümü seçer. Bu depolama düğümü başlatıcı düğüm olarak adlandırılmıştır. Ardından, bu başlatıcı düğüm Dolagh and Moazzami (2011) tarafından önerilen algoritmayı kullanarak kendisi ve seçtiği k farklı depolama düğümü arasında eğer bu düğümler arasında hazırda bir Steiner ağaç yoksa bir Steiner ağacı oluşturur. Sonra, bu düğüm oluşturulan Steiner ağacındaki yaprak düğümlere veri geri çatımı/düğüm tamiri isteği gönderir. Steiner ağaçtaki herhangi bir düğüm bu istek mesajını aldığında eğer yaprak düğüm değilse ağaçtaki komşularına, mesaj aldığı düğüm hariç, bu isteği iletir. Eğer bu düğüm aynı zamanda k depolama düğümünden biri ise kendisinde bulunan veri parçalarını da başlatıcıya gönderir. Steiner ağacının oluşturulması işlemi, veri geri çatımı/düğüm tamiri isteği ilk defa geldiğinde ya da hazırda oluşturulmuş bir ağaçta bulunan bir düğüm bozulduğunda gerçekleştirilmektedir. Sonuç olarak, eğer Steiner ağacı sıfırdan kurulmuşsa gerekli ağaç bilgisi de ağaçtaki düğümlere gönderilmelidir. Diğer durumlarda, başlatıcı düğüm önceden oluşturulmuş olan ağacı kullanabilmektedir.

Bu yaklaşımı kullanan örnek bir senaryo Şekil 3.1'de verilmiştir. Şekil 3.1 (a)'da 5. düğüm 3. düğüme veri geri çatımı/düğüm tamiri isteği göndermektedir, ardından üçüncü düğüm rastgele seçtiği $k = 3$ düğüm ile kendisi arasında 2, 3, 6,12 15 ve 17 düğümlerini içeren bir Steiner ağacı oluşturmaktadır (Şekil 3.1 (b)). Şekil 3.1 (c)'de ağaçtaki depolama düğümleri ilgili veri parçalarını başlatıcı düğüme geri göndermektedir, Şekil 3.1 (d)'de ise 3. düğüm aldığı kodlanmış veri parçaları



Şekil 3.1. STA kullanılarak gerçekleştirilen verinin geri çatımı ya da düğüm tamiri işlemi a) 5. düğüm 3. düğümden istekte bulunuyor, b) 3. düğüm diğer düğümlere isteği iletiyor, c) 3. düğüm paketleri topluyor, d) 3. düğüm istenilen işlemi yerine getirip sonucu 5. düğüme gönderiyor.

ile veriyi geri çatmakta (ya da bozulan düğümü tamir etmekte) ve istemci olan 5. düğüme sonuç veriyi göndermektedir.

Çoklu En Kısa Yol Yaklaşımı (MPA) Verinin geri çatılmasının ya da düğüm tamirinin başlatılması işlemleri için uygulanan ikinci yaklaşım çoklu teke gönderim (*muti unicast*) yaklaşımıdır. Bu yaklaşım çoklu en kısa yol yaklaşımı olarak (*Multi-shortest Path Approach* - MPA) adlandırılmıştır. Aynı alt-ağa ait olmayan bir grup düğüm için belirli bir birden çoğa gönderim çatısı yoksa, birden çoğa gönderim iletişimi için tek tek gönderim yöntemi uygulanmaktadır. Bu yüzden, birden çoğa gönderim için eğer hiçbir iletişim çatısı tanımlanmamışsa, bir depolama düğümü veri geri çatımı ya da düğüm tamiri isteklerini ilgili depolama düğümlerine tek tek gönder-

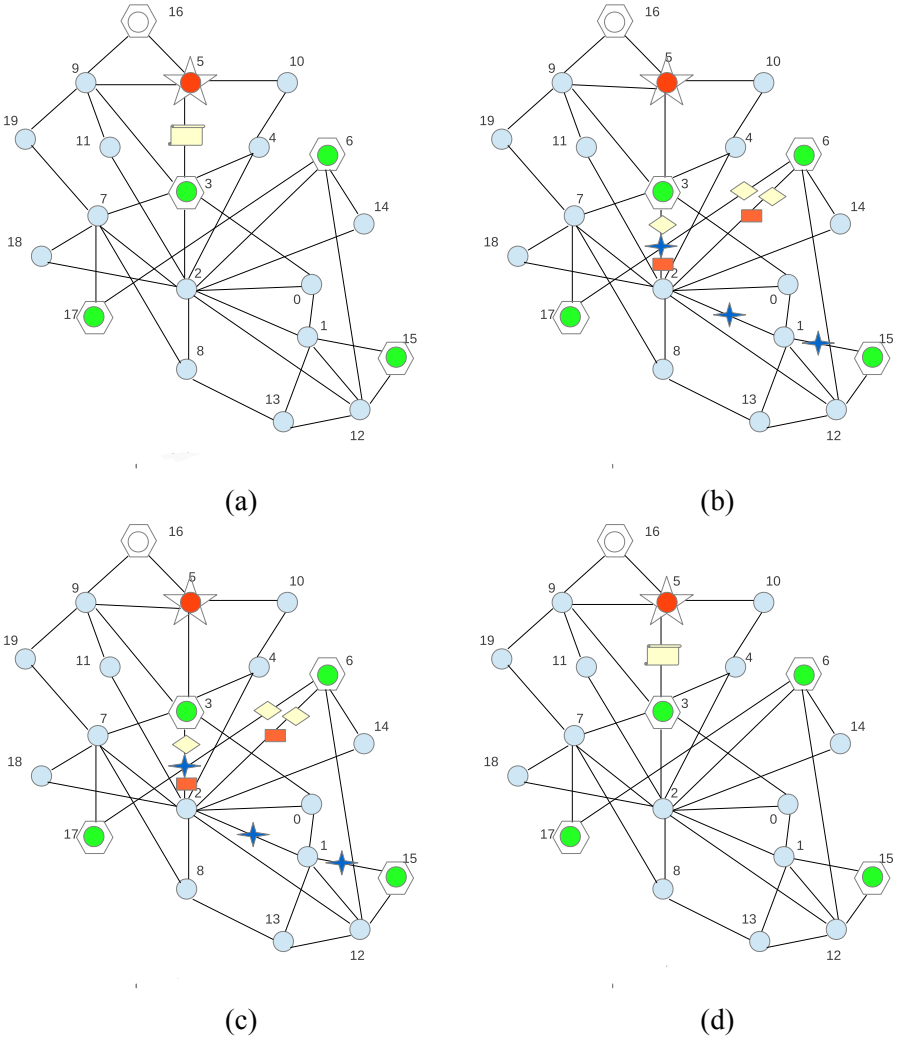
melidir. Her bir teke gönderim işlemi Dijkstra'nın en kısa yol algoritması ile optimal bir şekilde uygulanabilir (Dijkstra, 1959). Başlatıcı düğüm bir veri geri çatımı ya da düğüm tamiri isteği alır almaz bu isteği en kısa yollar üzerinden rastgele seçtiği k farklı depolama düğümüne gönderir. En kısa yol üzerindeki bir düğüm, istek mesajını aldığında bu mesajı hedefle arasında bulunan en kısa yol üzerindeki komşusuna iletir. Bu yaklaşımda isteğin ilk defa başlatıcı düğüme geldiği anda, başlatıcı düğüm en kısa yolları üretir ve ilgili yol bilgilerini en kısa yoldaki düğümlere iletir. Daha önceden oluşturulmuş kısa yollar üzerinde bulunan bir düğümün arızalanması söz konusuysa, bu en kısa yol da yeniden oluşturulur.

Bu yaklaşımı kullanan örnek bir senaryo Şekil 3.2'de gösterilmektedir. Şekil 3.2 (a)'da 5. düğüm (istemci düğüm) 3. düğüme veri geri çatımı ya da düğüm tamiri isteğinde bulunmuştur. Üçüncü düğüm Dijkstra'nın algoritmasını kullanarak rastgele seçtiği k tane depolama düğümü ile kendi arasında en kısa yolları oluşturur ve isteği bu yollar üzerinden birer birer bu depolama düğümlerine gönderir (Şekil 3.2 (b)). Şekil 3.2 (c)'de isteği alan depolama düğümleri kendilerinde depolanan veri bloklarını başlatıcı düğüme gönderir. Ardından 3. düğüm ilgili işlemi gerçekleştirir ve sonucu 5. düğüme gönderir (Şekil 3.2 (d)).

3.1.2.2 Veri güncelleme işlemi

$D = \{D_1, D_2, \ldots, D_k\}$ orijinal veri vektörü olsun. $\{C_1, C_2, \ldots, C_{n-k}\}$ ise D'nin kodlanmış parçalarını tutan kod vektörü olsun. D_i, D_i' olarak değiştirilmek istendiğinde, D_i'nin kodlanmış halini içeren C_j değerleri ($j \in \{1, \ldots, n-k\}$) bu güncelleme işleminden etkilenmektedir. Güncelleme işleminden önceki D_i değeri ile güncelleme işlemi gerçekleştikten sonraki değer arasındaki fark $\Delta = D_i' - D_i$ olsun. Güncelleme işlemi sonrasında C_j'nin yeni değeri olan C_j', $C_j' = C_j - G_{ji}\Delta$ olarak hesaplanmaktadır. Burada G_{ji} kodlama matrisinin j'inci satırının i'inci sütununda bulunan elemandır ve tüm aritmetik işlemler Galois sonlu cismi üzerinden gerçekleştirilmektedir. Diğer bir deyişle, güncelleme işlemi için ilk başta Δ değeri hesaplanmaktadır adından bu Δ değeri her j için C_j'yi depolayan düğüme gönderilmektedir. Ardından, Δ değerini alan her düğüm kendi etkilenmiş verisini $G_{ji}\Delta$ değerini kullanarak hesaplamaktadır. Δ değerinin tüm j düğümlerine gönderilmesi gerektiği için bu iletişim yöntemi birden çoğa iletişim modelini gerektirmektedir. MDS silinti kodlarında güncelleme işleminin daha detaylı açıklaması (Plank, 1997)'de mevcuttur.

İlerleyen alt bölümlerde MDS silinti kodlarının veri güncelleme işleminin iletişiminde kullanılması önerilen Steiner ağaç ve çoklu en kısa yol yaklaşımları anla-



Şekil 3.2. MPA kullanılarak gerçekleştirilen bir verinin geri çatımı ya da düğüm tamiri işlemi a) 5. düğüm 3. düğümden istekte bulunuyor, b) 3. düğüm diğer düğümlere isteği iletiyor, c) 3. düğüm paketleri topluyor, d) 3. düğüm istenilen işlemi yerine getirip sonucu 5. düğüme gönderiyor.

tılmıştır.

Steiner Ağaç Yaklaşımı (STA) Bir istemci veri güncelleme isteğini herhangi bir depolama düğümü aracılığıyla gerçekleştirebilir. Bu depolama düğümü kaynak düğüm olarak adlandırılmıştır. Bu yaklaşımda, hazırda bir Steiner ağaç yoksa güncellemeden etkilenen $(n-k)$ tane depolama düğümünün bulunduğu (ve bu düğümler dışında ara düğümlerin bulunduğu) bir Steiner ağaç Dolagh and Moazzami (2011) tarafından önerilen algoritma kullanılarak oluşturulmaktadır. Ardından, ağaç bilgileri $(n-k)$ depolama düğümüne iletilmektedir. Güncellenen parça D'nin i'inci parçası olsun. Kaynak düğüm, parçanın olması istenen değerinden mevcut değerini Galois cismi üzerinden çıkartarak Δ değerini hesaplar. Kaynak düğüm bu Δ değerini oluşturduğu Steiner ağacı üzerinden ilgili hedeflere gönderir. Eğer Δ değerinin büyüklüğü MTU'dan (Maksimum İletim Biriminden) büyük ise bu değer MTU'ya göre düzenlenerek gönderilmektedir. Δ değerini alan hedef düğümü j, kodlama matrisinin ilgili satırını (G_j) da depolamaktadır ve kendi depoladığı veriyi, bu veriden $G_{ji}\Delta$ değerini çıkartarak güncellemektedir.

Çoklu En Kısa Yol Yaklaşımı (MPA) D_i parçası için bir güncelleme isteği geldiğinde, istemde bulunulan depolama düğümü $\Delta = D_i - D_i^{'}$ değerini hesaplar. Eğer Δ değerinin büyüklüğü MTU'dan daha fazla ise, Δ paketlere bölünür. Ayrıca, en kısa yollar önceden hesaplanmamışsa, kaynak düğüm $n-k$ farklı hedef düğümle arasındaki en kısa yolları Dijkstra Algoritması ile hesaplamaktadır. Ardından, kaynak düğüm Δ'yı hedef düğümlerin her birine en kısa yollar üzerinden ayrı ayrı gönderir. MPA yaklaşımında birden çoğa gönderim modelinde herhangi bir çatı bulunmadığı için, Δ, hedeflere teker teker gönderilmektedir. Hedef düğüm j, tüm paketleri aldıktan sonra, yeni değeri $C_j^{'} = C_j - G_{ji}\Delta$ işlemi ile hesaplayarak depolar. Tüm hedef düğümler bu işlemi gerçekleştirdikten sonra güncelleme işlemi tamamlanmış olur.

3.2 HMBR (Homomorfik Minimum Bant Genişliği Tamir Kodları) Kodları

Günümüzde sayısal ortamlarda depolanan veri hacmi giderek artmaktadır. Dağıtık depolama sistemlerini oluşturan depolama düğümleri, disk hataları, ağ arızalanmaları gibi çeşitli nedenlerden dolayı zaman zaman çalışamaz veya erişilemez duruma gelmektedir. Bu arızalanmalar nedeniyle depolama düğümlerinin düşük maliyetle işler durumda tutulması önemli bir problemdir. Bununla ilgili olarak literatürde, değişik maliyet düşürme politikalarına sahip bir çok depolama tekniği mev-

cuttur. Bu tekniklerden en yalın olanı, verilerin ikinci bir kopyasının ayrı bir yerde tutulduğu salt-kopyalama yöntemidir.

Literatürde çeşitli dağıtık depolama sistemlerinen bahsedilmiştir. Bulut tabanlı depolama sistemi bunların en yenilerinden bir tanesidir. Bulut tabanlı depolama sistemleri kullanıcılarına verilerini depolayabilme ve yönetebilme imkanı sunar (Lee et al., 2013). Yaygın olarak kullanılan çeşitli bulut depolama sistemleri bulunmaktadır, *Dropbox* (Dropbox, 2015), *Windows Azure Storage* (WAS) (Calder et al., 2011) ve *Google Drive* (Google, 2015) bunların arasındadır. Bu sistemlerin her biri kendisine özel bir yedekleme yöntemi kullanmaktadır. Örneğin WAS 3-yönlü kopyalama yöntemini kullanmaktadır (Huang et al., 2012) (Yani her verinin üç kopyası farklı düğümlerde saklanmaktadır.).

Yedeklemede kullanılan salt-kopyalama yönteminin temel eksikliği depolama maliyeti açısından verimsiz olmasıdır. Değişik bir çözüm yöntemi olan, silinti kodlama yöntemi çok yüksek veri dayanıklılığına sahip olup aynı zamanda düşük depolama alanına ihtiyaç duymaktadır. Burada düğüm arızalanmalarına karşı orijinal verinin kaybı ne kadar düşük olasılıklarda gerçekleşiyorsa eldeki kod o kadar dayanıklıdır. Günümüzde birçok sistem silinti kodlama yöntemini kullanmaktadır. Çeşitli RAID tabanlı depolama sistemleri arasından RAID-6, yedeklemeyi sağlamak için Reed-Solomon kodlarını kullanmaktadır (Thomasian, 2005). Ek olarak, OceanStore (Kubiatowicz et al., 2000) ve TotalRecall (Bhagwan et al., 2004) depolama sistemleri de bu yöntemi kullanmaktadır. Ancak, geleneksel silinti kodları (Reed and Solomon, 1960), bir düğüm tamiri için tüm orijinal verinin büyüklüğü kadar bant genişliği kullanımına ihtiyaç duymaktadır.

Bant genişliği, I/O ve hesaplama karmaşıklığı maliyetleri dağıtık depolama sistemlerinin tasarım eğilimlerini etkileyen temel parametreleri oluşturmaktadır. Bir sistemde, bir alandaki maliyetin düşük olması istenirken başka bir maliyetin çok büyük bir önemi olmayabilmektedir. Örneğin, bulut tabanlı sistemlerin düğüm tamiri işleminde I/O yükü, yardımcı düğüm sayısı ile doğru orantılı bir şekilde artmaktadır (Papailiopoulos and Dimakis, 2012). Böylece, düğüm tamirinde daha az yardımcı düğüme bağlanılan ve daha çok bant genişliği harcaması gerektiren kodlama şemaları bazı sistemler için verimli olurken (Gopalan et al., 2012), bant genişliğinin temel darboğaz olduğu sistemlerde, düğüm tamirinde daha çok yardımcı düğüme bağlanılan ve toplamda daha az bant genişliği gereksinimine sahip olan kodlama şemalarının kulllanılması daha uygun olabilmektedir. Ancak, kodlama şeması tasarlarken sadece bir tane maliyet parametresinin iyileştirilmesi bazı durumlarda çok mantıklı olmayabilir. Çünkü bazı sistemlerin ihtiyaçları ve bu ihtiyaçların doğrultusunda ma-

liyet öncelikleri zaman içerisinde değişebilmektedir. Eğer bir sistem belli bir zaman aralığında birçok istek alırsa, bu sistemin daha hızlı servis sağlaması için daha az I/O yükü olan ve daha az hesapsal karmaşıklık gerektiren bir kodlama şeması kullanması daha uygun olabilmektedir. Aynı sistem başka bir zaman aralığında yoğun bir iş yükü altında olmayabilir, bu durumda ise yedekleme sisteminin gerektirdiği temel işlemlerde düşük bant genişliği harcaması gerektiren bir kodlama şemasının kullanılması (yüksek hesapsal karmaşıklığı gerektirse bile) mantıklı olabilir.

Bu bölümde doktora tez çalışmasının ikinci ana konusu olan Homomorfik Minimum Bant Genişliği Tamir Kodları (HMBR) anlatılmaktadır. Literatürde düzinelerce kodlama şeması geliştirilmiş olup, bu kodlama şemaları farklı farklı maliyet parametrelerini (düğüm tamirinde kullanılan bant genişliği, düğüm tamirinde bağlanılan düğüm sayısı, dayanıklılık, depolama alanı kullanımı ... vb.) iyileştirmektedir. Bu kodlama şemaları incelendiğinde bu parametrelerin farklı ağırlıklarla öncelendirilerek sistemin ihtiyaçlarına göre ayarlanabilen bir kodlama şemasının eksikliği görülmüştür.

Depolama sistemlerinin ihtiyaçları, üzerinde kullanılacak uygulamaya göre değişmektedir. Birçok tasarım çeşidi bulunmaktadır, bunlardan bazıları düşük bant genişliği kullanımını önemserken bazıları ise düşük I/O maliyetini ya da düşük hesaplama karmaşıklığını diğer maliyet unsurlarına nazaran daha önde tutabilmektedir. Örneğin özdeş ağlar genellikle düşük bant genişliği tüketimini daha ön planda tutabilir. Arşivsel ve adanmış veri merkezleri ise genellikle düşük depolama maliyetine daha çok fazla önem vermektedir. Bulut depolama sistemleri ise daha az I/O yüküne sahip kodlama şemalarına ihtiyaç duyar (Örneğin: *Facebook Hadoop Cluster* (Wikipedia, 2015)). Burada toplam I/O maliyetini doğrudan etkileyen bir unsur ise düğüm tamirinde kullanılan yardımcı düğüm sayısıdır. Tamir işleminde bağlanılması gereken düğüm sayısı ise literatürde yerel tamir (*repair locality*) olarak adlandırılmaktadır.

Depolama sistemleri için bir kodlama şeması tasarlarken sadece bir parametreyi iyileştirip diğer parametrelerle ilgilenmemek çok anlamlı bir strateji olmayabilir. Bu nedenle, bu tez çalışmasında bir sistemin farklı zaman dilimlerindeki ihtiyaçlarına göre farklı parametrelerini iyileştirebilecek yeni bir melez kodlama şeması geliştirilmiştir. Bu şemada, düğüm tamiri işleminde e-MBR kodlama şemasının (Rashmi et al., 2011b) düşük bant genişliği kullanımından yararlanılırken, HSRC kodlama şemasının (Oggier and Datta, 2011b) da düşük I/O kullanımından yararlanılmaktadır. Tez kapsamında geliştirilen melez kodlama şeması daha büyük boyutlu bir sonlu cisme ihtiyaç duyabilmektedir. Bu da toplam olarak daha yüksek depolama

ihtiyacı gerektirebilir. Ancak, yeni geliştirilen melez kodlama şeması düğüm tamirinde farklı seçenekler sağladığı için bazı sistemler için toplam maliyet olarak diğer iki kodlama şemasının (e-MBR, HSRC) ayrı ayrı kullanılmasından daha verimli çalışabilmektedir. Bu melez kodlama şeması temel düzeyde bir bütünsellik kontrolü mekanizması da içermektedir. Bir düğüm tamir edilirken, yardımcı düğümlerden çekilen veriler çeşitli sebeplerle (kötü niyetli düğümler, bit hataları vb.) hatalı gelebilir. Böyle bir durumda bir bitin bile hatalı gelmesi sonuçta tamir edilen verinin büyük ölçüde bozulmasına yol açabilir. Yeni geliştirilen melez kodlama şeması ek bir bant genişliği harcaması yapmadan tamir edilen verinin bütünselliğini basit işlemlerle kontrol edebilmektedir. Ayrıca, önerilen kodlama şeması iki farklı verinin geri çatılması yöntemini de barındırmaktadır. Yöntemlerden bir tanesi diğerine göre daha düşük bant genişliği kullanımına sahipken diğer yöntem ise ilkine göre daha düşük hesaplama karmaşıklığına sahiptir.

Önerilen melez kodlama şemasında düğüm tamiri başarım olasığının kuramsal analizleri yapılmış olup bu analizlerin sonuçları benzetim sonuçları ile desteklenmiştir. Düğüm tamiri başarım olasılığı için elde edilen bulgularda HMBR kodlarının başarım olasılığının, e-MBR kodlarının başarım olasılığından az miktarda düşük, HSRC kodlarının başarım olasılığından ise daha yüksek seviyede olduğu görülmüştür.

Alt bölüm 3.2.1'de önerilen yeni kodlama şemasının oluşturulması, Alt bölüm 3.2.2'de önerilen kodlama şeması ile bozulan bir düğümün iki farklı yöntem ile tamir edilmesi anlatılmaktadır. Alt bölüm 3.2.3'te önerilen kodlama şemasında verinin geri çatılması yöntemleri incelenmiştir. Alt bölüm 3.2.4'te ise tamir edilen bir düğümün bütünlük kontrolünün nasıl yapıldığı anlatılmıştır. Son olarak, Alt bölüm 3.2.5'te yeni kodlama şemasının oluşturulması, düğüm tamiri yöntemleri, düğüm tamirinde bütünlük kontrolü ve verinin geri çatılması işlemleri bir örnek ile gösterilmiştir. Düğüm tamiri ve verinin geri çatılması işlemlerinde başarım olasılıkları ise 4. bölümde ele alınmıştır.

3.2.1 HMBR kodlama şemasının oluşturulması

Bir $[n, k, d]$ HMBR kodlama şemasında $B = \binom{k+1}{2} + k(d-k)$ tane sembol kodlanır ve her biri d tane sembol içeren n tane parça oluşturulur. Bu n parçadan, kodlama matrisi satırları doğrusal bağımsız olan d tanesi ya da özel bir koşulu sağlayan iki ya da daha fazlası arızalanmış bir düğümü tamir edebilmek için kullanılabilmektedir.

Bu kodlama şemasında HMBR mesaj matrisinin oluşturulması (Rashmi et al., 2011b)'deki e-MBR mesaj matrisinin oluşturulma basamakları ile aynıdır. $d \times d$ boyutlarındaki mesaj matrisi M olarak isimlendirilmiştir. Bu mesaj matrisi kendi içerisinde dört farklı matris yapısına sahip olup bu alt matrisler S, T, T^t ve sıfır (0) matrisidir. Kodlanacak dosyanın ilk $\binom{k+1}{2}$ tane sembolü S matrisinin üst-üçgen yarısına sırasıyla yerleştirilmektedir. S matrisinin geriye kalan alt-üçgen kısmı üst üçgen kısmının simetriği olacak şekilde bu sembollerle doldurulmaktadır. Kodlanacak dosyadan geriye kalan $k \times (d-k)$ mesaj sembolü ise T matrisine yerleştirilmektedir. T^t matrisi ise T matrisinin devriği şeklinde oluşturulmaktadır. M matrisinin geriye kalan $(n-k) \times (n-k)$ elemanı sıfır sembolü ile doldurulmaktadır. Sonuç olarak ortaya çıkan mesaj matrisi aşağıdaki şekildedir:

$$M = \begin{bmatrix} S & T \\ T^t & 0 \end{bmatrix} \tag{3.1}$$

HMBR kodlarında kodlama matrisinin oluşturulması işlemi ise e-MBR kodlarındaki (Rashmi et al., 2011b) kodlama matrisinin oluşturulması işleminden farklıdır. e-MBR kodlarının kodlama matrisi aşağıdaki koşulları sağlamalıdır:

- Kodlama matrisinin herhangi d satırı doğrusal bağımsızdır.
- Kodlama matrisinin herhangi k satırının ilk k sütününu kapsayan bir matrisin satırlarının doğrusal bağımsız olması gerekmektedir.

e-MBR kodlama şemasında bu ihtiyaçlar Cauchy (Bernstein, 2005) ya da Vandermonde (Turner, 1966) matrislerinin kullanımı ile karşılanmaktadır. Ancak yeni kodlama şemasında homomorfizm özelliğinden yararlanılması istendiği için, bu yeni şemanın kodlama matrisinin yapısı büyük ölçüde HSRC kodlama şemasının kodlama matrisinin yapısına benzemektedir. HMBR kodlarında kodlama matrisinin oluşturulması için öncelikle d adet farklı zayıf doğrusal polinom hazırlanır. Ardından bu polinomlar n tane farklı girdi ile değerlendirilir. Genel bir zayıf doğrusal polinomun yapısı şu şekildedir:

$$p_j(X) = \sum_{i=0}^{d-1} M_{ij} X^{2^i} \text{ ve } j \in \{0, 1, 2, ..., (d-1)\} \tag{3.2}$$

Kodlama matrisinin oluşturulması için önce $\mathbb{F}_{max(2^d,(2^{\lceil log_2^n \rceil}+1-(\lceil log_2^n \rceil-\lfloor log_2^n \rfloor)))}$ sonlu cisminden n tane farklı eleman seçilir. Ardından $p_j(x)$'ler $w_1, w_2, w_3, w_4, ..., w_n$ girdileri ile değerlendirilir. $p_j(x)$ polinomu, $w_z \mid z \in \{0, 1, ..., n-1\}$ girdileri ile değerlendirildiğinde tüm i'ler için M_j kodlanmış olur. Burada M_{ij}; M matrisinin i. satırının j. sütununu, M_j ise M matrisinin j'inci sütununu temsil etmektedir. d farklı $p_j(X)$, n tane farklı girdi ile değerlendirildiğinde, aslında HMBR kodlama şemasının kodlama işlemi gerçekleştirilmiş olur. Bu değerlendirme işlemi ise M matrisini aşağıdaki kodlama matrisi ile çarpma işlemine denk gelmektedir.

$$E = \begin{bmatrix} w_1 & w_1^2 & w_1^4 & ... & w_1^{2^{d-1}} \\ w_2 & w_2^2 & w_2^4 & ... & w_2^{2^{d-1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_n & w_n^2 & w_n^4 & ... & w_n^{2^{d-1}} \end{bmatrix} \tag{3.3}$$

e-MBR kodlama şemalarında olduğu gibi E matrisinin ilk $n \times k$'lık kısmı Φ ve sağ tarafta geriye kalan $n \times (d-k)$ sembollük kısmı ise Δ olarak adlandırılmıştır. HMBR kodlama şemasının kodlama işlemi aşağıdaki şekildedir:

$$C = \begin{bmatrix} w_1 & {w_1}^2 & {w_1}^4 & ... & {w_1}^{2^{d-1}} \\ w_2 & {w_2}^2 & {w_2}^4 & ... & {w_2}^{2^{d-1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_n & {w_n}^2 & {w_n}^4 & ... & {w_n}^{2^{d-1}} \end{bmatrix} \times \begin{bmatrix} S & T \\ T^t & 0 \end{bmatrix} \tag{3.4}$$

Diğer bir deyişle, $p_j(x)$'lerin n tane farklı girdi ile değerlendirilmesi işlemi E matrisinin M matrisi ile çarpılmasına eş değerdir. Bu kodlama işlemi gerçekleştirildikten sonra kod matrisi C'nin her bir satırı farklı bir düğüme gönderilmektedir. C matrisinin satırları $i \in \{0, \ldots, n-1\}$ arasındaki sayılarla numaralandırıldığında, i'nci satırın gönderildiği düğümün belirleyicisi de i'dir.

3.2.2 HMBR kodları ile arızalanan bir düğümün tamiri işlemi

HMBR kodlama şemasını kullanan bir düğüm iki farklı şekilde tamir edilebilmektedir. İlk yol e-MBR kodlama şemasındaki tamir işleminin bir uyarlaması olup bu yol $AMBR_{NR}$ olarak adlandırılmıştır. Diğer yol ise HSRC kodlama şemasındaki düğümün tamir edilmesi işleminin bir varyasyonu şeklinde gerçekleştirilmektedir ve $ASRC_{NR}$ olarak adlandırılmıştır.

3.2.2.1 AMBR$_{NR}$ yöntemi ile düğüm tamiri

Bu yöntemde HMBR kodlama sistemine dahil olan bir düğüm arızalandığında, yerine geçen düğüm d farklı yardımcı düğümden birer tane sembol indirmektedir. Yalnız, seçilen yardımcı düğümlerin kodlama matrisi satırlarının doğrusal bağımsız olması gerekmektedir.

$d \times d$'lik E_{yrd} matrisi, E kodlama matrisinin ilgili yardımcı düğümlerin belirleyicilerine denk gelen satırların bulunduğu bir alt matris olsun.

Teorem 3.1. *E_{yrd} matrisi tersinir bir matris ise, bozulan bir düğüm $AMBR_{NR}$ yolu ile toplamda d sembol indirilerek tamir edilebilir.*

İspat. *Arızalanan düğümün belirleyicisi f olsun ve bu düğüme karşılık gelen kodlama matrisinin satırı E_f^t olarak isimlendirilsin. Herhangi bir i düğümünde depolanan kod satırı C_i ile gösterilsin. Yeni-gelen düğüm, düğüm$_x$, düğüm$_y$, ..., düğüm$_z$ yardımcı düğümler ise bu düğümlerin kodlama matrisi satırları doğrusal bağımsız olacak şekilde d tane yardımcı düğüme ağ üzerinden bağlanır. Bağlanılan her yardımcı düğüm i, C_iE_f çarpımını hesaplar ve tek bir sembol içeren bu işlemin sonucunu yeni-gelen düğüme gönderir. Yeni-gelen düğüm, d yardımcı düğümden tüm sembolleri aldığında bu değerleri alt alta dizerek bir matris oluşturur. E_{yrd} ve M matrisleri ile E_f vektörünün çarpımından oluşan bu matris şu şekildedir:*

$$E_{yrd}ME_f \tag{3.5}$$

Bozulmuş olan düğümün içeriği $C_f = E_f^tM$ idi. E_{yrd} matrisi (3.5)'inci eşitlikten elenebilirse, ${E_{yrd}}^{-1}E_{yrd}ME_f$, ME_f değeri elde edilir. M matrisi yapım aşamasında simetrik oluşturulduğu için, $(ME_f)^t = E_f^tM$'dir. Yani bir önceki işlem ile

bozulmuş düğümün içeriği tamir edilmiş olur. Böylece, $AMBR_{NR}$ yolu ile toplamda d sembollük bir bant genişliği harcaması ile düğüm tamiri işlemi gerçekleştirilebilir.

3.2.2.2 ASRC$_{NR}$ yöntemi ile düğüm tamiri

Bu yöntem, HSRC (Oggier and Datta, 2011b) kodlarındaki düğüm tamir etme işleminin bir varyasyonudur. Bu yolla bir düğümün tamiri toplamda τd sembollük bir bant genişliği harcaması ile gerçekleştirilmektedir.

Teorem 3.2. *Bozulmuş bir düğümün içeriği yani C_f, τ $|\tau \in \{2,3,...,k\}$ tane düğüme bağlanılıp hepsinden d tane sembol indirilerek tamir edilebilmektedir.*

İspat. *Bu tamir etme şemasında, çalışır durumda olan τ tane özel düğümün bulunması gerekmektedir. Bozulmuş olan f düğümünü girdi değeri w_f olan $j|\{1,2,...,(d-1)\}$ olmak üzere d tane $p_j(x)$ polinomunun sonuç sembolünü depolamaktadır.*

$$w_f = w_a + w_b, \tau = 2 \tag{3.6}$$

$$w_f = \sum_{1}^{\tau} w_{yrd_i}, \tau > 2 \tag{3.7}$$

Öncelikle $\tau = 2$ olabildiği varsayılsın. Eğer sistemde kodlama satırlarının ilk elemanları Eşitlik (3.6)'yı sağlayan düğümler bulunuyorsa, yeni-gelen düğüm (tamirci) a ve b düğümlerine bağlanır ve bu düğümlerden ilgili tüm sembolleri indirir. Yeni-gelen düğüm ilgili tüm sembolleri aldığında, aslında aşağıdaki polinomlara sahip olmuş olur:

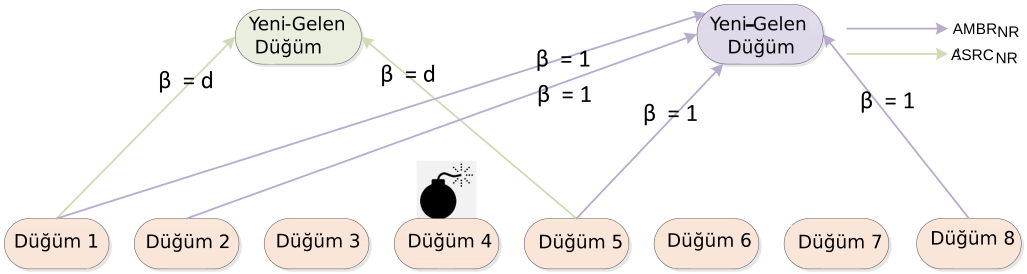
$$p_j(w_a) = \sum_{i=0}^{d-1} M_{ij} {w_a}^{2^i} \; j \in \{0,1,...,d-1\} \tag{3.8}$$

ve

$$p_j(w_b) = \sum_{i=0}^{d-1} M_{ij} {w_b}^{2^i} \; j \in \{0,1,...,d-1\}. \tag{3.9}$$

$C_f = E_f^t M$ ve $j \in \{0, 1, ...d-1\}$ değerleri için C_f'nin j. sembolü yani $E_f^t M_{*,j}$, $p_j(w_f)$ polinomuna denk gelmektedir. $w_f = w_a + w_b$ olduğu için homomorfizm özelliğinden $p_j(w_f) = p_j(w_a) + p_j(w_b)$'dir. $\tau \neq 2$ olduğu durumlarda (yani (3.6)'ıncı eşitliği sağlayan iki tane düğüm bulunamadıysa), (3.7)'inci eşitliği sağlayan minimum sayıda düğüm aranır. Sonuçta, bu yolla bozulmuş bir düğümü tamir etme işlemi tüm j ler için ($j \mid j \in \{0, 1, ..., d-1\}$) $p_j(w_f) = \sum_{i=1}^{\tau} p_j(w_{yrd_i})$ işlemini gerçekleştirmeyi gerektirmektedir.

$ASRC_{NR}$ yöntemi ile düğüm tamirinde, uygun özelliklere sahip τ tane düğümün her birinden d sembol indirilir ve ardından bu değerler sonlu cisim üzerinden toplanır. Böylece toplamda τd sembollük bir bant genişliği harcaması gerçekleştirilir. Şekil 3.3'te $AMBR_{NR}$ ve $ASRC_{NR}$ yöntemleri ile düğüm tamiri işlemlerindeki iletişim gereksinimleri β ile gösterilmektedir.



Şekil 3.3. HMBR kodlama şemasında 4 numaralı düğümün $AMBR_{NR}$ ve $ASRC_{NR}$ yöntemleri ile tamir edilmesi örnekleri.

3.2.3 HMBR kodları ile kodlanmış verinin geri çatılması

HMBR kodlarında verinin geri çatılması işlemi iki farklı şekilde yapılabilmektedir. Bunlardan ilki e-MBR kodlarındaki verinin geri çatılması işlemine benzemektedir ve $AMBR_{DR}$ olarak adlandırılmıştır. İkinci yöntem ise HSRC kodlarının geri çatma işleminin bir varyasyonu şeklinde gerçekleştirilmekte olup $ASRC_{DR}$ olarak adlandırılmıştır.

3.2.3.1 AMBR_{DR} yöntemi ile verinin geri çatılması

Bu yöntem e-MBR kodlarındaki verinin geri çatılması işlemine benzemektedir. Veri-toplayıcı düğüm kodlama matrisi satırları doğrusal bağımsız olan k tane çalışır durumdaki düğüme bağlanarak her birinden dosya ile ilgili kayıtlı tüm sembolleri indirir. Tüm düğümlerden indirme işlemlerinin tamamlanmasıyla, veri-toplayıcı düğümün elinde:

$$E_{VT}M = \begin{bmatrix} \Phi_{VT}S + \Delta_{VT}T^t & \Phi_{VT}T \end{bmatrix}. \tag{3.10}$$

matrisi bulunmaktadır.

Teorem 3.3. *Eğer veri-toplayıcı düğüm kodlama matrisinin satırları doğrusal bağımsız olan k tane düğüme bağlanıp ilgili tüm sembolleri indirebilirse, başlangıçta kodlanmış olan B orijinal veri sembolünü yeniden oluşturabilir.*

İspat. *Veri-toplayıcı düğüm kodlama matrisinin satırları doğrusal bağımsız olan k tane düğüm seçsin ve hepsinden ilgili tüm sembolleri indirerek Eşitlik (3.10)'daki sembollere sahip olsun. Bu düğümlerin ilgili kodlama matrisi satırları doğrusal bağımsız olduğundan veri-toplayıcı düğüm, bağlandığı k düğümün ilgili kodlama satırlarını kullanarak Φ_{VT}'yi oluşturur ve Φ_{VT}^{-1}'ni hesaplar. Ardından, veri-toplayıcı düğüm $\Phi_{VT}^{-1}\Phi_{VT}T$ işlemi ile T matrisini ortaya çıkarır. Veri-toplayıcı düğüm T değeri ile $\Delta_{VT}T^t$'ni hesaplar ve $\Phi_{VT}S + \Delta_{VT}T^t$'inden çıkartır. Sonunda veri-toplayıcı $\Phi_{VT}S$'e sahiptir, $\Phi_{VT}^{-1}\Phi_{VT}S$ ile S matrisini elde eder. Böylece ilk başta kodlanan B orijinal veri sembolünün tümü elde edilmiş olur.*

$AMBR_{DR}$ veri geri çatılması yönteminde indirilen toplam veri miktarı kd'dir.

3.2.3.2 ASRC$_{\text{DR}}$ yöntemi ile verinin geri çatılması

Bu veri geri çatılması yönteminde, aşağıda anlatıldığı gibi M matrisinin simetrik olmasından yararlanılmaktadır.

Teorem 3.4. *($ASRC_{DR}$) Veri-toplayıcı düğüm Φ matrisinde ilgili düğüm satırları doğrusal bağımsız olan k tane düğüm bulabilirse, toplamda B sembol bant genişliği harcamasıyla veriyi geri çatabilir.*

İspat. *$ASRC_{DR}$ yönteminde veri-toplayıcı düğüm verinin geri çatılmasına M matrisinin son sütunundan başlar. M matrisinin son sütunu $M_{*,d}$ olarak adlandırılmıştır. Ardından, M matrisinin diğer sütunlarını sağdan sola doğru sırasıyla geri çatar. T matrisi, 2^{k-1} derecesine sahip $d-k$ tane polinomun çeşitli noktalarda değerinin hesaplanması ile kodlanmaktadır. Diğer bir deyişle, T'yi kodlamada kullanılan $p_j(x)$ polinomu ($j \in \{k+1, k+2, \ldots, d\}$) k tane katsayı ile oluşturulmaktadır, çünkü T matrisinin k tane satırı bulunmaktadır. Eğer veri-toplayıcı düğüm Φ satırları doğrusal bağımsız olan k tane farklı düğümden $p_d(x)$ sonuçlarını indirebilirse,*

M matrisinin d'inci sütununa karşılık gelen $p_d(x)$ polinomunun katsayılarını geri çatabilir.

Veri-toplayıcı düğümün bu şekilde ilgili Φ matrisi satırları doğrusal bağımsız olan k farklı düğüm (düğüm$_1$, düğüm$_2$, . . ., düğüm$_k$) bulduğu varsayılsın. Veri-toplayıcı düğüm, bu düğümlere bağlanır ve her birinden $p_d(x)$ sonuçlarını indirir. Bu noktada, veri-toplayıcı düğüm aşağıdaki veriye sahip olur:

$$\Phi_d M_{*,d} = \begin{bmatrix} p_d(x_1) \\ p_d(x_2) \\ \vdots \\ p_d(x_k) \end{bmatrix} \tag{3.11}$$

Burada x_z, bağlanılan z düğümünün polinom girdisini temsil etmektedir. $j > k$ için $M_{*,j}$, T'nin $(j-k)$'ıncı sütununu temsil etmektedir. $j \leq k$ için ise $M_{1..j,j}$, M matrisinin j'inci sütununun ilk j satırını temsil etmektedir. $i > k$ için Φ_i, Φ_{VT}'dir, $i \leq k$ için ise Φ_i bağlanılan i düğümün kodlama matrisi satırlarının ilk i sütunundan oluşmaktadır:

$$\Phi_i = \begin{cases} \begin{bmatrix} x_1 & {x_1}^2 & \dots & {x_1}^{2^{k-1}} \\ x_2 & {x_2}^2 & \dots & {x_2}^{2^{k-1}} \\ \vdots & \vdots & \ddots & \vdots \\ x_k & {x_k}^2 & \dots & {x_k}^{2^{k-1}} \end{bmatrix}, d \geq i > k. \\ \\ \begin{bmatrix} x_1 & {x_1}^2 & \dots & {x_1}^{2^{i-1}} \\ x_2 & {x_2}^2 & \dots & {x_2}^{2^{i-1}} \\ \vdots & \vdots & \ddots & \vdots \\ x_i & {x_i}^2 & \dots & {x_i}^{2^{i-1}} \end{bmatrix}, 0 < i \leq k. \end{cases} \tag{3.12}$$

Veri-toplayıcı düğüm $M_{*,d}$'yi aşağıdaki şekilde geri çatar:

$$M_{*,d} = \Phi_d^{-1} \begin{bmatrix} p_d(x_1) \\ p_d(x_2) \\ \vdots \\ p_d(x_k) \end{bmatrix} \tag{3.13}$$

Benzer şekilde, $j > k$ için $M_{*,j}$ aşağıdaki gibi geri çatılır.

$$M_{*,j} = \Phi_j^{-1} \begin{bmatrix} p_j(x_1) \\ p_j(x_2) \\ \vdots \\ p_j(x_k) \end{bmatrix}, \quad k < j \leq d \tag{3.14}$$

Veri-toplayıcı düğümün $M_{1..k,k}$'yı geri çatmak için $M_{*,k}$'nın sıfırdan farklı elemanlarının sayısı d olmasına rağmen, Φ matrisi satırları doğrusal bağımsız olan d'den daha az sayıda yani k düğümden k tane polinom sonucu indirmesi yeterlidir. M'nin k'ıncı sütununda daha önceden çözülmüş $d-k$ tane eleman bulunmaktadır. Bu elemanlar aynı zamanda $p_k(x)$'nın kat sayıları arasındadır. Önceden çözülmüş $d-k$ tane elemanın bulunduğu terimlerin $p_k(x)$ polinomundan çıkarılması ile $p_k^{'}(x)$ polinomu oluşturulsun. Veri-toplayıcı düğüm indirdiği k farklı $p_k(x)$ sonucundan gerekli terimleri hesaplayıp çıkararak k farklı $p_k^{'}(x)$ sonucu oluşturabilir. Bu polinom indirgeme işlemleri diğer sütunların geri çatımı işlemleri için de yapılmaktadır. $l \leq k$ için $p_l(x)$ polinomunun $p_l^{'}(x)$ polinomuna indirgenmesi işlemi aşağıdaki gibi yapılmaktadır:

$$p_l^{'}(x) = p_l(x) - \sum_{z=l+1}^{d} M_{l,z}\, x^{2^{z-1}}, \qquad 1 \leq l \leq k \tag{3.15}$$

Veri-toplayıcı düğüm k tane $p_k^{'}(x)$ sonucunu hesapladıktan sonra M_k'yı aşağıdaki gibi geri çatar.

$$M_k = \Phi_k^{-1} \begin{bmatrix} p_k^{'}(x_1) \\ p_k^{'}(x_2) \\ \vdots \\ p_k^{'}(x_k) \end{bmatrix} \tag{3.16}$$

Veri-toplayıcı düğüm Φ_{k-1} satırları doğrusal bağımsız olan $k-1$ düğüme bağlanır ve bunlardan toplamda $(k-1)$ tane $p_{k-1}(x)$ sonucu indirir ve bu sonuçlarla $M_{*,k-1}$'in çözülmemiş elemanlarını geri çatar. $M_{*,(k-1)}$'in son $d-k+1$ elemanı önceki geri çatım işlemlerinde çözülmüştür. Veri-toplayıcı düğüm $k-1$ farklı $p'_{k-1}(x)$ sonucunu Eşitlik (3.15)'i kullanarak elde eder. Ardından, veri-toplayıcı düğüm $M_{(k-1)}$'i (3.17) eşitliğini kullanarak geri çatar. Benzer şekilde, $l \leq k$ için veri-toplayıcı düğüm Φ_l matrisi satırları doğrusal bağımsız olan l düğümden $p_l(x)$ sonuçlarını indirir ve l tane $p'_l(x)$ polinom sonucu hesaplar. Ardından, Eşitlik (3.17)'yi kullanarak $M_{*,l}$ sütununu geri çatar.

$$M_l = \Phi_l^{-1} \begin{bmatrix} p'_l(x_1) \\ p'_l(x_2) \\ \vdots \\ p'_l(x_l) \end{bmatrix}, \ 1 \leq l \leq k \tag{3.17}$$

Veri-toplayıcı düğüm Φ_d matrisi tersinir olan k tane düğüm bulabilirse, bu k düğümün $j \in \{1, 2, 3, \ldots, d-1\}$ için Φ_j matrisleri de Moore matrisi (Dickson, 1901) yapısından dolayı tersinirdir.

$ASRC_{DR}$ yöntemi ile verinin geri çatılmasında toplamda $B = \frac{k(k+1)}{2} + k(d-k)$ tane sembollük bant genişliği harcaması yapılmaktadır. Bu değer $AMBR_{DR}$ yöntemi için kd'dir. $k > 1$ için $ASRC_{DR}$ yöntemi $AMBR_{DR}$ yönteminden daha az bant genişliği kullanmaktadır. Ayrıca, $ASRC_{DR}$ yönteminde veri-toplayıcı düğüm, verinin geri çatılması işlemi için indirdiği B sembolü heterojen bir şekilde indirebilir. Örneğin, veri-toplayıcı düğüm kendisiyle arasındaki iletişim hattı çok yoğun olmayan bir düğümden yüksek miktarda veri indirebilirken, kendisiyle arasındaki iletişim hattının yoğun olarak kullanıldığı başka bir düğümden nispeten daha az veri indirebilir. Bu durum, ağdaki trafiğin daha homojen bir şekilde dağılmasını sağlayabilir.

Teorem 3.5. *Her iki kodlama şemasında kullanılan düğümlerin arızalanma olasılığı aynı ve her düğüme eş bir şekilde (uniform) dağılmışsa ve iki kodlama şeması da aynı sonlu alanı kullanıyorsa $[n, k \geq 2, d]$ HMBR kodlarında verinin geri çatılmasının başarım olasılığı $[n, k \geq 2]$ HSRC kodlarında verinin geri çatılması başarım olasılığı ile aynıdır.*[1]

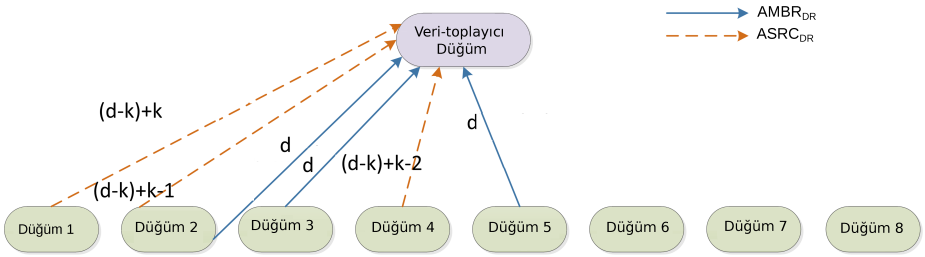
[1] İki kodlama şeması da aynı sonlu cismi kullanıyorsa $[n, k \geq 2, d]$ HMBR kodlama şeması $[n, k]$ HSRC kodlama şemasından daha yüksek miktarda veri kodlar ve depolar.

İspat. *İki kodlama şeması da katsayıları orijinal veri sembollerinden oluşan polinomlar kullanmaktadır. HMBR kodlarının kullandığı polinomlar, derecesi en fazla* 2^{k-1} *olan polinomlara indirgenebilmektedir.* 2^{k-1} *derecesine sahip bir polinom Lagrange interpolasyonu (Waring, 1779) kullanılarak* $2^{k-1}+1$ *nokta ile geri çatılabilmektedir. Eğer veri-toplayıcı düğüm* $\mathbb{F}_2$ *üzerinden doğrusal bağımsız* k *tane polinom sonucu bulabilirse, homomorfizm özelliği ile Oggier and Datta (2011b) tarafından bahsedildiği gibi* 2^k-1 *tane farklı polinom noktası elde edebilir.* $2^{k-1}+1$ *tane farklı nokta,* x *girdileri* $\mathbb{F}_2$ *üzerinden doğrusal bağımsız olan* k *tane* $(x, p_j(x))$ *noktasından elde edilebilir.* $R(x, d, r)$ *(Oggier and Datta, 2011b) fonksiyonu* $x \times d$ *boyutlu* r *kertesine sahip olan matris permütasyonlarının sayısını vermektedir* [1]*. HMBR kodlama şemasında en az* k *tane doğrusal bağımsız polinom girdisine sahip düğüm bulunduran çalışır durumdaki düğüm permütasyonları* $R(x, d, r)$ *fonsiyonu ile sayılabilir. Bu fonksiyon Bölüm 4'te detaylı bir şekilde anlatılacaktır.* $R(x, d, r)$ *fonksiyonu iki kodlama şeması için de aynı sonucu verir. Ek olarak, HMBR kodlama şemasında* d *tane farklı polinomun geri çatılması gerekmektedir. HSRC şemasında ise sadece bir tane polinomun geri çatılması gerekmektedir. HMBR kodlama şemasında, veri-toplayıcı düğüm* $p_d(x)$ *polinomunu* k *farklı düğümün* $(x, p_d(x))$ *noktaları ile geri çatabiliyorsa, bu düğüm geriye kalan* $d-1$ $p_j(x)$ *(*$j \in \{1, 3, ..., d-1\}$*) polinomlarını da aynı* k *düğümün* $(x, p_j(x))$ *noktaları ile geri çatabilir. Çünkü bağlanılan* k *düğümün her birinde depolanan* d *sembol, aynı polinom girdisi ile hesaplanmıştır. Sonuç olarak, polinom girdileri* $\mathbb{F}_2$ *üzerinden doğrusal bağımsız olan* k *düğüm bulunursa bu düğümlerde depolanan tüm semboller* d *tane polinomun geri çatılmasını garanti etmektedir. Böylece iki kodlama şeması da polinom girdileri* $\mathbb{F}_2$ *üzerinden doğrusal bağımsız olan* k *tane düğüm ile verinin geri çatılması işlemini garantilemiş olur.*

HMBR kodlama şemasının $AMBR_{DR}$ ve $ASRC_{DR}$ yöntemleri ile gerçekleştirilen verinin geri çatılması işlemlerinde ortaya çıkan bant genişliği gereksinimleri $[n = 8,\ k = 3,\ d = 4]$ için Şekil 3.4'te gösterilmektedir.

---

[1] Burada, d bir semboldeki bit sayısını temsil etmektedir.



Şekil 3.4. HMBR kodlama şemasında $AMBR_{DR}$ ve $ASRC_{DR}$ yöntemleri ile verinin geri çatılması işlemlerinin [n=8, k=3, d=4] için bant genişliği gereksinimleri.

3.2.4 HMBR kodlarında tamir edilen verinin bütünlüğünün doğrulanması

$AMBR_{NR}$ yolu ile tamir edilen düğümün içeriği C_f olsun, eğer yardımcı düğümler arasında $\sum_{i=1}^{\tau} w_{[yrd_i]} = w_{[f]}$ eşitliğini sağlayan τ tane düğüm bulunuyorsa, C_f verisi $E_f^t C_f^t = \sum_{i=1}^{\tau}(C_{yrd_i} E_f)$ eşitliğinin doğruluğu kontrol edilerek tamir edilebilir. $\sum_{i=1}^{\tau}(C_{yrd_i} E_f)$ toplamının içerisindeki her bir çarpımın sonucu yeni-gelen düğüm tarafından yardımcı düğümlerden indirilmektedir.)

$$\sum_{i=1}^{\tau} C_{yrd_i} E_f = \sum_{i=1}^{\tau} E_{yrd_i}^t M E_f \tag{3.18}$$

$$= (\sum_{i=1}^{\tau} E_{yrd_i}^t) M E_f = E_f^t M E_f = E_f^t C_f^t. \tag{3.19}$$

3.2.5 HMBR kodu örneği

Bu örnekte, $n = 8, k = 3$ ve $d = 4$ alınmış olup bu parametreler kullanıldığında $\alpha = d = 4$ ve kodlanacak sembol sayısı $B = 9$ olmaktadır. Sonlu cisim parametreleri olarak, $q = 4$ ve ilkel eleman $w = 2 \in \mathbb{F}_{2^q}$ olarak alınmıştır. S ve T matrisleri için ise aşağıdaki değerler kullanılmıştır:

$$S = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 3 & 4 \\ 2 & 4 & 5 \end{bmatrix} \text{ ve } T = \begin{bmatrix} 6 \\ 7 \\ 8 \end{bmatrix} \tag{3.20}$$

Kodlama matrisi E, mesaj matrisi M ve kod matrisi C aşağıdaki şekilde görülmektedir:

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 4 & 3 & 5 \\ 3 & 5 & 2 & 4 \\ 4 & 3 & 5 & 2 \\ 5 & 2 & 4 & 3 \\ 6 & 7 & 6 & 7 \\ 7 & 6 & 7 & 6 \\ 8 & 12 & 15 & 10 \end{bmatrix}}_{E} \times \underbrace{\begin{bmatrix} 0 & 1 & 2 & 6 \\ 1 & 3 & 4 & 7 \\ 2 & 4 & 5 & 8 \\ 6 & 7 & 8 & 0 \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} 5 & 1 & 11 & 9 \\ 15 & 10 & 6 & 8 \\ 10 & 11 & 13 & 1 \\ 5 & 8 & 5 & 12 \\ 0 & 9 & 14 & 5 \\ 10 & 2 & 3 & 4 \\ 15 & 3 & 8 & 13 \\ 8 & 5 & 15 & 6 \end{bmatrix}}_{C} \tag{3.21}$$

3.2.5.1 HMBR kodları için düğüm tamiri örneği

AMBR$_{NR}$ Yöntemi: Düğüm 4 arızalanmış olsun. Düğüm 4'ü tamir etmek için yeni-gelen düğüm olası düğüm kümelerinden, 1, 2, 5 ve 8 numaralı düğümlere bağlanır ve bu düğümlerden $C_i E_f^t | i = \{1, 2, 5, 8\}$ değerlerini indirir. Bu indirilen sembollerle yeni-gelen düğüm aşağıdaki vektöre sahiptir:

$$E_{yrd} M E_f = \begin{bmatrix} 4 \\ 10 \\ 1 \\ 3 \end{bmatrix} \tag{3.22}$$

Yeni-gelen düğüm ${E_{yrd}}^{-1}$'ı hesaplar ve $E_{yrd} M E_f$ matrisi ile çarpar. Bu işlemle yeni-gelen düğüm aşağıdaki matrisi elde etmiş olur:

$$ME_f = \begin{bmatrix} 5 \\ 8 \\ 5 \\ 12 \end{bmatrix} = E_f^t M \tag{3.23}$$

Bu vektörün devriği ise bozulmuş olan 4 numaralı düğümün içeriğidir.

ASRC$_{NR}$ Yöntemi: Yukarıda gösterilen tamir etme örneği $AMBR_{NR}$ yolu ile tamir etme idi. Şimdi de aynı düğüm $ASRC_{NR}$ yolu ile tamir edilecektir. Bu örnekte $w_{[4]} = w_{[1]} + w_{[5]}$ olduğu için yeni-gelen düğüm, düğüm 1'e ve düğüm 5'e bağlanarak bu düğümlerde depolanan ilgili sembolleri indirir. Bu işlemle yeni-gelen düğümün elinde aşağıdaki semboller bulunmaktadır:

$$p_0(w_{[1]}) = 5,\ p_1(w_{[1]}) = 1,\ p_2(w_{[1]}) = 11,\ p_3(w_{[1]}) = 9 \tag{3.24}$$

$$p_0(w_{[5]}) = 0,\ p_1(w_{[5]}) = 9,\ p_2(w_{[5]}) = 14,\ p_3(w_{[5]}) = 5 \tag{3.25}$$

Yeni-gelen düğüm bu polinomlarla aşağıdaki sonlu cisim toplama işlemlerini gerçekleştirir:

$$p_0(w_{[1]}) + p_0(w_{[5]}) = 5 \tag{3.26}$$

$$p_1(w_{[1]}) + p_1(w_{[5]}) = 8 \tag{3.27}$$

$$p_2(w_{[1]}) + p_2(w_{[5]}) = 5 \tag{3.28}$$

$$p_3(w_{[1]}) + p_3(w_{[5]}) = 12 \tag{3.29}$$

$$\begin{bmatrix} p_0(w_{[1]}) + p_0(w_{[5]}) & p_1(w_{[1]}) + p_1(w_{[5]}) & p_2(w_{[1]}) + p_2(w_{[5]}) & p_3(w_{[1]}) + p_3(w_{[5]}) \end{bmatrix} = \begin{bmatrix} 5 & 8 & 5 & 12 \end{bmatrix} \tag{3.30}$$

.

Eşitlik (3.30)'daki vektör düğüm 4'ün kaybolmuş içeriğidir.

3.2.5.2 HMBR kodlarında verinin geri çatılması örneği

AMBR$_{DR}$ Yöntemi Verinin geri çatılması işleminde veri-toplayıcı düğüm kodlama matrisi satırları doğrusal bağımsız k düğüme bağlanır. Veri-toplayıcı düğüm 2., 3. ve 5. düğümlere bağlanarak bunlardan ilgili sembolleri indirir ve aşağıdaki matrisi oluşturur:

$$\Phi_{VT}S + \Delta_{VT}T^t = \begin{bmatrix} 15 & 10 & 6 \\ 10 & 11 & 13 \\ 0 & 9 & 14 \end{bmatrix} \text{ ve } \Phi_{VT}T = \begin{bmatrix} 8 \\ 1 \\ 5 \end{bmatrix} \tag{3.31}$$

Veri-toplayıcı düğüm Φ_{VT}^{-1}'ı hesaplar ve aşağıdaki şekilde T matrisini açığa çıkartır:

$$T = \begin{bmatrix} 3 & 5 & 7 \\ 6 & 7 & 1 \\ 4 & 3 & 6 \end{bmatrix} \times \begin{bmatrix} 8 \\ 1 \\ 5 \end{bmatrix} = \begin{bmatrix} 6 \\ 7 \\ 8 \end{bmatrix} \tag{3.32}$$

T'nin hesaplanmasından sonra, veri-toplayıcı düğüm $\Delta_{VT}T^t$'yi hesaplar ve bunu $\Phi_{VT}S + \Delta_{VT}T^t$ matrisinden çıkartır:

$$\Phi_{VT}S = \begin{bmatrix} 15 & 10 & 6 \\ 10 & 11 & 13 \\ 0 & 9 & 14 \end{bmatrix} - \begin{bmatrix} 13 & 8 & 14 \\ 11 & 15 & 6 \\ 10 & 9 & 11 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 8 \\ 1 & 4 & 11 \\ 10 & 0 & 5 \end{bmatrix} \tag{3.33}$$

Son olarak veri-toplayıcı S matrisini $\Phi_{VT}^{-1}\Phi_{VT}S$ işlemi ile açığa çıkartır:

$$S = \begin{bmatrix} 3 & 5 & 7 \\ 6 & 7 & 1 \\ 4 & 3 & 6 \end{bmatrix} \times \begin{bmatrix} 2 & 2 & 8 \\ 1 & 4 & 11 \\ 10 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 3 & 4 \\ 2 & 4 & 5 \end{bmatrix} \tag{3.34}$$

Böylece mesaj matrisini oluşturan B tane orijinal veri sembolü yeniden elde edilmiş olur.

ASRC$_{\mathbf{DR}}$ Yöntemi Bu yöntemle, veri-toplayıcı düğüm ilk olarak T matrisini geri çatmaktadır. M_4 matrisini geri çatmak için, veri-toplayıcı düğüm $1,\ 2$ ve 4 numaralı düğümlere bağlanır ve $p_4(1) = 9,\ p_4(2) = 8$ ve $p_4(4) = 12$ değerlerini indirir. Ardından, veri-toplayıcı düğüm bağlandığı düğümlerin oluşturduğu Φ_d matrisini aşağıdaki şekilde oluşturur:

$$\Phi_4 = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 3 \\ 4 & 3 & 5 \end{bmatrix} \tag{3.35}$$

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 3 \\ 4 & 3 & 5 \end{bmatrix}}_{\Phi_4} \begin{bmatrix} M_{1..3,4} \end{bmatrix} = \begin{bmatrix} 9 \\ 8 \\ 12 \end{bmatrix} = \begin{bmatrix} p_4(1) \\ p_4(2) \\ p_4(4) \end{bmatrix} \tag{3.36}$$

Sonra, veri-toplayıcı düğüm Φ_4 matrisinin tersini alır ve önceden indirilmiş olan polinom sonuçları ile çarpar.

$$\underbrace{\begin{bmatrix} 2 & 6 & 7 \\ 6 & 1 & 1 \\ 5 & 7 & 6 \end{bmatrix}}_{\Phi_4^{-1}} \begin{bmatrix} 9 \\ 8 \\ 12 \end{bmatrix} = \begin{bmatrix} 6 \\ 7 \\ 8 \end{bmatrix} = M_{1..3,4} \tag{3.37}$$

M_4 sütunu geri çatıldıktan sonra veri-toplayıcı düğüm $M_{1..3,3}$ sütununu yani $p_3(x)$ polinomunu geri çatar. $M_{*,3}$ ile $M_{*,4}$ arasında ortak bir eleman vardır. Buna göre, veri-toplayıcı düğüm $p_3^{'}(x)$ polinomunu aşağıdaki şekilde oluşturur.

$$p_3^{'}(x) = p_3(x) - \sum_{i=3+1}^{d} x^{2^{i-1}} M_{i,3} \tag{3.38}$$

Bu örnekte, veri-toplayıcı düğüm yine $1,\ 2$ ve 4 numaralı düğümlere bağlanır ve $p_3(1) = 11, p_3(2) = 6$ ve $p_3(4) = 5$ değerlerini indirir. Ardından, veri-toplayıcı düğüm $p_3^{'}(1), p_3^{'}(2)$ ve $p_3^{'}(4)$ değerlerini hesaplar.

$$p_3^{'}(1) = p_3(1) - (8)(1) = 11 - 8 = 3 \tag{3.39}$$

$$p_3^{'}(2) = p_3(2) - (8)(5) = 6 - 14 = 8 \tag{3.40}$$

$$p_3^{'}(4) = p_3(4) - (8)(2) = 5 - 3 = 6 \tag{3.41}$$

Veri-toplayıcı düğüm Φ_3 matrisini kullanarak $M_{1..3,3}$ sütununu geri çatar.

$$\Phi_3 = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 3 \\ 4 & 3 & 5 \end{bmatrix} \tag{3.42}$$

Aşağıdaki şekilde, veri-toplayıcı düğüm $M_{*,3}$ matrisinin çözülmemiş elemanlarını ortaya çıkarmaktadır.

$$\underbrace{\begin{bmatrix} 2 & 6 & 7 \\ 6 & 1 & 1 \\ 5 & 7 & 6 \end{bmatrix}}_{\Phi_3^{'}} \begin{bmatrix} 3 \\ 8 \\ 6 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \\ 5 \end{bmatrix} = M_{1..3,3} \tag{3.43}$$

Sonra, veri-toplayıcı düğüm M_2'yi geri çatmaya başlar ve 1 ve 2 numaralı düğümlere bağlanır, ardından $p_2(1) = 1$ ve $p_2(2) = 10$ değerlerini indirir. Veri-toplayıcı düğüm indirdiği değerlerle $p_2^{'}(1)$ ve $p_2^{'}(2)$ değerlerini hesaplar.

$$p_2^{'}(1) = p_2(1) - (1)(4) - (1)(7) = 1 - 3 = 2 \tag{3.44}$$

$$p_2^{'}(2) = p_2(2) - (3)(4) - (5)(7) = 10 - 4 = 14 \tag{3.45}$$

Bu adımda, Φ_2 matrisi aşağıda gösterilmiştir:

$$\Phi_2 = \begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix} \tag{3.46}$$

Veri-toplayıcı düğüm Φ_2'yi kullanarak $M_{*,2}$'nin çözülmemiş elemanlarını hesaplar.

$$\underbrace{\begin{bmatrix} 15 & 7 \\ 14 & 7 \end{bmatrix}}_{\Phi_2^{-1}} \begin{bmatrix} 2 \\ 14 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} = M_2. \tag{3.47}$$

Son olarak, veri-toplayıcı düğüm M_1'i geri çatar. Bunun için veri-toplayıcı düğüm, 1 numaralı düğüme bağlanır ve $p_1(1) = 5$ değerini indirir. Ardından $p_1^{'}(1)$ değerini hesaplar:

$$p_1^{'}(1) = p_1(1) - (1)(1) - (1)(2) - (1)(6) = 5 - 5 = 0 \tag{3.48}$$

$$\Phi_1 = \begin{bmatrix} 1 \end{bmatrix} \tag{3.49}$$

Çözülmemiş son eleman da aşağıdaki gibi çözülür:

$$\underbrace{\begin{bmatrix} 1 \end{bmatrix}}_{\Phi_1^{-1}} \underbrace{\begin{bmatrix} 0 \end{bmatrix}}_{p_1^{'}(1)} = 0 = M_1. \tag{3.50}$$

Bu örnekte veri-toplayıcı düğümün orijinal veriyi toplamda 9 sembol yani orijinal verinin boyutu kadar sembol ($B = 9$) indirerek yapılandırdığı görülmektedir.

3.2.5.3 HMBR kodlarında tamir edilen verinin bütünlüğünün doğrulanması örneği

$$\sum_{i=1}^{\tau} C_{yrd_i} E_f = \sum_{i=1}^{\tau} E_{yrd_i}^t M E_f \tag{3.51}$$

$$= (\sum_{i=1}^{\tau} E_{yrd_i}^t) M E_f = E_f^t M E_f = E_f^t C_f^t. \tag{3.52}$$

Süregelen örneğimizde, arızalanan düğüm 4 numaralı düğüm idi. $w_{[4]} = w_{[1]} + w_{[5]}$ olduğu için $AMBR_{NR}$ yönteminde yeni-gelen düğüm, 1. ve 5. düğümden indirdiği sembolleri $\mathbb{F}_{2^4}$ sonlu cisminde toplayarak

$$\underbrace{4}_{C_1 E_4^t} + \underbrace{1}_{C_5 E_4^t} = \begin{bmatrix} 4 \\ 3 \\ 5 \\ 3 \end{bmatrix} \begin{bmatrix} 5 & 8 & 5 & 12 \end{bmatrix} \tag{3.53}$$

eşitliğini kontrol eder.

3.3 HMSR (Homomorfik Minimum Depolama Tamir Kodları) Kodları

Bu bölümde yeni bir melez kodlama şeması olan Homomorfik Minimum Depolama Tamir kodlama şeması anlatılmıştır. Bu yeni melez kodlama şemasında arızalanmış düğümler iki farklı seçenek ile tamir edilebilmektedir. Tamir için sunulan

ilk seçenek, diğer seçeneğe göre arızalanan düğümlerin, daha yüksek hesaplama karmaşıklığı ve daha az bant genişliği kullanımı ile tamir edilmesini, diğer seçenek ise öncekine göre daha az düğüme bağlanarak ve daha yüksek bant genişliği harcaması ve düşük hesaplama karmaşıklığı ile tamir edilmesini sağlamaktadır. Bu yeni kodlar minimum depolama yenileme kodlarındaki (Rashmi et al., 2011b) minimum depolama maliyeti noktasını sağlamaktadırlar.

Bu tez çalışması kapsamında önerdiğimiz yeni kodlama şeması, e-MSR (Rashmi et al., 2011b) ve HSRC (Oggier and Datta, 2011b) kodlarının düğüm tamirindeki avantajlarını birleştirmeyi amaçlamaktadır. $[n, k, d]$ e-MSR kodlarında (Rashmi et al., 2011b) yeni-gelen düğüm, arızalanan bir düğümün içeriğini tamir etmek için d tane düğüme bağlanır ve hepsinden birer tane sembol indirir, ardından çeşitli matris çarpımı işlemleri gerçekleştirir. İkinci bir kodlama şeması olan HSRC'de (Oggier and Datta, 2011b) ise arızalanan bir düğüm sadece iki düğümün tüm sembollerinin indirilmesinin ardından yalnız sonlu cisim üzerinden toplama işlemi yapılarak tamir edilebilmektedir. Bu kodlama şemalarının sırasıyla düğüm tamirinde düşük bant genişliği kullanımı, minimum depolama maliyeti ve düğüm tamirinde düşük hesaplama karmaşıklığı sağlamak gibi farklı noktalarda avantajları bulunmaktadır. Tasarlanan yeni melez kodlama şemasında, bu avantajlar tek bir kodlama şemasında toplanmıştır. Bu yeni kodlama şeması "Homomorfik Minimum Depolama Tamir Kodları (HMSR)" olarak adlandırılmıştır. Bir $[n, k, d]$ HMSR kodlama şeması $B = k\alpha = \alpha(\alpha + 1)$ tane sembolü kodlar ve her biri α adet sembol içeren n tane parça oluşturur. Bu n parçanın her biri farklı bir düğüme gönderilmektedir. Bu n parçadan doğrusal bağımsız olan d tanesi ya da özel bir koşulu sağlayan dört ya da daha fazlası arızalanmış bir düğümü tamir edebilmek için kullanılabilmektedir. Aşağıdaki bölümlerde bu yeni melez kodlama şemasının mimarisi, düğüm tamir etme işlemi ve verinin geri çatılması işlemleri gösterilecektir. Ayrıca düğüm tamiri ve verinin geri çatılması işlemlerindeki başarım olasılıkları bir sonraki bölümde detaylı bir şekilde anlatılmaktadır.

3.3.1 HMSR kodlama şemasının oluşturulması

HMSR kodlama şemasında, e-MSR kodlama şemasında olduğu gibi bir defada kodlanacak sembol miktarı $B = \alpha(\alpha + 1)$ olarak alınmıştır. Kodlama matrisi olan $(n \times d)$ boyutundaki E aşağıdaki şekilde oluşturulmaktadır:

$$E = \begin{bmatrix} \Phi & \Lambda\Phi \end{bmatrix} \tag{3.54}$$

Yukarıda görüldüğü gibi kodlama matrisinin genel yapısı e-MSR kodlarının kodlama matrisine benzemektedir. Ancak farklılık bu sefer $(n \times \alpha)$'lık Φ matrisinin ya-

pısından kaynaklanmaktadır. Φ matrisinin i'inci satırı e-MSR (Rashmi et al., 2011b) makalesindeki notasyon ile paralel olarak ϕ_i^t olarak adlandırılsın. Bu yeni melez kodlama şemasında ϕ_i^t vektörü aşağıdaki şekilde temsil edilmekte

$$\phi_i^t = \begin{bmatrix} x_i & {x_i}^2 & {x_i}^4 & \dots & {x_i}^{2^{\alpha-1}} \end{bmatrix}, \; i \in \{1, 2, \dots, n\}. \tag{3.55}$$

ve x_i sembolleri ikili gruplar halinde oluşturulmaktadır. Yani, 1. ve 2. düğümün ϕ_i^t vektörü aynı, 3. ve 4. düğümün ϕ_i^t vektörleri aynıdır ve bunun gibi diğer ikili grupların da vektörleri aynı ama her farklı grubun ϕ_i^t vektörleri farklıdır. Ayrıca her düğümün belirtecine denk gelen λ_i değeri farklıdır. Sonuç olarak kodlama matrisinin açık gösterimi aşağıdaki şekilde ifade edilebilir.

$$E = \begin{bmatrix} w_1 & w_1^2 & \dots & w_1^{2^{\alpha-1}} & \lambda_1 w_1 & \lambda_1 w_1^2 & \dots & \lambda_1 w_1^{2^{\alpha-1}} \\ w_1 & w_1^2 & \dots & w_1^{2^{\alpha-1}} & \lambda_2 w_1 & \lambda_2 w_1^2 & \dots & \lambda_2 w_1^{2^{\alpha-1}} \\ w_2 & w_2^2 & \dots & w_2^{2^{\alpha-1}} & \lambda_3 w_2 & \lambda_3 w_2^2 & \dots & \lambda_3 w_2^{2^{\alpha-1}} \\ w_2 & w_2^2 & \dots & w_2^{2^{\alpha-1}} & \lambda_4 w_2 & \lambda_4 w_2^2 & \dots & \lambda_4 w_2^{2^{\alpha-1}} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{n/2} & w_{n/2}^2 & \dots & w_{n/2}^{2^{\alpha-1}} & \lambda_{n-1} w_{n/2} & \lambda_{n-1} w_{n/2}^2 & \dots & \lambda_{n-1} w_{n/2}^{2^{\alpha-1}} \\ w_{n/2} & w_{n/2}^2 & \dots & w_{n/2}^{2^{\alpha-1}} & \lambda_n w_{n/2} & \lambda_n w_{n/2}^2 & \dots & \lambda_n w_{n/2}^{2^{\alpha-1}} \end{bmatrix} \tag{3.56}$$

Mesaj matrisi M burada da kodlama matrisi ile çarpılan orijinal veri sembollerinden oluşmaktadır. M matrisinin yapısı e-MSR (Rashmi et al., 2011b) kodlama şemasındaki mesaj matrisi ile aynıdır. M matrisi aşağıda verilmiştir.

$$M = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} \tag{3.57}$$

B tane sembolün $\binom{\alpha+1}{2}$'lik kısmı S_1 matrisini simetrik yapacak şekilde mesaj matrisine doldurulur. Geri kalan $\binom{\alpha+1}{2}$'lik kısmı ise yine e-MSR'da olduğu gibi S_2 simetrik olacak şekilde doldurulur. Kodlama işlemi aşağıdaki gibi gerçekleştirilmektedir:

$$C = E \times M \tag{3.58}$$

Sonuçta oluşan $(n \times \alpha)$'lık C matrisinin her bir satırı başka bir düğüme gönderilmektedir. Dolayısıyla, sistemde toplamda n tane düğüm bulunmaktadır.

3.3.2 HMSR kodları ile arızalanan bir düğümün tamiri işlemi

HMSR kodlama şemasında iki farklı düğüm tamiri seçeneği bulunmaktadır. Bunlardan ilki e-MSR'daki tamir etme yönteminin bir adaptasyonudur, $AMSR_{NR}$ olarak adlandırılmıştır; diğeri ise HSRC kodlama şemasındaki tamir etme işleminin bir varyasyonudur ve $ASRC_{NR}$ olarak adlandırılmıştır. Bu tamir etme seçenekleri aşağıda sırasıyla anlatılmıştır.

3.3.2.1 AMSR$_{NR}$ yöntemi ile düğüm tamiri

Arızalanmış düğüm f ile ifade edilmiş olsun. Bu düğümün yerine gelen düğüm d tane düğüme bağlanır, öyle ki; bu düğümlerin sorumlu oldukları kodlama matrisi satırları olan e_i^t'lerin doğrusal bağımsız olmaları gerekmektedir. Bu yöntemin e-MSR kodları ile düğüm tamirinden farkı seçilen yardımcı düğümlerin e_i^t'lerinin doğrusal bağımsız olması gerekliliğidir. Yeni-gelen düğüm bu koşulu sağlayan d tane düğüm bulduğunda ($\{t_1, t_1, \ldots, t_d\}$) bu yardımcı düğümlere bağlanır. Her yardımcı düğüm depolamakta olduğu kod satırı ($e_i^t M$) ile arızalanmış düğümün Φ'deki kodlama satırının devriğini (ϕ_f) çarparak, $e_i^t M \phi_f$, sonucu yeni-gelen düğüme gönderir. Tüm yardımcı düğümlerden sonuçları alan yeni-gelen düğüm aşağıdaki matrisi elde etmiş olur:

$$\left[E_{yrd} M \phi_f \right] \tag{3.59}$$

Eşitlik (3.59)'te bulunan E_{yrd}'in içeriği aşağıda gösterilmiştir:

$$E_{yrd} = \begin{bmatrix} e_{t_1}^t \\ e_{t_2}^t \\ \vdots \\ e_{t_d}^t \end{bmatrix} \tag{3.60}$$

Yani E_{yrd}, her yardımcı düğümün ilgili kodlama matrisi satırlarından oluşmaktadır. Başlangıçta bu düğümler kodlama matrisi satırları doğrusal bağımsız olacak şekilde seçildikleri için E_{yrd} tersinir bir matristir. Dolayısıyla, yeni-gelen düğüm E_{yrd}'in tersini hesaplayarak $M\phi_f$'i elde edebilir. $M\phi_f$'nin içeriği aşağıdaki gibidir:

$$M\phi_f = \begin{bmatrix} S1 \\ S2 \end{bmatrix} \phi_f = \begin{bmatrix} S_1\phi_f \\ S_2\phi_f \end{bmatrix} \tag{3.61}$$

$S_1\phi_f$ ve $S_2\phi_f$ matrislerinin devriği, S_1 ve S_2 simetrik olduğu için e-MSR

kodlarında olduğu gibi, $\phi_f^t S_1$ ve $\phi_f^t S_2$ olarak elde edilir. Arızalamış düğümün kayıp verisi $\phi_f^t S_1 + \lambda_f \phi_f^t S_2$ idi. λ_f her düğüm tarafından bilindiği için Eşitlik (3.61) kullanılarak kayıp veri tamir edilir.

3.3.2.2 ASRC$_{NR}$ yöntemi ile düğüm tamiri

Bu yöntem ile tamirde, yeni-gelen düğüm toplamda en az dört tane düğüme bağlanmalıdır. Bu yöntemin HSRC kodları ile düğüm tamirinden farkı bağlanılması gereken düğüm sayısı ve yeni-gelen düğümün gerçekleştirdiği işlemlerdir. Yine arızalanmış olan düğüm f ifade edilsin. w_f ise f düğümüne karşılık gelen kodlama matrisinin ilk sembolü olsun.

$$w_f = w_{a_1} + w_{b_1} = w_{a_2} + w_{b_2} \tag{3.62}$$

Eğer Eşitlik (3.62)'yi sağlayabilen $a_1,\ b_1$ düğümleri ve bunların eşleniği iki düğüm (yani aynı ϕ_i^t vektörüne sahip iki düğüm), $a_2,\ b_2$, bulunabilirse, sadece bu dört düğüme bağlanılarak arızalanmış düğüm tamir edilebilir. a_1 düğümünün içerisinde $C_{a_1} = \phi_a^t S_1 + \lambda_{a_1} \phi_a^t S_2$, a_2 düğümünün içerisinde $C_{a_2} = \phi_a^t S_1 + \lambda_{a_2} \phi_a^t S_2$ bulunmaktadır. $\phi_a^t S_1$'e x ve $\phi_a^t S_2$'ye y denilirse aşağıdaki iki fonksiyon değerine ulaşılır.

$$C_{a_1} = x + \lambda_{a_1} y \tag{3.63}$$

$$C_{a_2} = x + \lambda_{a_2} y \tag{3.64}$$

Yeni-gelen düğüm Eşitlik (3.63) ve (3.64)'ü kullanarak, yani iki bilinmeyenli iki tane denklemi kullanarak x ve y'yi yani $\phi_a^t S_1$ ve $\phi_a^t S_2$ değerlerini elde edebilir. Benzer şekilde, b_1 ve b_2 düğümlerinde, sırasıyla $\phi_b^t S_1 + \lambda_{b_1} \phi_b^t S_2$ ve $\phi_b^t S_1 + \lambda_{b_2} \phi_b^t S_2$ değerleri bulunmaktadır. Burada $\phi_b^t S_1$'ye x' ve $\phi_b^t S_2$'ye y' denilirse bu düğümlerden (b_1 ve b_2) aşağıdaki değerler elde edilebilir.

$$C_{b_1} = x' + \lambda_{b_1} y' \tag{3.65}$$

$$C_{b_2} = x' + \lambda_{b_2} y' \tag{3.66}$$

Burada (3.65) ve (3.66) eşitlikleri kullanılarak x', y' çözülüp $\phi_b^t S_1$ ve $\phi_b^t S_2$ değerlerine erişilebilir. Bu elde edilen dört farklı değer ile arızalı düğümde kaybolan veri yeniden elde edilebilir.

$$\phi_a^t S_1 + \phi_b^t S_1 = \phi_f^t S_1 \tag{3.67}$$

$w_f = w_{a_1} + w_{b_1}$ olduğu için Eşitlik (3.67), homomorfizm özelliği ile sağlanmaktadır. Yine benzer şekilde aşağıdaki eşitlik de sağlanmaktadır.

$$\phi_a^t S_2 + \phi_b^t S_2 = \phi_f^t S_2 \tag{3.68}$$

Eşitlik (3.68) λ_f ile çarpılıp Eşitlik (3.67) ile sonlu cisim üzerinden toplanırsa kayıp veri tamir edilmiş olur.

$$\phi_f^t S_1 + \lambda_f \phi_f^t S_2 \tag{3.69}$$

3.3.3 HMSR kodları ile kodlanmış verinin geri çatılması

Verinin geri çatılması işlemi, kodlanan orijinal verinin, kodlanmış veriyi kullanarak yeniden elde edilmesidir. Buradaki yöntem e-MSR'daki verinin geri çatılması yöntemine benzemektedir. Burada ϕ_i^t değerleri doğrusal bağımsız k tane düğüme bağlanılmalıdır. Verinin geri çatılması işlemini gerçekleştiren düğüme veri-toplayıcı düğüm denilmektedir. Veri-toplayıcı düğüm bu şekilde ϕ_i^t değerleri doğrusal bağımsız k tane düğüme bağlanır ve o düğümlerde saklanan ilgili tüm verileri indirir. Bu verileri aldıktan sonra veri-toplayıcı düğüm, $\begin{bmatrix}\Phi_{VT}S_1 + \Lambda_{VT}\Phi_{VT}S_2\end{bmatrix}$ matrisini elde etmiş olur. Bu matris ise aşağıdaki şekilde ifade edilebilir:

$$\underbrace{\begin{bmatrix}\Phi_{VT} & \Lambda_{VT}\Phi_{VT}\end{bmatrix}}_{E_{VT}} \begin{bmatrix}S_1 \\ S_2\end{bmatrix} = \begin{bmatrix}\Phi_{VT}S_1 + \Lambda_{VT}\Phi_{VT}S_2\end{bmatrix} \tag{3.70}$$

Yukarıda görülen E_{VT}, bağlanılan k düğüme karşılık gelen kodlama matrisi satırlarından oluşmaktadır. Eşitlik (3.70), sağ taraftan Φ_{VT}^T ile çarpılırsa aşağıdaki matris elde edilir.

$$\Phi_{VT}S_1\Phi_{VT}^T + \Lambda_{VT}\Phi_{VT}S_2\Phi_{VT}^T \tag{3.71}$$

e-MSR'da notasyonuna paralel olarak $\Phi_{VT}S_1\Phi_{VT}^T$, P; $\Phi_{VT}S_2\Phi_{VT}^T$ ise Q olarak adlandırılmıştır. S_1 ve S_2 simetrik olduğu için P ve Q de simetrik matrislerdir. Eşitlik (3.71)'deki matris $P + \Lambda_{VT}Q$ ile gösterilebilir. Bu matrisin (i, j)'inci elemanı $P_{ij} + \lambda_i Q_{ij}$ ile ifade edilmektedir. Aynı şekilde (j, i)'inci elemanı ise $P_{ji} + \lambda_j Q_{ji}$ ile gösterilmektedir. P_{ij}, P_{ji}'ye; Q_{ij}, Q_{ji}'ye eşit olduğu için P ve Q matrislerinin köşegen olmayan elemanlarının hepsi çözülebilir. P'nin i'inci satırı aşağıda verilmiştir.

$$\phi_i^t S_1 \begin{bmatrix}\phi_1 & \dots & \phi_{i-1} & \phi_{i+1} & \dots & \phi_{\alpha+1}\end{bmatrix} \tag{3.72}$$

Eşitlik (3.72)'deki matrisin sağ tarafı tersinirdir (ϕ_i^t'ler doğrusal bağımsız olarak seçildiği için). Veri-toplayıcı düğüm, Eşitlik (3.72)'nin sağ tarafındaki matrisi tersi ile çarpıp aşağıdaki değerleri elde elde edebilir:

$$\{\phi_i^t S_1 | 1 \leqslant i \leqslant \alpha + 1\} \tag{3.73}$$

Yukarıdaki değerlerin α tanesi kulanılarak veri-toplayıcı düğüm aşağıdaki matrisi oluşturur.

$$\begin{bmatrix} \phi_1^t \\ \vdots \\ \phi_\alpha^t \end{bmatrix} S_1 \tag{3.74}$$

Yukarıdaki ifadenin sol tarafındaki matris tersinir olduğu için S_1 yeniden elde edilmiş olur. Bu işlemler aynı şekilde Q için de tekrarlanarak S_2 matrisi de elde edilerek, verinin geri çatılması işlemi tamamlanmış olur.

3.3.4 HMSR kodu örneği

Bu bölümde önerilen HMSR kodlarının bir örneği gösterilmektedir. Bunun için $n = 10,\ d = 6,\ k = 4,\ \alpha = 3$ ve sonlu cisim boyutu $q = 64$ alınmıştır. Alt bölüm 3.3.1'de anlatıldığı gibi oluşturulan kodlama matrisi ve Λ aşağıda gösterilmiştir:

$$E = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 12 & 12 & 12 \\ 2 & 4 & 16 & 16 & 32 & 6 \\ 2 & 4 & 16 & 6 & 12 & 48 \\ 3 & 5 & 17 & 35 & 38 & 56 \\ 3 & 5 & 17 & 48 & 19 & 28 \\ 4 & 16 & 12 & 24 & 35 & 40 \\ 4 & 16 & 12 & 16 & 3 & 48 \\ 5 & 17 & 13 & 59 & 18 & 62 \\ 5 & 17 & 13 & 5 & 17 & 13 \end{bmatrix} \tag{3.75}$$

$$\Lambda = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.76}$$

Kodlanması gereken orijinal $B = 12$ sembol ise M matrisi içerisinde aşağıdaki gibi verilmiştir:

$$M = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 3 & 4 \\ 2 & 4 & 5 \\ 6 & 7 & 8 \\ 7 & 9 & 10 \\ 8 & 10 & 11 \end{bmatrix} \tag{3.77}$$

Kodlama sonucu oluşan matris C ise aşağıdaki şekildedir:

$$E \times M = C = \begin{bmatrix} 17 & 14 & 17 \\ 44 & 54 & 44 \\ 18 & 46 & 52 \\ 30 & 25 & 21 \\ 22 & 7 & 33 \\ 30 & 13 & 55 \\ 56 & 36 & 23 \\ 40 & 5 & 54 \\ 19 & 39 & 47 \\ 10 & 55 & 12 \end{bmatrix} \tag{3.78}$$

Yukarıdaki C matrisinin her bir satırı başka bir düğüme gönderilmektedir. Yani her düğüm $\alpha = 3$ sembol depolamaktadır.

Aşağıda arızalanan bir düğümün sırasıyla $AMSR_{NR}$ ve $ASRC_{NR}$ yöntemleri ile tamir edilmesi ilerleyen iki alt bölümde gösterilmiştir.

3.3.4.1 AMSR$_{NR}$ yöntemi ile düğüm tamiri

Birinci düğümün arızalandığını varsayalım; bu durumda kodlama matrisinin $\{17, 14, 17\}$ verisi kayıptır. Yeni-gelen düğüm, düğüm tamiri için bağlanması gereken doğrusal bağımsız veri içeren $d = 6$ adet düğüm olarak $5,\ 6,\ 7,\ 8,\ 9,\ 10$ numaralı düğümlere bağlanır ve bu yardımcı düğümlerin her biri kendi kod satırı $e_{t_i}^t M$ ile ϕ_1 vektörünü $\mathbb{F}_{64}$ sonlu cismi üzerinde çarpıp yeni-gelen düğüme gönderir.

$$\begin{bmatrix} 22 & 7 & 33 \\ 30 & 13 & 55 \\ 56 & 36 & 23 \\ 40 & 5 & 54 \\ 19 & 39 & 47 \\ 10 & 55 & 12 \end{bmatrix} \times \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 48 \\ 36 \\ 11 \\ 27 \\ 27 \\ 49 \end{bmatrix} \tag{3.79}$$

E_{yrd} matrisi ise Eşitlik (3.60) kullanılarak aşağıdaki şekilde elde edilir:

$$\begin{bmatrix} 3 & 5 & 17 & 35 & 38 & 56 \\ 3 & 5 & 17 & 48 & 19 & 28 \\ 4 & 16 & 12 & 24 & 35 & 40 \\ 4 & 16 & 12 & 16 & 3 & 48 \\ 5 & 17 & 13 & 59 & 18 & 62 \\ 5 & 17 & 13 & 5 & 17 & 13 \end{bmatrix} \tag{3.80}$$

Bu matrisin tersi şu şekildedir:

$${E_{yrd}}^{-1} = \begin{bmatrix} 37 & 9 & 57 & 4 & 9 & 29 \\ 3 & 6 & 48 & 40 & 55 & 21 \\ 38 & 15 & 11 & 47 & 49 & 6 \\ 14 & 14 & 63 & 63 & 9 & 9 \\ 4 & 4 & 12 & 12 & 55 & 55 \\ 10 & 10 & 18 & 18 & 49 & 49 \end{bmatrix} \tag{3.81}$$

E_{yrd} matrisini (3.79)'uncu eşitlikteki matristen aşağıdaki yöntemle elersek $M\phi_1$ matrisini elde etmiş oluruz:

$$M\phi_1 = {E_{yrd}}^{-1} E_{yrd} M\phi_1 = \begin{bmatrix} 3 \\ 6 \\ 3 \\ 9 \\ 4 \\ 9 \end{bmatrix} \tag{3.82}$$

Bu adımdan sonra $\phi_1^t S_1 + \lambda_1 \phi_f^t S_2$ hesaplanır, yani:

$$\begin{bmatrix} 3 & 6 & 3 \end{bmatrix} + 2 \begin{bmatrix} 9 & 4 & 9 \end{bmatrix} = \begin{bmatrix} 17 & 14 & 17 \end{bmatrix} = C_1 \tag{3.83}$$

3.3.4.2 ASRC_{NR} yöntemi ile düğüm tamiri

Yine birinci düğümün arızalandığını yani C_1 verisinin kaybolduğunu farzedelim. $\mathbb{F}_{64}$'e göre $1 = 2+3$ olduğu için çalışır durumdaki düğümler arasından $3,\ 4,\ 5$ ve 6. düğümlere bağlanılarak aşağıdaki veriler indirilir:

$$\begin{bmatrix} 18 & 46 & 52 \\ 30 & 25 & 21 \\ 22 & 7 & 33 \\ 30 & 13 & 55 \end{bmatrix} \tag{3.84}$$

3 ve 4. düğüme karşılık gelen λ_i değerleri sırasıyla: 8 ve 3'tür. Bu iki düğümde sırasıyla aşağıdaki değerler bulunmaktadır:

$$\phi_3^t S_1 + 8\phi_3^t S_2 = \begin{bmatrix} 18 & 46 & 52 \end{bmatrix} \tag{3.85}$$

$$\phi_3^t S_1 + 3\phi_3^t S_2 = \begin{bmatrix} 30 & 25 & 21 \end{bmatrix} \tag{3.86}$$

Yukarıdaki iki denklemden $\phi_3^t S_1$ ve $\phi_3^t S_2$'yi çözersek, $\phi_3^t S_1 = \begin{bmatrix} 36 & 13 & 7 \end{bmatrix}$ olarak ve $\phi_3^t S_2 = \begin{bmatrix} 22 & 12 & 14 \end{bmatrix}$ olarak elde edilmektedir. Benzer şekilde 5 ve 6. düğümlere karşılık gelen λ_i değerleri sırasıyla 32 ve 16'dir ve bu düğümler aşağıdaki değerlere sahiptir:

$$\phi_5^t S_1 + 32\phi_5^t S_2 = \begin{bmatrix} 22 & 7 & 33 \end{bmatrix} \tag{3.87}$$

$$\phi_5^t S_1 + 16\phi_5^t S_2 = \begin{bmatrix} 30 & 13 & 55 \end{bmatrix} \tag{3.88}$$

Yukarıdaki işlemlere benzer şekilde $\phi_5^t S_2 = \begin{bmatrix} 31 & 8 & 7 \end{bmatrix}$; $\phi_5^t S_1 = \begin{bmatrix} 39 & 11 & 4 \end{bmatrix}$ olarak bulunmaktadır. Kayıp düğümün içerisinde $\phi_1^t S_1 + 2\phi_1^t S_2$ değerleri bulunmaktaydı. Yani $\phi_1^t S_1$ ve $\phi_1^t S_2$ değerleri çözüldüğünde kayıp veri yeniden hesaplanabilir. Homomorfizm özelliğinden aşağıdaki hesaplamalar ile $\phi_1^t S_1$ ve $\phi_1^t S_2$ değerleri şu şekilde bulunabilir:

$$\phi_1^t S_1 = \phi_3^t S_1 + \phi_5^t S_1 = \begin{bmatrix} 36 & 13 & 7 \end{bmatrix} + \begin{bmatrix} 39 & 11 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 6 & 3 \end{bmatrix} \tag{3.89}$$

$$\phi_1^t S_2 = \phi_3^t S_2 + \phi_5^t S_2 = \begin{bmatrix} 22 & 12 & 14 \end{bmatrix} + \begin{bmatrix} 31 & 8 & 7 \end{bmatrix} = \begin{bmatrix} 9 & 4 & 9 \end{bmatrix} \tag{3.90}$$

Son olarak kayıp veri aşağıdaki şekilde bulunur:

$$\phi_1^t S_1 + 2\phi_1^t S_2 = \begin{bmatrix} 3 & 6 & 3 \end{bmatrix} + 2 \begin{bmatrix} 9 & 4 & 9 \end{bmatrix} = \begin{bmatrix} 17 & 14 & 17 \end{bmatrix} \tag{3.91}$$

3.3.4.3 Verinin geri çatılması örneği

Burada orijinal verinin geri çatılması için $k = 4$ tane düğüme bağlanılması gerekmektedir. Öyle ki bağlanılan düğümlerin herhangi α tanesinin ϕ^t vektörleri doğrusal bağımsız olmalıdır. Bu örnekte veri-toplayıcı düğüm $3,\ 5,\ 7$ ve 9. düğümlere bağlanır ve onlardan tüm veri sembollerini (ilgili C_i satırlarını) indirir:

$$E_{VT}M = \begin{bmatrix} 18 & 46 & 52 \\ 22 & 7 & 33 \\ 56 & 36 & 23 \\ 19 & 39 & 47 \end{bmatrix} \tag{3.92}$$

Burada, E matrisi içerisinden elde edilen Φ_{VT} aşağıda verilmiştir:

$$\Phi_{VT} = \begin{bmatrix} 2 & 4 & 16 \\ 3 & 5 & 17 \\ 4 & 16 & 12 \\ 5 & 17 & 13 \end{bmatrix} \tag{3.93}$$

Alınan veriler Φ_{VT} matrisinin devriği ile çarpılır.

$$E_{VT}M{\Phi_{VT}}^T = \begin{bmatrix} 13 & 5 & 9 & 1 \\ 56 & 8 & 46 & 30 \\ 26 & 17 & 31 & 20 \\ 17 & 10 & 41 & 50 \end{bmatrix} = P + \Lambda_{VT}Q \tag{3.94}$$

$P_{ij} + \lambda_i Q_{ij}$ ve $P_{ji} + \lambda_j Q_{ji}$ değerleri kullanılarak P ve Q çözülür.

$$P = \begin{bmatrix} * & 34 & 39 & 9 \\ 34 & * & 28 & 52 \\ 39 & 28 & * & 51 \\ 9 & 52 & 51 & * \end{bmatrix} \tag{3.95}$$

$$\phi_3^t S_1 \begin{bmatrix} \phi_5 & \phi_7 & \phi_9 \end{bmatrix} = \begin{bmatrix} 34 & 39 & 9 \end{bmatrix} \tag{3.96}$$

$$\phi_3^t S_1 = \begin{bmatrix} 36 & 13 & 7 \end{bmatrix} \tag{3.97}$$

$$\phi_5^t S_1 = \begin{bmatrix} 39 & 11 & 4 \end{bmatrix} \tag{3.98}$$

$$\phi_7^t S_1 = \begin{bmatrix} 8 & 4 & 55 \end{bmatrix} \tag{3.99}$$

$$\Phi_{VT_\alpha}^{-1} \begin{bmatrix} 36 & 13 & 7 \\ 39 & 11 & 4 \\ 8 & 4 & 55 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 2 \\ 1 & 3 & 4 \\ 2 & 4 & 5 \end{bmatrix} = S_1 \tag{3.100}$$

$$Q = \begin{bmatrix} * & 45 & 21 & 1 \\ 45 & * & 29 & 13 \\ 21 & 29 & * & 47 \\ 1 & 13 & 47 & * \end{bmatrix} \tag{3.101}$$

$$\phi_3^t S_2 = \begin{bmatrix} 22 & 12 & 14 \end{bmatrix} \tag{3.102}$$

$$\phi_5^t S_2 = \begin{bmatrix} 31 & 8 & 7 \end{bmatrix} \tag{3.103}$$

$$\phi_7^t S_2 = \begin{bmatrix} 8 & 49 & 49 \end{bmatrix} \tag{3.104}$$

$$\Phi_{VT\alpha}{}^{-1} \begin{bmatrix} 22 & 12 & 14 \\ 31 & 8 & 7 \\ 8 & 49 & 49 \end{bmatrix} = \begin{bmatrix} 6 & 7 & 8 \\ 7 & 9 & 10 \\ 8 & 10 & 11 \end{bmatrix} = S_2 \tag{3.105}$$

3.4 Kümeleme Tabanlı Dağıtık Depolama Sistemi

Kümeleme tabanlı kodlama şeması çalışmasında, farklı kodlama şemalarının aynı çatı altında toplanarak farklı ihtiyaçlara sahip düğümlerin kendisine uygun olan kodlama şemasını kullanabilmesi hedeflenmiştir. Bu sayede, benzer ihtiyaçlara sahip düğümlerin ihtiyaçlarına yönelik kodlama şemalarının kullanılması sağlanmış olacaktır. Örneğin, bazı düğümler için depolama alanı kısıtlı olabilir, bant genişliği ya da CPU kullanımının kısıtlı olduğu durumlar bulunabilir. Böyle durumlarda tüm düğümlerin aynı kodlama şemasını kullanması kaynakların kullanımı açısından çok verimli olmayacaktır.

Önerilen kümeleme tabanlı dağıtık depolama sisteminde kodlama şemaları olarak e-MBR (Rashmi et al., 2011b), HSRC (Oggier and Datta, 2011b) ve HMBR (Haytaoglu and Dalkilic, 2013a) kullanılmıştır. e-MBR kodlama şeması düğüm tamiri işleminde HSRC kodlama şemasına göre daha yüksek işlem karmaşıklığına sahiptir, ancak düğüm tamiri başarım olasılığı daha yüksektir. HMBR kodlama şeması ise iki farklı düğüm tamiri yöntemi sunmaktadır. Bu yöntemlerden biri az düğüme bağlanarak düşük işlem karmaşıklığı ile bir düğümü tamir edebilirken diğeri ilkine göre daha fazla düğüme bağlanarak ve daha az bant genişliği kullanılarak tamir işlemini gerçekleştirebilmektedir. Buna göre, daha düşük CPU kaynağına sahip olan düğümler HSRC kodlama şemasını, ağ trafiği kısıtlı olmayan düğümler HMBR kodlama şemasını, düğüm tamirinde süre kısıtı olan düğümler ise e-MBR kodlama şemasını kullanmak üzere kümelendirilmiştir.

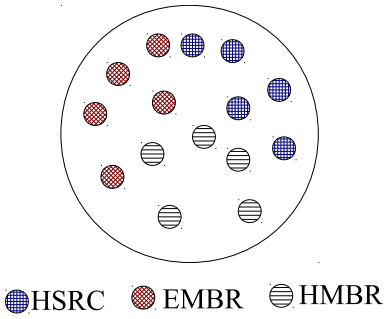
Bu çalışmanın getirilerinin ölçülmesi için önerilen kümeleme tabanlı depolama sistemi ile (Şekil 3.5) böyle bir kümeleme çatısı kullanmayan depolama sistemlerinin (Şekil 3.6) ns-3 (ns-3, 2015) benzetim ortamında uygulaması gerçekleştirilmiştir. Şekil 3.5'de verilen kümeleme tabanlı dağıtık depolama sistemi uygulaması için yukarıda bahsedilen e-MBR (Rashmi et al., 2011b) kodları, HSRC (Oggier and Datta, 2011b) ve HMBR (Haytaoglu and Dalkilic, 2013a) kodlarının kodlama, düğüm tamiri ve veriyi geri çatma işlevsellikleri ayrı ayrı kodlanmıştır. Kümeleme tabanlı olmayan depolama sistemlerinde düğümler rastgele eşit büyüklükte üç farklı

gruba ayrılmıştır. Her bir grup aynı kodlama şemasını çalıştırmaktadır. Örneğin Şekil 3.6'da HSRC sisteminde her bir grup 5 düğümden oluşmakta ve her grup HSRC kodlama şemasını kullanmaktadır. Her bir kodlama şeması için, Şekil 3.6'da görüldüğü gibi, toplamda üç farklı kümeleme tabanlı olmayan depolama sistemi gerçekleştirilmiştir.

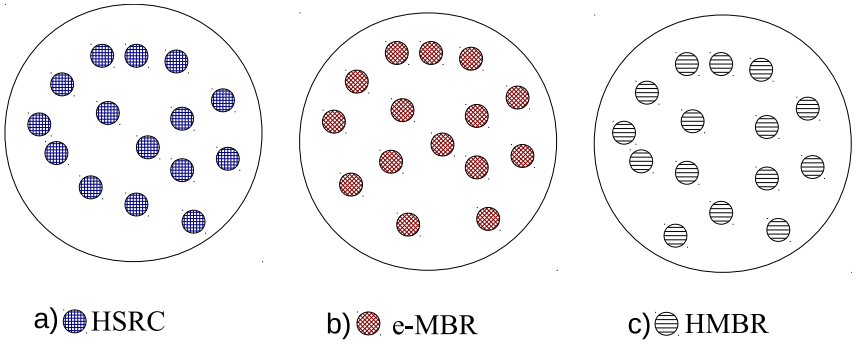
Şekil 3.5'de görülen kümeleme tabanlı depolama sisteminde düğümler ağdaki fiziksel yakınlıklarına göre üç eşit kümeye ayrılmaktadır. Farklı renk ve desenler, bir düğümün kullandığı kodlama şemasını dolayısıyla ait olduğu kümeyi belirtmektedir. Kümelere sırasıyla e-MBR kodları, HSRC ve HMBR kodlama şemalarından biri atanmaktadır. Kümedeki her düğüme ağın genelindeki düğümlerin hangi küme kapsamında çalıştığı bilgisi gönderilmektedir. Bu şekilde başka kümeye ait mesajların geçişi tespit edilebilmektedir. Her düğüm kendi kümesi ile ilgili iş parçacıklarını yerine getirirken kendi gönderdiği ya da ilettiği mesajların sayısını tutmakta olup bu iş parçacıkları ile ilgili geçen süreleri kaydetmektedir. Bir küme içerisindeki her düğüme kodlama şemasında kullanılmak üzere bir mantıksal belirteç (ilave kimlik) atanmaktadır.

Her düğüm kendi grubu ya da kümesindeki düğümlerin IP adreslerini bilmekte olup aynı kümedeki (Şekil 3.5) ya da gruptaki (Şekil 3.6) düğümlere karşılık gelen mantıksal belirteçlerden de haberdardır. Bu mantıksal belirteçler düğüm tamiri ve verinin geri çatılması işlemlerinde görevlendirilen düğümün hangi düğümlere bağlanacağını seçmesi için kullanılmaktadır. Şekil 3.5'de kümeleme tabanlı depolama sistemin taslağı bulunmaktadır.

İzleyen alt bölümde her küme ya da grupta çalıştırılması planlanan e-MBR kodları, HSRC ve HMBR kodlama şemalarının sonlu durum makinaları incelenmiştir.



Şekil 3.5. Kümeleme tabanlı depolama sistemi.



Şekil 3.6. Kümeleme tabanlı olmayan depolama sistemleri a) Sadece HSRC kodlama şemasının kullanıldığı sistem, b) Sadece e-MBR Kodlama şemasının kullanıldığı sistem, c) Sadece HMBR kodlama şemasının kullanıldığı sistem.

3.4.1 Kodlama şemalarının işleyişi

Bölüm 2'de, e-MBR (Rashmi et al., 2011b) ve HSRC (Oggier and Datta, 2011b), Bölüm 3'te ise HMBR kodlama şeması (Haytaoglu and Dalkilic, 2013a) ayrıntılı bir şekilde anlatılmıştı. Bu yüzden bu bölümde kodlama şemalarının ayrıntılarından ziyade bu kodların kullanıldığı bir depolama sisteminde bulunan düğümlerin genel olarak çalışma prensibinden bahsedilmiştir.

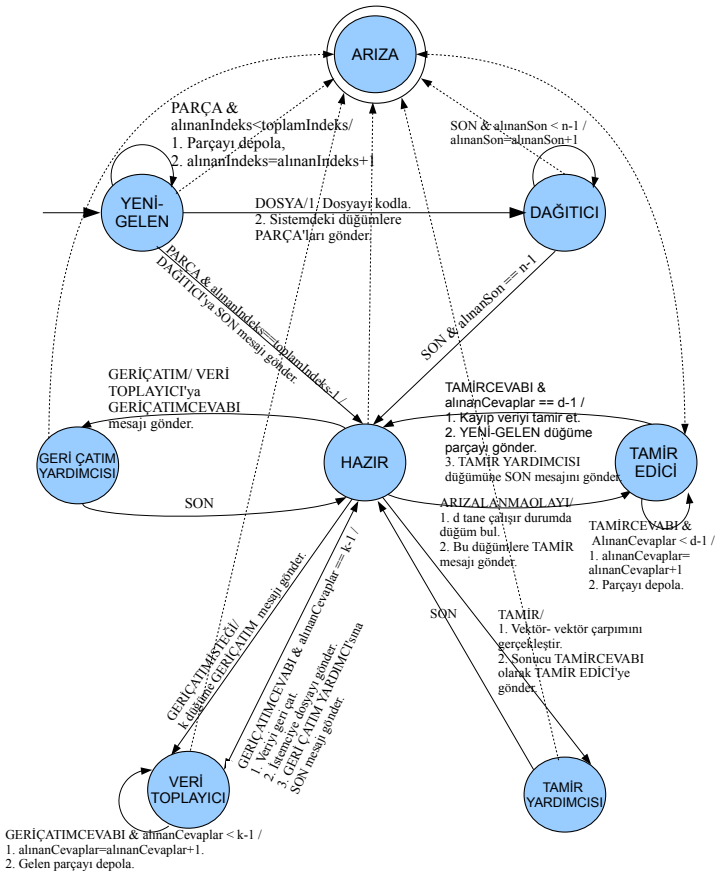
e-MBR kodlama şeması kullanılan bir depolama sisteminde bulunan bir düğümün sonlu durum makinası Şekil 3.7'de verilmiştir[1]. Sadece e-MBR kodlama şemasının kullanıldığı bir sistemde bulunan bir düğümün durumları YENİ-GELEN, DAĞITICI, HAZIR, TAMİREDİCİ, TAMİRYARDIMCISI, VERİ TOPLAYICI, GERİ ÇATIM YARDIMCISI ve ARIZA'dır.

YENİ-GELEN durumundaki bir düğüm bozulmuş bir düğüme ait parçayı depolamak için *PARÇA* mesajı alırsa bu parçanın tamamını alana kadar bekler ardından HAZIR durumuna geçer. **YENİ-GELEN** durumundaki bir düğüm eğer *DOSYA* mesajı alırsa sistemde depolanması istenen orijinal dosyayı e-MBR kodlama sistemine göre kodlayıp n tane *PARÇA* oluşturur ve sistemdeki n düğüme farklı bir *PARÇA* göndererek DAĞITICI durumuna geçer.

DAĞITICI durumundaki bir düğüm *PARÇA* gönderdiği her düğümden *SON* mesajını aldığında artık işlemi tamamlanmıştır ve *HAZIR* durumuna geçer.

---

[1] Şekil 3.7'de ARIZA durumuna gelen geçişler diğer oklarla karışmaması için kesiklidir.



Şekil 3.7. e-MBR kodlama şemasını kullanan bir dağıtık depolama sisteminde çalışan bir düğümün sonlu durum makinası.

HAZIR durumundaki düğüm, bir düğümün arızalandığını farkettiğinde d farklı düğüme *TAMİR* mesajı göndererek TAMİREDİCİ durumuna geçer. Eğer verinin geri çatılmasına ilişkin bir istek alırsa HAZIR durumundaki düğüm VERİ TOPLAYICI durumuna geçer. HAZIR durumundaki bir düğüme VERİ TOPLAYICI düğümden bir *GERİÇATIM* mesajı gelirse GERİ ÇATIM YARDIMCISI durumuna geçer. Benzer şekilde, HAZIR durumundaki bir düğüm TAMİR EDİCİ düğümden *TAMİR* mesajı alırsa TAMİRYARDIMCISI konumuna geçer.

TAMİR EDİCİ durumundaki bir düğüm *TAMİR* mesajı gönderdiği düğümlerin hepsinden *TAMİRCEVABI* alana kadar bekler, tüm cevapları aldığında tamir işlemini gerçekleştirir, tamir edilen *PARÇA*'yı *YENİ-GELEN* durumunda olan düğüme gönderir. TAMİR YARDIMCISI durumundaki düğümlere ise *SON* mesajı göndermektedir.

TAMİR YARDIMCISI durumuna geçen bir düğüm kendi depoladığı parçayı e-MBR kodlama şemasına özel kodlama matrisi satırının devriği ile çarpar ve sonu-

cunu TAMİR EDİCİ durumunda olan düğüme gönderir.

VERİ TOPLAYICI; sistemde asıl dosyanın oluşturulması istendiğinde HAZIR durumundaki bir düğüm seçtiği k farklı düğüme *GERİÇATIM* mesajı gönderir ve VERİ TOPLAYICI durumuna geçer. Bu k düğümden *GERİÇATIMCEVABI* mesajlarını bekler ve hepsini aldığında verinin geri çatılması işlemini gerçekleştirir. VERİ TOPLAYICI düğüm bu işlemi gerçekleştirdikten sonra GERİ ÇATIM YARDIMCISI durumundaki düğümlere SON mesajı göndererek kendisi HAZIR durumuna geçer. Görüldüğü üzere, HAZIR durumundaki düğüm depolama siteminde her türlü işleme hizmet etmek üzere hazır bir şekilde beklemektedir.

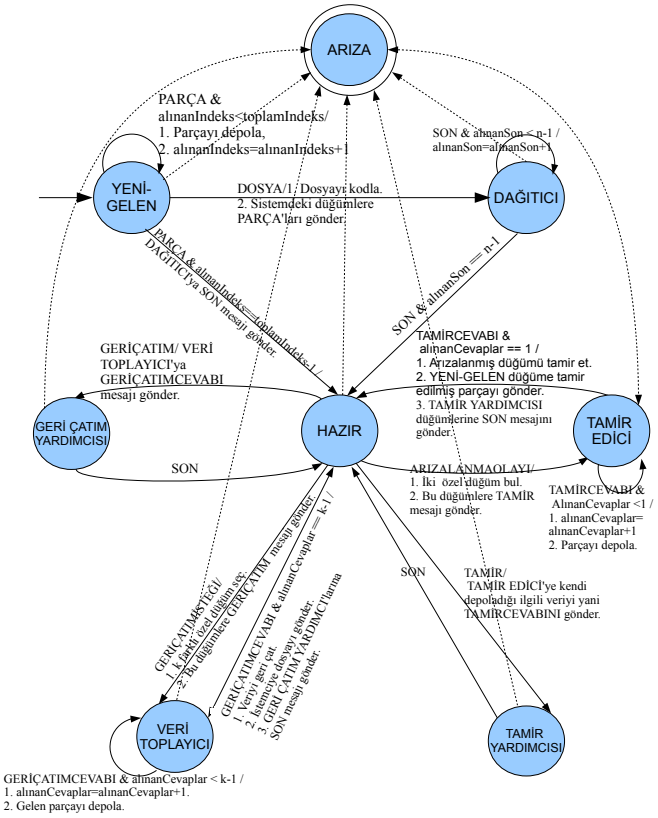
GERİ ÇATIM YARDIMCISI; HAZIR durumundaki bir düğüm *GERİÇATIM* mesajı aldığında kendi depoladığı parçayı *VERİ TOPLAYICI*'ya gönderirerek GERİ ÇATIM YARDIMCISI durumuna geçer ve *VERİ TOPLAYICI*'dan *SON* mesajını aldığında HAZIR durumuna geçer.

ARIZA; herhangi bir nedenle arızalanmış düğümün durumudur. Tüm durumlardan bu duruma arızalanma ile geçiş bulunmaktadır. Geçişi gerektiren mesajlar sonlu durum makinalarının anlaşılırlığı açısından belirtilmemiştir.

HSRC kodlama şeması kullanılan bir depolama sisteminde bulunan bir düğümün sonlu durum makinası Şekil 3.8'de verilmiştir [1]. Genel yapı itibarıyla e-MBR kodlama şeması kullanan bir depolama sistemi ile aynı durumlara sahiptir. HSRC kodlama şeması kullanan bir düğümün sonlu durum makinası; orijinal verinin kodlanması, düğüm tamiri ve verinin geri çatılmasında istek gönderilecek düğümlerin seçiminde ve yine aynı işlemlerin düğüm içi süreçlerinde e-MBR kodlama şemasının sonlu durum makinasından ayrılmaktadır. Bu kodlama şemasında da düğümlerin durumları ARIZA, YENİ-GELEN, DAĞITICI, GERİ ÇATIM YARDIMCISI, HAZIR, TAMİR EDİCİ, VERİ TOPLAYICI, TAMİR YARDIMCISI olarak belirlenmiştir.

---

[1] Şekil 3.8'de ARIZA durumuna gelen oklar diğer oklarla karışmaması için kesiklidir.

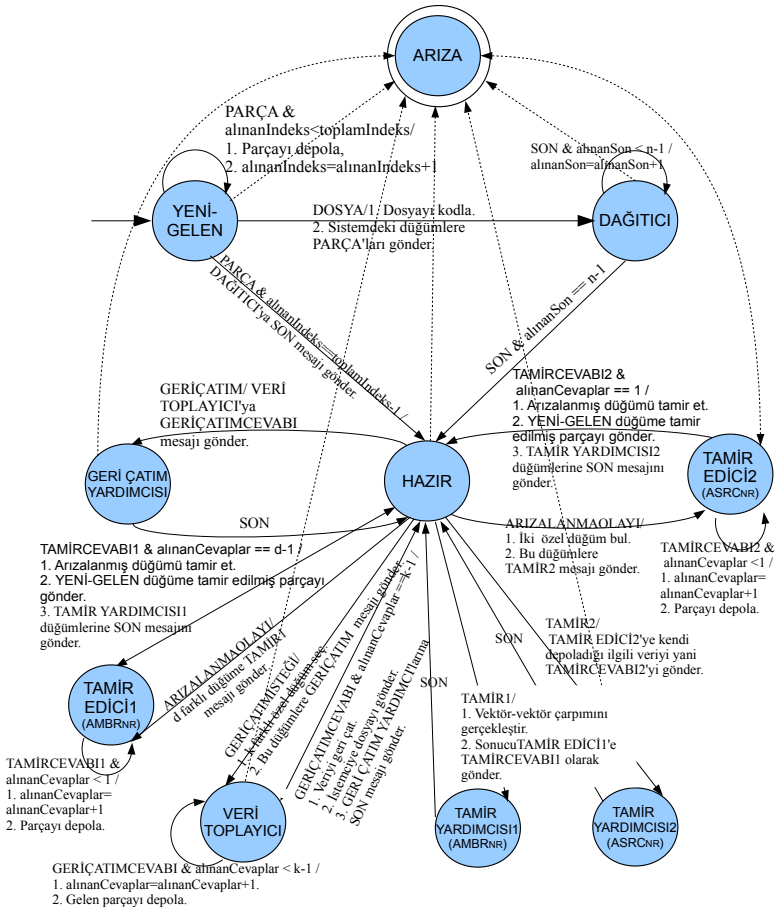


Şekil 3.8. HSRC kodlama şemasını kullanan bir dağıtık depolama sisteminde çalışan bir düğümün sonlu durum makinası.

Şekil 3.8'den de görüldüğü gibi verinin geri çatılması ve düğüm tamiri işlemlerinde yardımcı düğüm seçiminde e-MBR kodlama şemasına göre farklılıklar bulunmaktadır. Buna göre HAZIR durumundaki bir düğüm, *GERİÇATIMİSTEĞİ* mesajı aldığında HSRC kodlama şemasının yapısı gereği sağlanması gereken koşullara sahip k farklı düğüm seçip GERİÇATIM yardımcısı olarak atamaktadır. Ayrıca düğüm tamirinde d farklı yardımcı düğüm yerine iki farklı yardımcı düğümden yararlanılmaktadır. TAMİR YARDIMCISI düğümler tamir etme işlemi için kendi depoladıkları veriyi iletmek dışında fazladan bir işlem gerçekleştirmez.

HMBR kodlama şeması kullanan bir dağıtık depolama sisteminde ise iki farklı düğüm tamiri ve verinin geri çatılması yöntemi bulunmaktadır. Ancak kümeleme tabanlı dağıtık depolama sistemi çalışması kapsamında verinin geri çatılması yöntemlerinin bir tanesi olan $AMBR_{DR}$ yöntemi gerçekleştirilmiş olup, düğüm tamiri yöntemlerinin ikisi de ($AMBR_{NR}$ ve$ASRC_{NR}$) uygulanmıştır. Bu nedenle, HMBR kodlama şemasının sonlu durum makinası diğer kodlama şemalarından daha fazla durumu barındırmaktadır. HMBR kodlama şemasının kullanıldığı bir depolama sis-

teminde düğümlerin sonlu durum makinası Şekil 3.9'da görülmektedir.



Şekil 3.9. HMBR kodlama şemasını kullanan bir dağıtık depolama sisteminde çalışan bir düğümün sonlu durum makinası.

Bu kodlama şemasını kullanan bir düğümün durumları ARIZA, YENİ-GELEN, DAĞITICI, GERİ ÇATIM YARDIMCISI, HAZIR, TAMİR EDİCİ1, TAMİR EDİCİ2, VERİ TOPLAYICI, TAMİR YARDIMCISI1 ve TAMİR YARDIMCISI2'dir. Bu kodlama şemasının $AMBR_{NR}$ yöntemi ile ilgili düğüm tamiri işlemleri TAMİR EDİCİ1, TAMİR YARDIMCISI1, $ASRC_{NR}$ yöntemi ile ilgili işlemleri ise TAMİR EDİCİ2, TAMİR YARDIMCISI2 durumları ile ifade edilmiştir.

Kümeleme tabanlı olmayan dağıtık depolama sisteminde her düğüm tek bir kodlama şemasına ait sonlu durum makinasını çalıştırmaktadır. Ancak yardımcı düğüm seçimleri ya da kodlanmış dosyanın parçalarının gönderimi ait olunan gruptaki düğümler ile sınırlandırılmıştır. Kümeleme tabanlı dağıtık depolama sisteminde ise her küme farklı bir kodlama şeması çalıştırdığı için her kümeye ait düğümler il-

gili kodlama şemasının sonlu durum makinasını çalıştırmaktadır. e-MBR, HSRC ve HMBR kodlama şemalarının kullanıldığı sistemlerde düğüm arızalanması olduğunda sisteme arızalanmış düğümün yerine yeni bir düğüm ekleyen bir sistem süreci bulunmaktadır.

3.5 Silinti Kodlarında Dosya Geri Çatma İşlemi İçin TCP Bağlantılarının Yönetimi

Klasik Reed-Solomon (Reed and Solomon, 1960) kodlama şeması dağıtık depolama sistemlerinde verilerin hata toleranslı bir şekilde saklanması için yaygın olarak kullanılır. Bu kodlama şemasında, bir dosya matris çarpımı yolu ile genişletilerek n parçaya bölünür. Her bir parça faklı bir bilgisayarda (düğümde) depolanır ve $n - k$ tane düğüm arızalanmasına kadar dosya yeniden elde edilebilir. Kodlanmış dosyanın yeniden elde edilmesi (geri çatılması) istendiğinde, depolanan dosya parçaları herhangi k düğümden ağ üzerinden iletişim kurularak ile istenir. Alınan parçalar üzerinde Reed-Solomon kod çözücü uygulanarak orjinal dosya yeniden elde edilir. Reed-Solomon kodlamanın dağıtık depolama sistemlerinde kullanılması Plank (1997) tarafından sunulan çalışmadan daha detaylı olarak incelenebilir. Dosyanın geri çatılması işleminde, düğümlerde saklanan dosya parçalarının eksiksiz bir şekilde alınması gerekmektedir. Bu yüzden bu işlem TCP protokolü ile gerçekleştirilmektedir. Dosyanın geri çatılması işleminde, k farklı düğümün, dosya parçalarını ağ üzerinden eş zamanlı göndermesi gerektiğinden, bu işlem ağda ciddi bir trafik oluşturabilmekte ve ağ tıkanıklığı durumundan etkilenebilmektedir. Ağda tıkanıklık meydana geldiğinde, alıcı tarafa ulaşmayan veri paketlerinin TCP protokolü uyarınca tekrar tekrar gönderilmesi tıkanıklığı daha da kötüleştirebilmektedir.

Bu tez çalışmasında, Reed-Solomon kodlama kullanan dağıtık depolama sistemlerinde dosya geri çatılması işleminde ağ tıkanıklığı yaşandığında, TCP bağlantılarını daha etkin kullanarak işlem süresini ve ağ trafiğini azaltan dağıtık bir yöntem önerilmektedir. Önerilen yöntemin kullanılmadığı durum “kontrolsüz” yöntem olarak isimlendirilmiştir. Kontrolsüz yöntemde her veri düğümü, dosya parçalarını göndereceği düğüm ile arasında uçtan uca TCP bağlantısı kurarak dosya parçalarını doğrudan göndermektedir. Önerilen yöntem ve kontrolsüz yöntem, ns-3 (ns-3, 2015) benzetim ortamında gerçekleştirilmiş ve karşılaştırmalı sonuçlar Bölüm 4.5’te verilmiştir. Bu çalışma bildiri olarak sunulmuştur (Haytaoglu and Dalkilic, 2015).

3.5.1 Önerilen yöntem

Ağda toplamda N ($N \geq n > k$) tane düğüm bulunmaktadır. Bir dosya Reed-Solomon (Reed and Solomon, 1960) kodlama ile kodlandıktan sonra, elde edilen her bir dosya parçası (toplamda n tane) farklı bir düğümde depolanmaktadır. Bu n düğüm, veri düğümü olarak adlandırılmıştır ve N düğüm arasından rastgele seçilmiştir. Diğer $N-n$ düğüm ise sıradan düğüm olarak adlandırılmaktadır. Dosyanın geri çatılma işlemini başlatan veri düğümüne veri toplayıcı düğüm denilmektedir. Geri çatma işlemi için k farklı düğümde depolanan dosya parçalarının bir araya getirilmesine ihtiyaç duyulmaktadır. Bunun için n veri düğümünün herhangi k tanesinden depoladığı dosya parçaları istenir. Dosya parçaları kendi içerisinde lime (*strip*) adı verilen numaralandırılmış daha küçük kısımlara bölünmüştür. Veri-toplayıcı düğüm k farklı düğümden aynı numaraya sahip limeleri aldıkça bunları Reed-Solomon kod çözücü işleminden geçirir (kod çözme işlemi tüm lime grupları için tekrarlanır). Yani veri-toplayıcı düğüm aynı lime numarasına ait $k-1$ tane limeye sahip olsa bile son limeyi elde etmeden o lime grubunu kod çözücüden geçiremez. Bu yüzden önerilen yöntemde lime numaralarının dengeli bir şekilde veri-toplayıcı düğüme gönderilebilmesi hedeflenmiştir.

Önerilen yöntem iki temel adımdan oluşmaktadır. İlk adımda veri düğümleri kendi aralarında bir hiyerarşi (ata-çocuk-torun) oluşturmaktadır. İkinci adımda ise hiyerarşi ilişkisi oluşturulmuş düğümler arasında paket gönderim işlemi başlatılmakta ve TCP bağlantıları yönetilmektedir. Bu adımda, bir ata düğümü çocuğundan veri paketi aldığında bu paketi kendi atasına göndermek için ilgili çocuğu için açtığı soketi kullanmakta ve ağın tıkanıklık durumuna göre farklı çocuklarından gelen paketleri iletmek için aynı soketi kullanabilmektedir. Böylece bu ata düğüm kendi çocuklarından üst seviyeye iletilen veri gönderim hızını sınırlayabilmektedir. Burada, TCP soketlerinde alıcı-gönderici pencerelerinin maksimum boyutunun kısıtlı olmasından yararlanılmaktadır. Çünkü, TCP soketlerinin pencere boyutları veri gönderim hızını kısıtlamaktadır. Daha çok TCP soketi kullanılması yeterli bant genişliğinde daha yüksek veri gönderim hızı demektir.

3.5.1.1 Düğümler arasında hiyerarşi belirleme

İlk adımda, ağdaki n veri düğümü içerisinden dosyanın geri çatılması işlemi için seçilen k düğüm arasında bir hiyerarşi oluşturulmaktadır. Bu hiyerarşi oluşturulurken Şekil 3.10'da gösterildiği gibi veri ve veri-toplayıcı düğümler bu hiyerarşinin bir parçası olmakta, öte yandan hiyerarşi oluşturmada bu hiyerarşinin bir parçası olmamasına karşın sıradan düğümlerden de yararlanılmıştır. Herhangi bir veri dü-

ğümünden bir *ÇOCUK* paketi alan sıradan bir düğüm, bu paketle ilgili herhangi bir işlem yapmadan; kendi yönlendirme tablosundan veri-toplayıcı düğüme bir paket gönderildiğinde paketin iletilmesi gereken sonraki düğümün adresini bulup o düğüme aldığı *ÇOCUK* paketini iletmektedir.

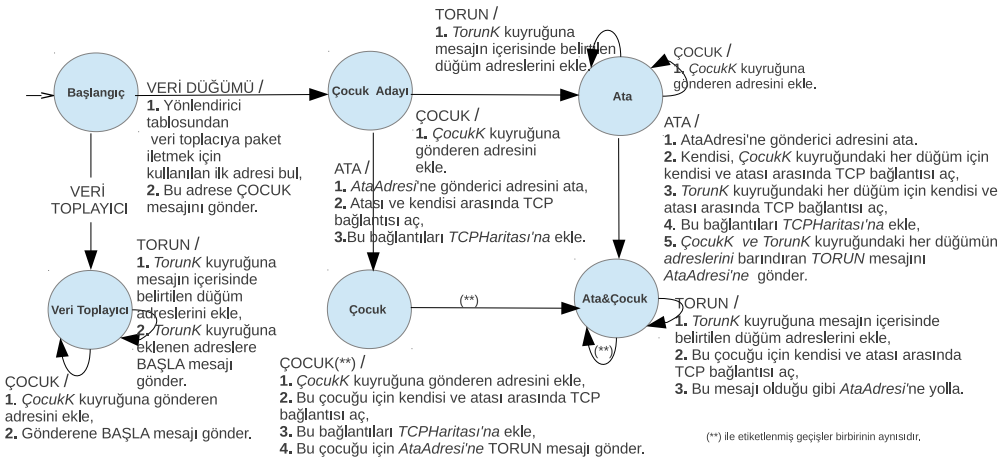
Veri ve veri-toplayıcı düğümlerin hiyerarşi belirleme algoritmasının sonlu durum makinası Şekil 3.10'da görülmektedir. Ağdaki veri düğümleri yerine göre veri toplayıcı ya da normal veri düğümü olabilmektedir. Hangi veri düğümünün veri-toplayıcı olacağı, merkezi bir otorite tarafından rastgele seçilmektedir ve bu otorite tarafından seçilen veri-toplayıcı düğümüne *VERİ TOPLAYICI* mesajı, diğer $k-1$ düğüme ise *VERİ DÜĞÜMÜ* mesajı gönderilmektedir.

VERİ DÜĞÜMÜ mesajı alan düğümler; veri-toplayıcı düğüme bir paket gönderilirken kullanılan bir sonraki adrese yönlendirici tablolarından bakarak, bu adrese *ÇOCUK* mesajı göndermektedir ve kendileri *Çocuk Adayı* durumuna geçmektedir. *Çocuk Adayı* durumundaki bir veri düğümü *ÇOCUK* mesajı alırsa, o mesajı gönderen veri düğümünün atası olmaktadır. Gönderen düğümü, kendi çocuklarını tuttuğu bir kuyruğa ekleyerek *Ata* durumuna geçmektedir. *Çocuk Adayı* durumundaki bir düğüm *ATA* mesajı alırsa, artık kendi atası belirlenmiş demektir ve *Çocuk* durumuna geçmektedir.

Çocuk durumundaki bir düğüm; *ÇOCUK* mesajı alırsa artık başka bir veri düğümünün atası olmuştur, bu düğüm için kendisi ve atası arasında TCP bağlantısı açar ve bu yeni çocuğunu bildirmek için atasına *TORUN* mesajı gönderir, *TCP-Haritasi* veri yapısına, yeni çocuğundan gelen paketleri iletirken kullanmak için soket-çocuk ilişkisini kaydeder ve *Ata&Çocuk* durumuna geçer. *Ata* durumundaki bir düğüm *TORUN* veya *ÇOCUK* mesajı alırsa ilgili kuyruğa aldığı mesajın içindeki bilgileri ekler. Bu düğüm, eğer *ATA* mesajı alırsa kendisi ve çocuklarını ve torunlarını tuttuğu kuyruklardaki her bir düğüm için atasıyla arasında farklı bir TCP soketi açar, bunları *TCPHaritasi*'na kaydeder ve kendi atasına bu kuyruklardaki düğümlerin kimliklerini belirten bir *TORUN* mesajı göndererek *Ata&Çocuk* durumuna geçer [1]. *Ata&Çocuk* durumundaki bir düğüm de *ÇOCUK* veya *TORUN* mesajı aldığında benzer işlemler gerçekleştirerek *Ata&Çocuk* durumunu korur. *Veri Toplayıcı* durumundaki düğüm ise *ÇOCUK* ya da *TORUN* mesajı alırsa, mesaj tipine göre ilgili kuyruğuna gelen düğüm kimliklerini ekler ve bu düğümlere *BAŞLA* mesajı gönderir. *BAŞLA* mesajı alan düğümler TCP bağlantı kontrolü algoritması olan Ek 3'teki

[1] Bir ata düğümünün çocuğundan alt seviyedeki (çocuğu ya da çocuğunun çocuğu vb.) veri düğümleri onun torunudur.

Algoritma 1'i de çalıştırmaya başlar. Ancak bu düğümler aynı zamanda düğümler arası hiyerarşi belirleme algoritmasını çalıştırmaya devam eder.



Şekil 3.10. Veri ve veri-toplayıcı düğümlerinin hiyerarşi belirleme sonlu durum makinası.

3.5.1.2 TCP bağlantılarının yönetimi

Geri çatma işleminde, k farklı düğümden elde edilen aynı numaralı limeler bir şeridi (*stripe*) oluşturur. Aynı şeritteki limeler toplu olarak sonlu cisim (*finite field*) işlemlerinden geçmektedir (Plank et al., 2009). *BAŞLA* mesajını alan düğümler, kendi depoladıkları dosya parçasını kendi atalarına göndermeye başlar. Bu düğümler, veri hattı katmanının (*data link layer*) bir seferde gönderebileceği maksimum paket boyutunun sınırlı olmasından (Kurose and Ross, 2005) dolayı bu parçaları küçük paketlere bölerek gönderebilmektedir. Bu düğümler, dosya parçasının bir kısmını içeren bir paketi gönderirken paketin içerisine gönderilen limelerin numaralarını da eklemektedir. Bu paketler *LİME* mesajı olarak adlandırılmıştır. Veri düğümleri, aldıkları *BAŞLA* mesajı ile limeleri gönderirken bir yandan da Ek 3'teki Algoritma 1'i çalıştırmaktadır. Algoritma 1'i çalıştıran bir düğüm başka bir düğümden *LİME* mesajı alırsa, aldığı *LİME* mesajları ile ilgili bilgilerin tutulduğu *AkışK* kuyruğunu limenin asıl kaynağı ve lime numarası bilgisiyle günceller. *LİME* mesajını alan düğüm, veri-toplayıcı düğüm ise, bu düğüm f_1 fonksiyonunu çalıştırır. f_1 fonksiyonunda mesajın içerisindeki limeler ilgili yere kopyalanır; aynı lime numarasından kaç tane alındığını gösteren değişken bir arttırılır, aynı numaraya sahip k tane lime alınmışsa Reed-Solomon kod çözücüsü çalıştırılır ve elde edilen sonuç geri çatılan dosyaya yazılır. Bu kod çözme işlemi tüm lime grupları (şeritler) için gerçekleştirilmişse ağdaki düğümlere, dosya geri çatma işleminin tamamlandığını

ifade eden *SON* mesajı gönderilir [1]. *LİME* mesajını alan düğüm bir veri düğümü ise mesajı ilk gönderenin adresini alır, bu adresin paketlerini göndermek için kullandığı soketi *TCPHaritası* veri yapısından bularak, o soket üzerinden *LİME* mesajını atasına iletir.

Bir veri düğümü ya da veri-toplayıcı düğümde paket kuyruğu taşması olursa, bu düğüm bir *PAKET İPTALİ* olayı üretir ve *LİME* paketi aldıkça güncellediği *AkışK* kuyruğunu kullanarak farklı düğümler tarafından en son alınan lime numaralarının ortalamasını ve standart sapmasını sırasıyla f_2 ve f_3 fonksiyonunda hesaplar. Ortalamadan daha yüksek lime numarasına sahip çocukları, torunları varsa sırasıyla daha önce uyarılmamış çocuklarına ya da torunlarının paketini ileten uyarılmamış çocuklarına *YAVAŞLA* mesajı gönderir. Bu mesajı gönderdiği düğümleri bir daha belirli bir süre uyarmamak için *UyarılanK* kuyruğuna atmaktadır. Çünkü, paket kuyruğu taşması yaşayan bir düğümün yakın zamanda bir daha taşma yaşama ihtimali yüksektir. Bu süre bitiminde daha önceden uyarılmış düğümleri yeniden uyarabilmek için süre tutulması işlemi de burada yapılmaktadır. Bu süre bittiğinde düğüm *ZAMAN AŞIMI* olayı üretir. Bu olayın oluştuğu düğüm *UyarılanK* kuyruğunu boşaltır.

YAVAŞLA mesajı alan bir düğüm, ağ tıkanıklığının giderilmesine yardımcı olmak üzere gönderdiği paket trafiğini azaltmak için mevcut soketlerinin bazılarını birleştirerek aktif olarak kullandığı soket sayısını azaltır. Veri-toplayıcı düğüm aynı şeride ait k tane limenin hepsini elde edemedikçe kod çözme işlemini gerçekleştiremediği için aldığı limeler arasında yüksek numaralı olanları, o şeride ait k tane limenin hepsini elde edene kadar bekletmektedir. Bu nedenle *YAVAŞLA* mesajı alan düğüm göndereceği paket trafiğini azaltırken kendi üzerinden geçen yüksek numaralı limelere ait TCP bağlantılarını birleştirmek için harekete geçer. Böylece dosya geri çatma işlemi için kullanılan toplam sürede artış olmaması hedeflenmiştir. Yani, aynı şeride ait k tane limenin hepsini elde edilmedikçe kod çözme işlemi gerçekleştirilemediği için yüksek numaralı limeye sahip bağlantıların bekletilmesi kod çözme işlemi için toplamda geçen süreyi uzatmayacaktır. Bu düğüm kendisi, çocukları ve torunlarının paketlerini göndermek için kullandığı soketlerin bilgisini tutan *TCP-Haritası*, açtığı soketleri kullanan düğümlerin gönderdiği *LİME* paketlerinin son bilgilerinin tutulduğu *AkışK*; çocuk ve torunların bilgilerinin tutulduğu *ÇocukK, TorunK* parametrelerini alan f_4 fonksiyonunu çalıştırır. f_4 fonksiyonunda bu düğümün açtığı her bir soketin içinden geçen farklı düğümlere ait akışların son lime numaralarının ortalaması alınır ve her bir soketten kaç farklı düğümün (çocuğu, ken-

[1] Şekil 3.10'un anlaşılırlığı açısından bitiş durumları gösterilmemiştir. Her durumdan *SON* mesajı ile bitiş durumuna geçilebilmektedir. İki algoritma da bu mesajla sonlanmaktadır.

disi veya torunu) paket akışının geçtiği hesaplanır. Ardından, bu düğümün soketleri önce küçükten büyüğe içerisinden geçen farklı akış sayısına göre sıralanır. En küçük farklı akış sayısına sahip birden fazla soket varsa bu soketler lime numarası ortalamalarına göre büyükten küçüğe sıralanır. f_4 fonksiyonun çalıştırılması neticesinde sıralanmış soketlerin ilk ikisi (*sSoket*) alınarak f_5 fonksiyonuna gönderilmektedir. Mevcut düğümün ikinci soketine atanmış olan çocuğunun, torunun, ya da kendisinin paketlerinin gönderimi için artık ilk soket kullanılacaktır. Yani, f_5 fonksiyonunda ikinci sokete atanan düğümlerin hepsi ilk sokete atanmaktadır.

4. BULGULAR

Bu bölümde tez boyunca gerçekleştirilen beş çalışma hakkında elde edilen bulgular ayrı alt başlıklarda verilmiştir. Bu çalışmalardan ilki MDS silinti kodlarını kullanan dağıtık depolama sistemleri için topoloji farkındalıklı bir çalışmadır. İkincisi melez bir kodlama şeması olan Homomorfik Minimum Bant Genişliği Tamir (HMBR) kodları çalışmasıdır. Üçüncü alt başlıkta Homomorfik Minimum Depolama Tamir (HMSR) kodları çalışması ile ilgili olarak elde edilen çalışmalar bulunmaktadır. Dördüncü alt başlıkta ise kümeleme tabanlı dağıtık depolama sistemi çalışması hakkında elde edilen bulgular görülmektedir. Beşinci ve son alt başlıkta ise MDS silinti kodlama kullanan dağıtık depolama sistemlerinde verinin geri çatılması işleminde oluşan ağ tıkanıklığı ele alınmaktadır.

4.1 MDS Silinti Kodlama Kullanan Dağıtık Depolama Sistemleri İçin Topoloji Farkındalıklı Bir Çalışma İle İlgili Elde Edilen Bulgular

MDS silinti kodlama kullanan dağıtık depolama sistemlerinde verinin geri çatılması, düğüm tamirinin başlatılması ve güncelleme işlemlerinin iletişim maliyetleri hakkında elde edilen kuramsal ve deneysel bulgular aşağıda sırasıyla incelenmiştir.

4.1.1 Kuramsal bulgular

Bu bölümde MPA ve STA yaklaşımlarının verinin geri çatılması, düğüm tamiri işlemlerinin başlatılması ve veri güncelleme işlemlerinin iletişiminde kullandıkları mesaj ve zaman karmaşıklıklarının kuramsal analizi gerçekleştirilmiştir.

4.1.1.1 Verinin geri çatılması ve düğüm tamiri işlemlerinin başlatılmasının iletişim maliyeti üzerine kuramsal bulgular

Bu alt bölümde verinin geri çatımının ve düğüm tamirinin başlatılması ile ilgili elde edilen bulgular kuramsal olarak incelenecektir. Verinin geri çatılması ve düğüm tamiri işlemlerinin başlatılması için kullanılan STA ve MPA yaklaşımlarının bu işlemlerde kullandığı mesaj sayılarının ve harcadığı sürelerin kuramsal analizleri yapılmıştır. Analizlerde kullanılan parametreler Çizelge 4.1'de açıklanmıştır. Önerilen iletişim şemaları verinin geri çatılmasının ve düğüm tamir etme işlemlerinin başlatılması için aynıdır. Bir başka ifadeyle, bu bölümdeki analizler bu iki işlem için de geçerlidir.

Çizelge 4.1. Kuramsal analizlerde kullanılan değişkenler

Değişken	Anlamı
d	Bir kenardaki yayılma gecikmesi
t	Bir paketin iletim süresi (Bir paketin boyutu MTU'dan daha küçük ya da MTU'ya eşittir.)
N	Çizgedeki toplam düğüm sayısı
k	Düğüm tamiri ve verinin geri çatılması için gerekli olan hedef düğüm sayısı
n	Çizgedeki toplam depolama düğümü sayısı
δ	Çizgedeki ortalama düğüm derecesi
L_k	k tane hedef düğüm içeren bir çoğa gönderim ağacındaki kenar sayısı

I. Steiner Ağaç Yaklaşımının (STA) Analizi, Verinin geri çatılması ve düğüm tamiri işlemlerinin başlatılmasında STA yaklaşımı kullanıldığında, bu yaklaşım kullanıldığındaki iletişimin mesaj ve zaman karmaşıklık analizleri aşağıda yapılmıştır.

Mesaj Karmaşıklığı: Bu yaklaşımda iletişimin mesaj karmaşıklığı Dolagh and Moazzami (2011) tarafından önerilen algoritmanın ürettiği Steiner ağacının yapısına bağlıdır.

Ön Kuram 4.1. *N düğümlü bir güç kanunu çizgesinde (İng. Power Law Graph), herhangi iki düğüm arasındaki uzaklık $E(L_u)$ yaklaşık olarak $O(log(N)/log(\tilde{d}))$'dır. Burada $\tilde{d}$ düğümlerin derecelerinin beklenen değerlerinin karelerinin ağırlıklı ortalamasıdır (Chung and Lu, 2003). (Analizler boyunca bu kuramdaki koşulların bu çalışmadaki ağlar tarafından sağlandığı varsayılmıştır.)*

Ön Kuram 4.2. *Chuang-Sirbu kanununa (Chuang and Sirbu, 2001) göre m tane hedef düğümü olan bir çoğa-gönderim ağacının (multicast tree) beklenen kenar sayısı $E(L_m)$, $O(E(L_u)m^{0,8})$ tarafından üstten sınırlandırılmıştır. Burada $E(L_u)$, bir teke gönderim (unicast) iletişim yolu için beklenen kenar sayısını ifade etmektedir.*

Teorem 4.1. *Dolagh and Moazzami (2011) tarafından önerilen algoritma kullanılarak oluşturulan ağaçların, ortaya çıkma olasılıklarının eşit dağıldığını varsaydığımızda, STA yaklaşımının bir tane çoğa gönderim iletişiminde harcadığı ortalama mesaj sayısı $O(k^{0,8}\frac{log(N)}{log(\tilde{d})})$ ile üstten sınırlandırılmıştır.*

İspat. *Bir tane çoğa gönderim iletişiminde, toplam mesaj sayısı bahsedilen algoritma (Dolagh and Moazzami, 2011) ile oluşturulan Steiner ağacının kenar sayısına*

eşittir. Ön-kuram 4.1'den iki düğüm arasındaki beklenen uzaklık $\frac{log(N)}{log(\tilde{d})}$ *olduğundan ve Ön-kuram 4.2'ten* L_u *ve* L_k *arasındaki oran* $k^{0,8}$ *olduğu için, bu yaklaşım kullanıldığında oluşan mesaj sayısı* $O(k^{0,8}\frac{logN}{log(\tilde{d})})$*' dir.*

Teorem 4.2. *Steiner ağaç yaklaşımı ile verinin geri çatılması ve düğüm tamirinin başlatılması için en fazla* $O(N)$ *mesaj gerekmektedir.*

İspat. *STA yaklaşım algoritması (Dolagh and Moazzami, 2011) tarafından üretilen bir çoğa-gönderim ağacının* k *tane hedef düğüm ve* z *tane ara düğümden oluştuğunu varsayalım. Ağaç yapısından dolayı toplam mesaj karmaşıklığı ağacın kenar sayısına eşittir. Burada, mesaj karmaşıklığının üst sınırını bulmak maksimum düğüm sayılı bir çoğa-gönderim ağacını bulmaya eşdeğerdir. Bu durumda, mesaj karmaşıklığı* $O(N)$ *tarafından üstten sınırlandırılmıştır.*

STA yaklaşımı için mesaj karmaşıklığı analizi burada tamamlanmış olup aşağıda zaman karmaşıklığı incelenmektedir.

Zaman Karmaşıklığı: Bir çoğa gönderim iletişiminde harcanan toplam süre, kaynak düğümün mesaj göndermeye başladığı an ile mesajı en son alan düğümün mesajı alma anı arasında geçen süre olarak hesaplanır. STA yaklaşımında kullanılan süreyi hesaplamak için kullanılan çoğa gönderim ağacının yüksekliğinin beklenen değerinin bulunması gerekmektedir.

Ön Kuram 4.3. *Steiner ağaç yaklaşım algoritmasında (Dolagh and Moazzami, 2011) oluşturulan Steiner ağaçlarının beklenen yüksekliği;* $f(k)$*,* k*'nın monoton artan bir fonksiyonu,* k *hedef düğüm sayısı,* N *ise çizgedeki toplam düğüm sayısı olmak üzere* $O(\frac{klog(N)}{log(\tilde{d})f(k)})$*'dır.*

İspat. *Dolagh and Moazzami (2011) tarafından önerilmiş olan Steiner ağaç üreten yaklaşım algoritması, başlangıçta ağacın maliyetini arttırabilecek (kenar sayısı açısından) kenarları çizgeden silmektedir. Ardından, kaynak ve hedef düğümleri iki gruba ayırmaktadır. İlk grupta sadece (kaynak düğüm) bir depolama düğümü vardır ve geriye kalan* $k-1$ *tane depolama düğümü ikinci gruptadır. Ardından, ilk döngüde algoritma ilk gruptaki düğüme en yakın depolama düğümünü bulur. Sonra bu düğüm ikinci gruptan çıkartılarak, kaynak düğümüyle arasındaki en kısa yoldaki düğümler ile birlikte ilk gruba eklenir. İkinci döngüde, algoritma ilk gruptaki düğümler ile ikinci gruptaki düğümler arasındaki en yakın düğüm çiftini seçer. Ardından, bu iki düğüm arasındaki düğümler ikinci düğüm de dahil olmak üzere ilk gruba eklenir ve bu düğümlerden ikinci gruba giren düğümler ikinci gruptan çıkartılır. Algoritma bu şekilde ikinci düğüm grubunda düğüm kalmayana kadar devam eder. Algoritma* $(k-1)$ *döngüde tamamlanır. Sonuç olarak, her döngüde eklenen en kısa yol Steiner ağacının dallarını oluşturmaktadır.*

N düğümlük bir çizgede herhangi iki düğüm arası beklenen uzaklık Ön kuram 4.1'den $O(log(N)/log(\tilde{d}))$'dir. Çizgeden rastgele k tane düğümün seçildiği varsayılsın (ilk döngüdeki durum). Kaynak düğüm ile geri kalan $k-1$ düğüm arasındaki en kısa yolun beklenen değeri $O(log(N)/log(\tilde{d}))$'den küçük olmalıdır. Bu durumda en kısa yol $k-1$ tane en kısa yol arasından birisi olup bu yolların uzunluğunun beklenen değeri ise $O(log(N)/log(\tilde{d}))$'dir [1]. En kısa yolların uzaklıkları eş dağılıma sahip olmadığı için $k-1$ düğüm çifti (biri kaynak düğüm diğer $k-1$ düğüm ise geriye kalan depolama düğümleri arasındadır) arasındaki beklenen en kısa yol $O(log(N)/log(\tilde{d}))$'den küçüktür. Dahası, $k-1$ çift arasındaki en kısa yol uzunluğu $f(k)$ ile ters orantılıdır. $f(k)$ fonksiyonu k'nın monoton artan bir fonksiyonudur [2]. Diğer bir deyişle, bir grup düğümün içindeki en kısa yolun uzunluğu düğüm çiftlerinin sayısı k ile ters orantılıdır. Aynı durum, diğer döngüler için de geçerli olup en kısa yol herhangi bir döngüde, o döngüde oluşan çift sayısının bir fonksiyonu ile ters orantılıdır.

STA yaklaşım algoritması tarafından oluşturulan Steiner ağacının yüksekliğinin beklenen değeri şu şekilde hesaplanabilir. İlk en yakın düğüm çifti seçildiğinde (1. döngüde), oluşan ağacın bir kaynak ve bir hedef düğümü bulunmakta olup bu ağacın beklenen yüksekliği $O(\frac{log(N)}{log(\tilde{d})f(k)})$'dır. Sonraki döngülerdeki en yakın çift seçimlerinde ağacın yüksekliği en fazla bir sonraki en kısa yolların uzunluğu kadar artabilmektedir. Ayrıca, i tane hedef düğüme sahip bir ağaçtaki kenar sayısı Ön Kuram 4.1 ve Ön Kuram 4.2'den $O(i^{0.8\frac{log(N)}{log(\tilde{d})}})$ ile sınırlandırılmıştır. Bu da demek oluyor ki $i>1$ iken i'inci döngüde oluşturulan ağaç $O((i)^{0.8\frac{log(N)}{log(\tilde{d})}})$ mertebesinde düğüme (dolayısı ile $O((i)^{0.8\frac{log(N)}{log(\tilde{d})}})$ kenara) sahiptir. Buna göre, i'inci döngü başladığında $O((i)^{0.8\frac{log(N)}{log(\tilde{d})}}\times(k-i))$ tane olası düğüm çifti vardır ve algoritma bu $O((i)^{0.8\frac{log(N)}{log(\tilde{d})}}\times(k-i))$ çiften en yakınını bulmaya çalışmaktadır. Algoritmanın i'inci döngüsü başladığında düğüm çifti sayısının mertebesi aşağıda verilmiştir:

$$P(i)=\begin{cases}O((i)^{0.8\frac{log(N)}{log(\tilde{d})}}\times(k-i)), & i>1,\ i<k,\ i\in\mathbb{N}^{+}\\ k-1, & i=1\end{cases} \tag{4.1}$$

$i>1$ iken $P(i)$ içbükey bir fonksiyondur, çünkü bu fonksiyonun i'ye göre ikinci türevi negatiftir. $i>1$ için $P(i)$ fonksiyonunda i değeri $[2,k-1]$ aralığında ol-

---

[1] *Sample Mean* teoreminden bilinmektedir ki, bir kümenin alt kümesinde bir parametrenin beklenen değeri, ilgili parametrenin o kümenin tamamındaki beklenen değerine eşittir.

[2] $k=2$ olduğu durumda, beklenen en kısa yol $O(logN/log(\tilde{d}))$ ile sınırlandırılmıştır, ama eğer $k=500=N$ ise beklenen en kısa yolun değeri $O(1)$ ile sınırlandırılmıştır.

duğu için $P(i)$ fonksiyonunun minimum değeri $P(2)$ ya da $P(k-1)$'dedir. $P(2)$, $O(k\frac{log(N)}{log(\tilde{d})})$ ile sınırlandırılırken, $P(k-1)$ ise $O(k^{0.8}\frac{log(N)}{log(\tilde{d})})$ ile sınırlandırılmaktadır. Öte yandan $i=1$ için bu mertebe $O(k)$'dadır. Bu yüzden N'in yüksek değerleri için en küçük değer $P(1)$'de elde edilmektedir (N'nin düzeyinin $\tilde{d}^{k^{0.2}}$'den daha yüksek olduğu zaman.)

i'inci döngüde oluşturulmuş olan ağacın yüksekliği $H(i)$ ile ifade edilsin, $(i+1)$'inci en yakın çift seçiminde, yeni eklenecek yol mevcut ağacın herhangi bir seviyesine eklenebilir. Yeni eklenen yol ağacın köküne en uzak olan seviyede ise ağacın yüksekliğini $O(\frac{log(N)}{log(\tilde{d})f(P(i+1))})$ kadar arttırabilir. Ancak yeni eklenecek düğüm ağacın derinliğinden bir eksik seviyedeki bir düğümden eklenirse ağacın yüksekliğini $O(\frac{log(N)}{log(\tilde{d})f(P(i+1))}-1)$ düzeyinde arttırabilir ve ağacın diğer seviyeleri için bu değer birer birer azalmaktadır. Bu şekilde $(i+1)$'inci döngüde ağacın yüksekliğindeki artış miktarı aşağıda hesaplanmıştır:

$$O(E(H(i+1)-H(i))) = O(\sum_{l=1}^{\frac{log(N)}{log(\tilde{d})f(P(i+1))}} \frac{1}{H(i)}l) \tag{4.2}$$

$$\begin{aligned} O(E(H(i+1)-H(i))) = O(&\frac{1}{H(i)}\frac{log(N)}{log(\tilde{d})f(P(i+1))} \\ &(\frac{log(N)}{log(\tilde{d})f(P(i+1))}+1)/2) \end{aligned} \tag{4.3}$$

Sonuç olarak, i'inci döngüde $H(i+1)$'in beklenen değeri aşağıdaki gibidir:

$$O(H(i+1)) = O(H(i)) + O(\sum_{l=1}^{\frac{log(N)}{log(\tilde{d})f(P(i+1))}} \frac{1}{H(i)}l) \tag{4.4}$$

burada $O(H(0))=0$, $O(H(1))=O(\frac{log(N)}{log(\tilde{d})f(P(1))})$'dir.

Ağacın yüksekliğinin üst sınırının bulunması istendiği için $i>1$ iken $P(i+1)$ yerine $P(1)$ kullanılabilir. (Çünkü tüm $P(i)$'ler arasında $P(1)$ minimum düzeye sahiptir.) Diğer bir deyişle, eğer Eşitlik (4.4)'te toplam sembolünün içinde $P(i+1)$ yerine $P(1)$ konulursa üst sınır halen korunmaktadır, çünkü bu değer ağacın yüksekliğini arttırmaktadır. Böylece $i+1$'inci döngüde oluşturulan ağacın beklenen yüksekliği aşağıdaki gibi hesaplanabilir:

$$O(H(i+1)) = O(H(i)) + O(\sum_{l=1}^{\frac{log(N)}{log(\tilde{d})f(P(1))}} \frac{1}{H(i)} l) \tag{4.5}$$

$$O(H(i+1)) = O(H(i)) + O(\sum_{l=1}^{\frac{log(N)}{log(\tilde{d})f(k)}} \frac{1}{H(i)} l) \tag{4.6}$$

$$O(H(i+1)) = O(H(i)) + O(\frac{1}{H(i)}(\frac{log(N)}{log(\tilde{d})f(k)})(\frac{log(N)}{log(\tilde{d})f(k)}+1)/2) \tag{4.7}$$

burada yine $O(H(0)) = 0$, $O(H(1)) = O(\frac{log(N)}{log(\tilde{d})f(k)})$*'dir.*

Eşitlik (4.7)'yi sadeleştirmek için $\frac{log(N)}{log(\tilde{d})f(k)}$ *yerine* a *ve* $(\sum_{l=1}^{\frac{log(N)}{log(\tilde{d})f(k)}} l)$ *yerine* c *koyalım. Böylece,* $k-1$*'inci döngüden sonra oluşan ağacın beklenen yüksekliğinin mertebesi aşağıdaki gibi ifade edilebilir.*

$$H(k-1) = a + \frac{c}{a} + \underbrace{\frac{c}{a+\frac{c}{a}} + \frac{c}{a+\frac{c}{a}+\frac{c}{a+\frac{c}{a}}} + \dots}_{(k-3)\ tane\ terim} \tag{4.8}$$

Eşitlik (4.8)'de i*'inci terimin her zaman* $(i+1)$*'inci terimden yüksek olduğu açıktır (*$i > 0,\ i \in Z^+$*). Steiner ağacı yaklaşım algoritması tarafından üretilen* k *tane hedefe sahip bir çoğa gönderim ağacının beklenen yüksekliğinin üst sınırı,* $\rho(k) = H(k-1)$*,* $k-1$ *tane döngü işletildikten sonra aşağıdaki gibi gösterilebilir.*

$$\rho(k) = a + \frac{c}{a} + (k-3)\frac{c}{a+\frac{c}{a}} \tag{4.9}$$

$$\rho(k) = O(\frac{log(N)}{log(\tilde{d})f(k)} + \frac{log(N)}{log(\tilde{d})f(k)} + \frac{(k-3)log(N)}{log(\tilde{d})f(k)}) \tag{4.10}$$

$$\rho(k) = O(\frac{log(N)k}{log(\tilde{d})f(k)}) \tag{4.11}$$

Teorem 4.3. *Toplamda* N *tane düğüme sahip bir çizgede* k *tane hedef düğüm için STA yaklaşımında harcanan süre* $O(k\frac{log(N)}{log(\tilde{d})f(k)}(t+d))$ *ile sınırlandırılmıştır. Burada* $f(k)$ k*'nın monoton artan bir fonksiyonudur.*

İspat. *Steiner ağacın yüksekliğinin beklenen değeri Ön Kuram 4.3'ten* $\rho(k) = O(k\frac{log(N)}{log(\tilde{d})f(k)})$ *ile üstten sınırlandırılmıştır. Bir paketin hatta basım süresi her düğüm için* t*, yayılma gecikmesi ise her kenar için* d *olarak temsil edilmiştir. Bir paketin hedefe ulaşması için, kaynak ve hedef arasındaki yolun her kenarında hatta basım süresi (t) ve yayılma gecikmesi (d) süreleri kadar zaman geçer. Yani, verinin geri çatılması ve düğüm tamirinin başlatılması için Steiner ağaç yaklaşımında beklenen değer* $O(\rho(k)(t+d))$ *ile sınırlandırılmıştır.*

Teorem 4.4. *STA yaklaşımının kullandığı süre en fazla* $O(N(t+d))$*'dir.*

İspat. *Bu yaklaşımda en kötü durumda oluşan Steiner ağacındaki düğümler bir zincir gibi dizilmiştir ve kaynak düğüm ile en uzak hedef düğüm arasında* $N-1$ *tane kenar bulunmaktadır. Sonuç olarak oluşabilecek en kötü zaman karmaşıklığı da* $O(N(t+d))$*'dir.*

II. Çoklu En Kısa Yol Yaklaşımının (MPA) Analizi Bu kısımda, verinin geri çatılması ve düğüm tamiri işlemlerinin başlatılmasında çoklu en kısa yol yaklaşımının kullandığı mesaj ve zaman karmaşıklığı analizi incelenmiştir.

Mesaj Karmaşıklığı: Önerilen yaklaşımın mesaj karmaşıklığı, kaynak düğüm ile hedef düğümler arasındaki en kısa yolların kenar sayılarının toplamına eşittir.

Teorem 4.5. *Çoklu en kısa yol yaklaşımının bir çoklu teke gönderim iletişimi için harcadığı ortalama mesaj sayısı üstten* $O(k\frac{log(N)}{log(\tilde{d})})$ *ile sınırlıdır.*

İspat. *Herhangi iki düğüm arasındaki ortalama kenar sayısı Ön-kuram 4.1'den* $O(\frac{log(N)}{log(\tilde{d})})$ *ve hedef düğüm sayısı* k *olduğu için, çoklu en kısa yol yaklaşımında kullanılan ortalama mesaj sayısı* $O(k(\frac{log(N)}{log(\tilde{d})}))$ *ile sınırlandırılmıştır.*

Teorem 4.6. *MPA yaklaşımının kullandığı mesaj sayısı en fazla* $O(kN)$*'dir (*$N > k$*'dır.).*

İspat. *Bir kaynaktan bir hedefe en uzak yol* $N-1$ *kenar içermektedir. Aynı kaynaktan ikinci bir hedefe* $N-1$ *uzaklığında bir yol bulunamaz, çünkü ikinci hedef önceki* $N-1$*'lik yolun içerisinde olmak zorundadır. Bu durumda ikinci en uzak yol* $N-2$ *uzunluğunda olabilmektedir. Üçüncü en uzak yol ise* $N-3$ *kenardan oluşur ve hedef düğüm sayısı arttıkça bu şekilde devam eder. En uzak* k*. yol* $(N-k)$ *olarak hesaplanır. Sonuç olarak, bu yaklaşımdaki en kötü durum mesaj karmaşıklığı,* $(N-1)+(N-2)+\ldots+(N-k)$*,* $O(kN)$ *ile üstten sınırlandırılmıştır.*

Çoklu en kısa yol yaklaşımının zaman karmaşıklığı analizi aşağıda verilmiştir.

Zaman Karmaşıklığı: Önerilen yaklaşımın zaman karmaşıklığı, kaynak düğümle hedef düğümler arasındaki en kısa yolların kaynak düğüme bağlı ortak kenarların sayısına ve kaynak düğümle en uzak hedef düğüm arasındaki en kısa yolun uzunluğuna bağlıdır.

Ön Kuram 4.4. *Üs değeri $\beta > 3$ olan ve ortalama düğüm derecesi, δ, 1'den büyük olan güç kanununa uygun rastgele bir çizgenin çapı (diameter) $\theta(log(N))$'dir (Chung and Lu, 2003) (β değeri düğüm dereceleri ile ilgilidir.).*

Teorem 4.7. *MPA yaklaşımında bir çoklu teke gönderim oturumu için geçen ortalama süre $O(\frac{k}{\delta}t + log(N)(t + d))$ ile üstten sınırlandırılmıştır.*

İspat. *Kaynak düğüm, paketleri bitişik kenarlarından paralel olarak gönderdiği için, son mesajını göndermeden önce $(\frac{k}{\delta} - 1)t$ zaman geçecektir. MPA yaklaşımı ile oluşturulan bir çoklu teke gönderim ağacı düşünüldüğünde, kaynak düğümün ortalama derecesi δ olduğu için kaynak düğümün her bir kenarının altında ortalama $\frac{k}{\delta}$ hedef düğüm bulunmaktadır. Ön-kuram 4.4'ten çizgenin ortalama çapının $\theta(log(N))$ olduğu bilinmektedir. Yani bir çoklu teke gönderim iletişim oturumu için ortalamada geçen süre $(\frac{k-1}{\delta})t + log(N)t + log(N)d$ ile sınırlandırılmıştır. Bu süre de $O(\frac{k}{\delta}t + log(N)(t + d))$ ile sınırlandırılmıştır.*

Teorem 4.8. *Çoklu en kısa yol yaklaşımının harcağı süre en fazla $O(N(t + d))$ karmaşıklığına sahiptir.*

İspat. *En kötü durumda, kaynak düğüm ile tüm hedef düğümler arasındaki en kısa yolların ilk kenarları ortaktır. Bu durumda kaynaktan çıkan mesajlar seri olarak gönderilmek zorunda kalır. En son mesaj gönderilmeden önce $k - 2$ mesajın gönderilmesi gerekmektedir. Yani son mesaj gönderilmeye başlamadan $(k - 2)t$ süre geçer. En kötü durumda en son gönderilecek mesaj en uzak hedeftedir ($(N - 1)$ kenar sayısına sahiptir). Yani mesajın iletim süresi $(k-2)t+(N-1)t+(N-1)d$ olarak hesaplanır. Bu süre de $N > k$ olduğundan $O(N(t + d))$ ile sınırlandırılmıştır.*

STA yaklaşımı mesaj karmaşıklığı bağlamında genel olarak MPA yaklaşımına göre daha avantajlıdır. MPA yaklaşımı ise zaman parametresi açısından STA yaklaşımına göre daha iyi sonuçlar vermektedir. Teorik analizin işaret ettiği bu bulgular, deneysel sonuçlar tarafından da desteklenmektedir. Verinin geri çatılması ve düğüm tamiri işlemlerinin başlatılmasında kullanılan mesaj ve zaman karmaşıklıklarının ortalama durum özeti Çizelge 4.2'de de gösterilmektedir.

Ortalama durum için MPA yaklaşımı $O(\frac{k}{\delta}t + logN(t + d))$ zaman karmaşıklığına sahiptir. STA yaklaşımında ise bu karmaşıklık $O(\rho(k)(t + d))$ ile ifade

edilmektedir. Genel olarak küçük boyutlu paketler için t değeri, k ve $log(N)$ değerlerine göre çok düşüktür. Sonuç olarak MPA yaklaşımında $\frac{k}{\delta}t$ değeri büyük k değerleri için bile çok düşük olmaktadır. Diğer bir taraftan STA yaklaşımında ağacın yüksekliği MPA yaklaşımında oluşan ağacın yüksekliğinden daha büyük olmaktadır (STA ve MPA yaklaşımlarında ağaç ve en kısa yol hesabı için geçen süreler gözardı edilmiştir. Çünkü bu işlemler sadece verinin ilk isteminde, düğüm tamiri ya da geri çatım yüzünden, ya da ağacın (ya da en kısa yolun) içerisinde olan herhangi bir düğümün arızalanması sonucunda gerçekleştirilir.).

Çizelge 4.2. STA ve MPA yaklaşımlarının verinin geri çatılması ve düğüm tamirinin başlatılması işlemlerindeki ortalama karmaşıklık analizleri.

Yaklaşımlar	**Ortalama Durum Karmaşıklıkları**	
	Mesaj Karmaşıklığı	**Zaman Karmaşıklığı**
STA	$O(k^{0.8\frac{logN}{log\tilde{d}}})$	$O(\rho(k)(t+d))$
MPA	$O(k\frac{logN}{log\tilde{d}})$	$O(\frac{k}{\delta}t + logN(t+d))$

4.1.1.2 Veri güncellemesi işleminin iletişim maliyeti ile ilgili kuramsal bulgular

MDS silinti kodları kullanılarak yedeklenmiş verinin orijinal hali üzerinde bir güncelleme işlemi gerçekleştiğinde, kodlanmış veriyi saklayan her depolama düğümünün de kendi kodlanmış verisini güncellemesi gerekmektedir. Bu yüzden, bu bölümde k tane hedef düğüm yerine $n-k$ tane hedef düğüm bulunmaktadır. İlerleyen kısımlarda veri güncelleme işleminde STA ve MPA yaklaşımı kullanıldığında oluşan mesaj ve zaman karmaşıklarının analizi gerçekleştirilmiştir.

I. Steiner Ağaç Yaklaşımının (STA) Analizi Bölüm 3.1.2.2'de Steiner ağaç yaklaşımıyla veri güncelleme işleminin nasıl yapıldığı anlatılmıştı. Bu bölümde güncelleme işleminde kullanılan Steiner ağaç yaklaşımının mesaj ve zaman karmaşıklığı analizlerine yer verilmiştir.

Mesaj Karmaşıklığı: Bu iletişim yaklaşımında ortaya çıkan mesaj karmaşıklığı (Dolagh and Moazzami, 2011)'de verilen Steiner ağaç yaklaşım algoritması tarafından oluşturulan Steiner ağacın toplam kenar sayısı ve güncellenecek verinin miktarı ile doğru orantılıdır. Ayrıca, *MTU* limiti (veri hattı katmanının tek seferde taşıyabi-

leceği maksimum veri boyutu (Kurose and Ross, 2005)) bulunduğu için büyük boyutlu veriler tek seferde gönderilemez. Aşağıdaki analizlerde güncellenecek verinin miktarı Δ_{size}, *MTU* ise *MTU*'nun bayt cinsinden büyüklüğü olarak ifade edilmiştir.

Teorem 4.9. *Steiner ağaç yaklaşımının veri güncelleme işlemi için kullandığı ortalama mesaj sayısı* $O((n-k)^{0,8}\frac{log(N)}{log(\tilde{d})}\frac{\Delta_{size}}{MTU})$*'dır.*

İspat. *Güncelleme işlemi için kullanılan mesaj sayısı, Steiner ağacı üreten yaklaşım algoritması tarafından oluşturulan ağacın toplam kenar sayısının* $\frac{\Delta_{size}}{MTU}$ *katıdır. Ön-kuram 4.1'den çizgedeki herhangi iki düğüm arasındaki uzaklığın* $\frac{log(N)}{log(\tilde{d})}$ *ve Ön Kuram 4.2'den* $L_{m=(n-k)}$ *ve* L_u *arasındaki oranın* $(n-k)^{0,8}$ *olduğu bilgisi bulunmaktadır. Sonuç olarak, ağacın beklenen kenar sayısı* $(n-k)^{0,8}\frac{log(N)}{log(\tilde{d})}$ *olduğundan, güncelleme işlemi için kullanılan ortalama mesaj sayısının karmaşıklığı* $O((n-k)^{0,8}\frac{logN}{log\tilde{d}}\frac{\Delta_{size}}{MTU})$*'dır.*

Teorem 4.10. *Steiner ağaç yaklaşımının veri güncelleme işlemi için kullandığı mesaj sayısı en kötü durumda* $O(N\frac{\Delta_{size}}{MTU})$ *karmaşıklığına sahiptir.*

İspat. *Dolagh and Moazzami (2011) tarafından önerilen algoritma tarafından oluşturulan Steiner ağacının* $n-k$ *hedef düğüm ve* z *tane ara düğümden (* $0 \leq z \leq N-(n-k)$ *) oluştuğu farzedilsin. Veri güncelleme işleminde kullanılan toplam mesaj sayısı ağacın kenar sayısıyla doğru orantılıdır. Bu ağaçta* $n-k+z$ *tane kenar vardır ve* $(n-k+z)$*,* $O(N)$ *ile üstten sınırlıdır. Ayrıca, mesajın tamamının gönderimi için* $\frac{\Delta_{size}}{MTU}$ *tane çoğa-gönderim oturumu gerçekleştirilecektir. Sonuç olarak en kötü durumda toplam mesaj sayısı* $O(N\frac{\Delta_{size}}{MTU})$ *karmaşıklığına sahiptir.*

Aşağıda STA yaklaşımının zaman karmaşıklığının analizi incelenmiştir.

Zaman Karmaşıklığı: Veri güncelleme işlemi için bu yaklaşım kullanıldığında meydana gelen iletişimin zaman karmaşıklığı güncellenecek verinin miktarına ve kullanılan Steiner ağacının yüksekliğine bağlıdır.

Teorem 4.11. *Steiner ağaç yaklaşımının bir veri güncelleme işlemi için harcadığı süre ortalamada* $O(t\frac{\Delta_{size}}{MTU} + \rho(k)(t+d))$ *karmaşıklığına sahiptir.*

İspat. *Ön-kuram 4.3'te bir Steiner ağacın ortalama yüksekliği* $\rho(k)$ *olarak gösterilmiştir. Ortalama süre ise şu şekilde hesaplanır: tüm mesaj paketlerinin,* Δ*, kaynaktan çıkma süresi artı son mesajın en uzak düğüme ulaşması için geçen süredir. Tüm* Δ *değerinin kaynaktan çıkma süresi* $O(t\frac{\Delta_{size}}{MTU})$*'dir. Mesajın kaynak düğümden en uzak düğüme ulaşma süresi ise ortalamada* $\rho(k)$ *ile doğru orantılıdır. Yani toplamda ortalama geçen sürenin karmaşıklığı* $O(t\frac{\Delta_{size}}{MTU} + \rho(k)(t+d))$*'dır.*

Teorem 4.12. *Bir güncelleme işlemi için STA yaklaşımının harcadığı süre en kötü durumda* $O(t\frac{\Delta_{size}}{MTU} + N(t+d))$ *ile üstten sınırlıdır.*

İspat. *En kötü durum yine bir hedef düğümün kaynak düğümden en uzak konumda olduğu durumda oluşur. En uzak yolun kenar sayısı $N-1$ olduğu için, Steiner yaklaşımıyla gerçekleşen bir güncelleme işlemi iletişimi üstten $O(t\frac{\Delta_{size}}{MTU}+N(t+d))$ ile sınırlıdır.*

II. Çoklu En Kısa Yol Yaklaşımının (MPA) Analizi Çoklu en kısa yol yaklaşımının veri güncelleme işleminin iletişiminde gerektirdiği mesaj karmaşıklığı ve zaman karmaşıklığı analizleri aşağıda yapılmıştır.

Mesaj Karmaşıklığı: Burada mesaj karmaşıklığı MPA ile oluşturulan çoklu en kısa yol ağacındaki kenar sayısına bağlıdır.

Teorem 4.13. *Veri güncelleme işlemi için çoklu en kısa yol yaklaşımının ortalamada kullandığı mesaj sayısı $(n-k)\frac{\Delta_{size}}{MTU}\frac{log(N)}{log(\tilde{d})}$ ile üstten sınırlanmıştır.*

İspat. *Ön-kuram 4.1'den iki düğüm arasındaki ortalama kenar sayısının $\frac{log(N)}{log(\tilde{d})}$ olduğu bilinmektedir. N düğümlü bir çizgede, kaynak düğümün $n-k-1$ tane düğüme mesaj göndermesi gerekmektedir ve bu düğümün herhangi bir düğüme veri iletimi $\frac{\Delta_{size}}{MTU}$ peşpeşe gönderim gerektirir. Bu yüzden, ortalamada harcanan mesaj sayısı $O((n-k)\frac{\Delta_{size}}{MTU}\frac{log(N)}{log(\tilde{d})})$ karmaşıklığındadır.*

Teorem 4.14. *Veri güncelleme işlemi için çoklu en kısa yol yaklaşımının en kötü durumda kullandığı mesaj sayısı $O((n-k)\frac{\Delta_{size}}{MTU}N)$ ile sınırlanmıştır.*

İspat. *MPA yaklaşımında, Δ değeri MTU büyüklüğünde paketler halinde tüm hedef düğümlere ayrı ayrı gönderilmelidir. En kötü senaryoda hedeflerin hepsi kaynağa en uzak yerlerdedir ve kaynak düğümle hedef düğüm arasındaki yolların ilk kenarı hepsinde ortaktır. Böyle bir durumda kaynağa en uzak düğüm mesafesi $(N-1)$ tane kenar, kaynaktan hedefe $(n-k)$'ıncı en uzak yol ise $(N-n+k)$ tane kenar içermektedir. Sonuç olarak en kısa yol ağacı en fazla $(N-1)+(N-2)+\ldots+(N-n+k)$ tane kenar içermektedir. Gönderme işlemleri $\frac{\Delta_{size}}{MTU}$ defa tekrarlanacağı için toplamda kullanılan mesaj sayısı $O(\frac{\Delta_{size}}{MTU}N(n-k))$ ile üstten sınırlandırılmıştır.*

İlerleyen kısımda MPA yaklaşımının zaman karmaşıklığının analizi ele alınmıştır.

Zaman Karmaşıklığı: Güncelleme işleminde gerçekleştirilmesi gereken iletişimin zaman karmaşıklığı güncellenecek verinin miktarına, kaynak düğüm ve hedef düğümler arasındaki yolların ilk kenarlarının ortak olup olmamasına ve kaynak düğümle kaynak düğüme en uzak hedef düğüm arasındaki kenar sayısına bağlıdır.

Teorem 4.15. *Veri güncelleme işlemi için MPA yaklaşımının ortalama durumda kullandığı süre* $O(\frac{(n-k)}{\delta}\frac{\Delta_{size}}{MTU}t + log(N)(t+d))$ *ile sınırlandırılmıştır.*

İspat. *Kaynak düğümün ortalamada* δ *tane kenarı vardır ve kaynak düğümden çıkan her kenar ortalamada* $\frac{(n-k)}{\delta}$ *tane en kısa yolun ilk kenarını oluşturmaktadır. Ön-Kuram 4.2'den* N *düğümlü bir çizgenin ortalama çapının* $\theta(log(N))$ *olduğu bilinmektedir. Kaynak düğüm son mesajı göndermeden önce ortalamada* $O((\frac{(n-k)}{\delta} - 1)\frac{\Delta_{size}}{MTU}t)$ *kadar süre geçmektedir. Bu nedenle ortalamada geçen süre* $(\frac{(n-k)}{\delta}\frac{\Delta_{size}}{MTU}t + (logN)t + (logN)d)$ *olarak formüle edilebilir. Bu değer de* $O(\frac{(n-k)}{\delta}\frac{\Delta_{size}}{MTU}t + logN(t+d))$ *karmaşıklığına sahiptir.*

Teorem 4.16. *Veri güncelleme işlemi için MPA yaklaşımının en kötü durumda kullandığı süre* $O((n-k)\frac{\Delta_{size}}{MTU}t + N(t+d))$ *ile üstten sınırlandırılmıştır.*

İspat. *En kötü durum; kaynak düğümle hedef düğümler arasındaki en kısa yolların ilk çıkışlarının hepsinin aynı olması durumuyla birlikte paket gönderilen son hedef düğümünün kaynak düğüme en uzak noktada bulunması durumunun aynı anda gerçekleşmesiyle meydana gelir. Son mesaj gönderilmeden önce darboğaz kenar yüzünden* $(n-k-1)\frac{\Delta_{size}}{MTU}t$ *kadar süre geçer. Son mesajın en uzak düğüme gönderildiği varsayıldığında sadece en son mesajın iletim süresi* $O((N-1)(t+d))$ *olur. Böylece en kötü durumda geçen süre toplamda* $O((n-k)\frac{\Delta_{size}}{MTU}t + N(t+d))$ *ile sınırlandırılır.*

Çizelge 4.3'te veri güncellemesi işleminin iletişim süreci için hesaplanan ortalama karmaşıklık analizlerinin özeti verilmiştir. Yapılan analizlere göre STA yaklaşımı güncelleme işlemi için genel olarak daha avantajlıdır. Mesaj kullanımı açısından ise MPA yaklaşımına kıyasla STA yaklaşımı daha verimlidir. Ayrıca, iki yaklaşım için de zaman karmaşıklığının ikinci terimleri gözardı edilebilir, çünkü bu terim gerçek değerler aldığında oldukça küçük değerlere sahip olmaktadır. Dolayısıyla, STA yaklaşımı MPA yaklaşımına göre daha düşük zaman karmaşıklığına sahiptir.

Çizelge 4.3. STA ve MPA yaklaşımlarının verinin güncellenmesi işlemindeki ortalama karmaşıklık analizleri.

Yaklaşımlar	**Ortalama Durum Karmaşıklıkları**	
	Mesaj Karmaşıklığı	**Zaman Karmaşıklığı**
STA	$O((n-k)^{0,8}\frac{log(N)}{log(\tilde{d})}\frac{\Delta_{size}}{MTU})$	$O(t\frac{\Delta_{size}}{MTU} + \rho(k)(t+d))$
MPA	$O((n-k)\frac{\Delta_{size}}{MTU}\frac{log(N)}{log(\tilde{d})})$	$O(\frac{(n-k)}{\delta}\frac{\Delta_{size}}{MTU}t + log(N)(t+d))$

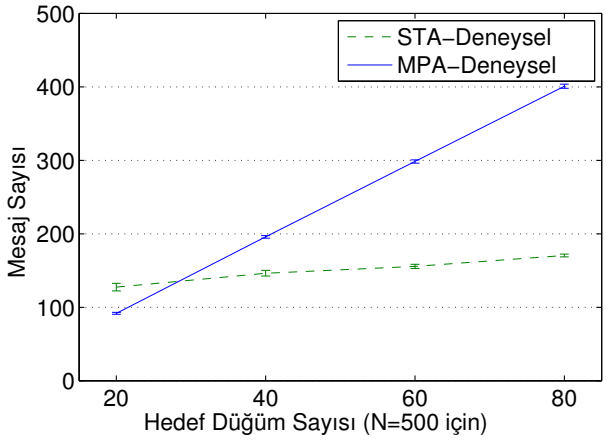
4.1.2 Deneysel bulgular

Verinin geri çatılmasının, düğüm tamirinin başlatılması ve veri güncellemesi işlemleri, sırasıyla Steiner ağaç yaklaşımı ve çoklu en kısa yol iletişim yaklaşımları kullanılarak ns-3 ağ benztim ortamında (ns-3, 2015) gerçekleştirilmiştir. Ağ topolojisini üretmek için BRITE aracı kullanılmıştır (Medina et al., 2000). Ağların üretiminde Waxman topoloji modeli (Waxman, 1988) kullanılmıştır. Benzetimlerde ağların toplam düğüm sayısı 500 olarak alınmıştır. Ağın bant genişliği değerleri 10Mbps-100 Mbps aralığından eş dağılımlı olarak olarak seçilmiştir. Ayrıca, düğümler arasındaki hatların yayılma gecikmesi 50 ms olarak alınmıştır. Ağdaki her bir hattın (kenarın) ağırlığı 1 birim ve iki düğüm arasındaki ağırlık ise bu düğümler arasındaki hat sayısı olarak alınmıştır. Testlerde MDS kodu olarak Reed-Solomon kodlama şeması (Reed and Solomon, 1960) kullanılmıştır. Benzetimlerde iki farklı performans parametresine bakılmıştır. Bu parametreler; verinin geri çatılmasının ve düğüm tamirinin başlatılması ve verinin güncellenmesi işlemlerinde harcanan mesaj sayısı ve bu işlemler için gerekli olan süredir. Mesaj sayısı bir yol üzerindeki her hattan geçen paket sayısıdır. Mesaj boyutu MTU'dan büyük ise bu mesaj Δ/MTU tane farklı paket olarak sayılmaktadır.

Dağıtık depolama sisteminin hangi platforma hizmet ettiği silinti kodlarında yer alan n ve k parametrelerinin değerini etkilemektedir. Örneğin depolama sistemi olarak da kullanılabilen özdeş (P2P) ağlarda (Gupta and Awasthi, 2011), düğüm arızalanmaları sistemden geçici ya da kalıcı olarak çıkma gibi nedenlerden dolayı oldukça çok yaşanmaktadır (Duminuco and Biersack, 2008). Böyle sistemlerdeki n ve k değerleri, depolama sistemine adanmış veri merkezlerinin yer aldığı platformların sahip olduğu n ve k değerlerine göre yüksek olmalıdır. İletişim modeli yaklaşımlarının bu farklı platformlarda vereceği sonuçları yansıtabilmek için benzetimlerde farklı n ve k değerleri kullanılmıştır.

4.1.2.1 Verinin geri çatılması ve düğüm tamiri işlemlerinin başlatımasının iletişim maliyeti ile ilgili deneysel bulgular

500 düğümlük (N) bir ağda, değişen hedef sayıları (k) için verinin geri çatılması ve düğüm tamiri işlemlerinin başlatılması işlemlerinde harcanan mesaj sayıları Şekil 4.1'de verilmiştir. Bu şekilde, hedef düğüm sayısı tüm düğümlerin %6'sını geçtiğinde STA yaklaşımının daha az mesaj harcadığı açıkça görülmektedir. Sabit boyutlu bir ağda hedef depolama düğümlerinin sayısı arttıkça, STA yaklaşımının avantajı da artmaktadır.

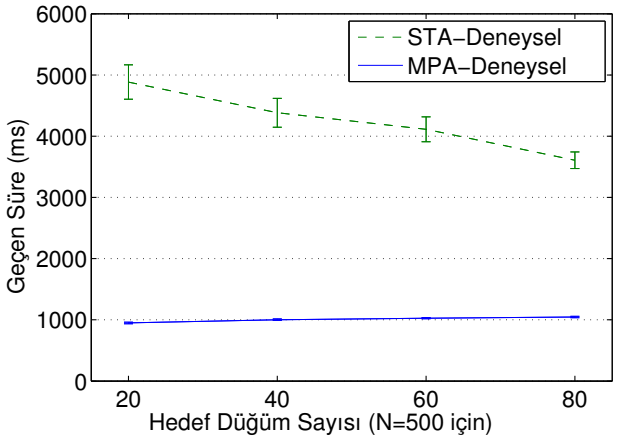


Şekil 4.1. 500 düğümlük bir ağda, değişik hedef düğümü sayısına göre verinin geri çatılmasının ve düğüm tamirinin başlatılması işlemlerinde kullanılan mesaj sayısı.

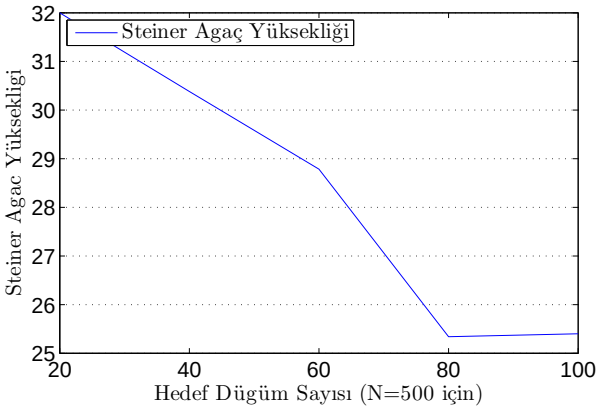
MPA yaklaşımı, 500 düğümlük bir ağda hedef düğüm sayısı 28'den az olduğunda mesaj sayısı parametresinde daha iyi bir performans sağlamaktadır. Ancak 28'den büyük tüm hedef düğüm sayıları için STA yaklaşımı mesaj sayısında daha iyi sonuçlar vermektedir. Hedef düğümü 28'den düşük olduğunda STA yaklaşımının kullandığı Steiner ağaçtaki toplam hat sayısı MPA yaklaşımı tarafından oluşturulan çoklu teke gönderim ağacının oluşturduğu kenar sayısından daha yüksektir. Yani, Şekil 4.1'de görülen mesaj sayısı sonuçları bu ağaçlardaki kenar sayısına bağlı olarak şekillenmektedir.

Mesaj parametresinde iyi sonuçlar veren STA yaklaşımı zaman parametresinde iyi sonuçlar vermemektedir. Şekil 4.2'de, 500 düğümlük (N) bir ağda, değişik hedef düğüm sayıları (k) için STA ve MPA yaklaşımlarında verinin geri çatımı ve düğüm tamiri işlemlerinin başlatılması işlemlerinde harcanan süreler görülmektedir. Hedef düğüm sayısı arttıkça MPA yaklaşımının kullandığı süre sabite yakın bir şekilde kalıyorken STA yaklaşımı bir azalma göstermekte ancak hala MPA yaklaşımına göre daha fazla süre harcamaktadır. Hedef düğüm sayısı arttıkça daha kısa yol bulma şansı arttığı için, STA yaklaşımında oluşturulan Steiner ağaçların yüksekliği hedef düğüm sayısı arttıça düşmekte ancak hedef düğüm sayısı belli bir değerin üstündeyken ağaç yüksekliği yaklaşık olarak sabitlenmektedir. STA'nın zaman parametresinde MPA'ya göre iyi olmamasının nedeni ise oluşan Steiner ağaçlarının yüksekliğinin çoklu teke gönderim ağacı yüksekliklerinden daha fazla olmasıdır. Sonuç olarak STA'da en uzaktaki düğüme ulaşılması daha uzun sürmektedir. Şekil 4.3'te 500 düğümlük bir ağda Steiner ağaç yüksekliğinin hedef sayısına göre değişimi görülmektedir. STA yaklaşımının zaman parametresindeki kötü performansının sebebi, oluşan ağacın yüksekliğinin çoklu en kısa yol yaklaşımında oluşan ağaçtan

ortalamada daha yüksek olmasıdır. Bu bulgu kuramsal sonuçlarda da görülmektedir.



Şekil 4.2. 500 düğümlük bir ağda, değişik hedef düğümü sayısına göre verinin geri çatılmasının ve düğüm tamirinin başlatımı işlemlerinde harcanan süre.



Şekil 4.3. 500 düğümlük bir ağda Steiner Ağaç yüksekliğinin hedef sayısına göre değişimi.

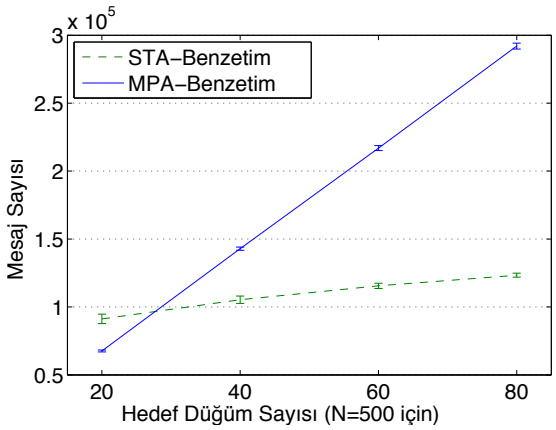
4.1.2.2 Güncelleme işleminin iletişim maliyeti ile ilgili deneysel bulgular

Benzetimlerde ilk olarak 1MB'lık güncelleme işlemleri test edilmiştir. Güncelleme işleminin gerçekleştirilmesi için kodlanmış veri depolayan her depolama düğümünün 1MB boyutunda veriyi, Δ, almış olması gerekmektedir.

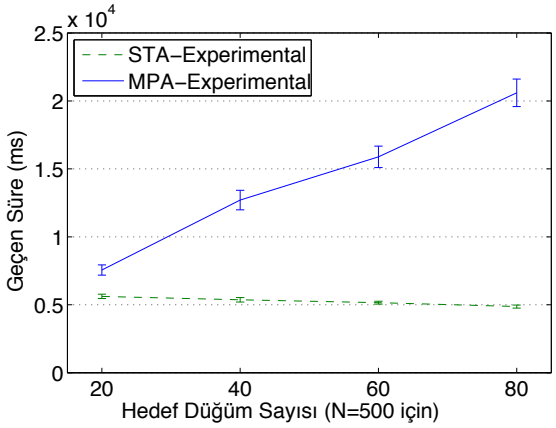
500 düğümlük bir ağda farklı hedef düğüm sayıları için her iki yaklaşımda kullanılan mesaj sayıları Şekil 4.4'te gösterilmiştir. Hedef düğüm sayısı belirli bir sayıyı aştığında Steiner ağaç yaklaşımı çoklu en kısa yol yaklaşımına göre daha az

mesaj kullanır. Sabit sayıda düğüme sahip bir ağda hedef düğüm sayısının artmasıyla STA'nın mesaj parametresindeki avantajı da artmaktadır.

500 düğüme sahip bir ağda veri güncellemesi işlemi için değişen hedef düğüm sayılarına göre harcanan süre Şekil 4.5'te gösterilmiştir. STA yaklaşımı her durumda MPA yaklaşımından daha az süre gerektirmektedir. STA yaklaşımında bir yol üzerindeki düğümlere aynı mesaj çoğa-gönderim ile iletilmekte, MPA yaklaşımında ise her hedef için ayrı bir mesaj hazırlanıp gönderilmektedir. Bu durum fazladan paket gönderim süresi ek yükü doğurmaktadır. Ayrıca, sabit düğüm sayısına sahip bir ağda hedef düğüm sayısı arttıkça STA yaklaşımının zaman parametresinde sağladığı avantaj artmaktadır.



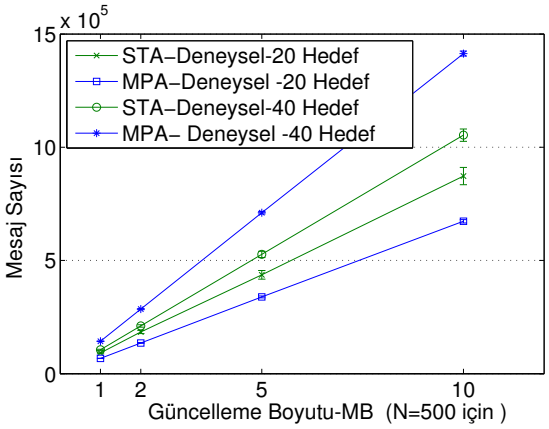
Şekil 4.4. STA ve MPA yaklaşımlarında 500 düğümlük bir ağda veri güncellemesi işleminde harcanan mesaj sayısı.



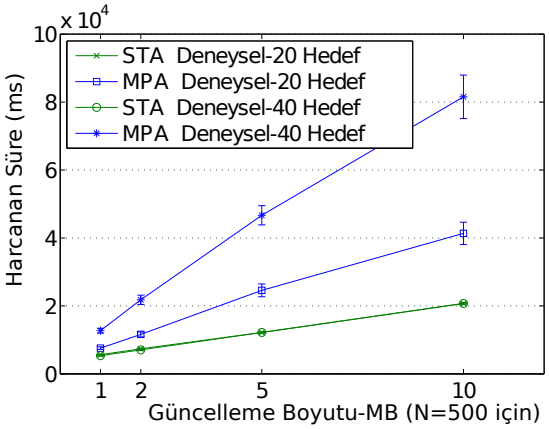
Şekil 4.5. STA ve MPA yaklaşımlarında 500 düğümlük bir ağda veri güncellemesi işleminde harcanan süre.

Farklı büyüklükteki güncellemeler için STA ve MPA yaklaşımlarında kulla-

nılan mesaj sayıları ve harcanan süreler Şekil 4.6 ve Şekil 4.7'de verilmiştir. Hedef düğüm sayısı 20 iken STA yaklaşımının kullandığı Steiner ağaçlarının kenar sayısı, MPA yaklaşımının kullandığı çoklu teke gönderim ağaçlarından daha yüksek olduğu için bu hedef düğüm sayısında farklı güncelleme boyutları için STA yaklaşımının kullandığı her mesaj sayısı MPA yaklaşımındakine göre daha fazla olacaktır. Kullanılan süreler ise her durumda STA yaklaşımında daha düşük çıkmıştır.



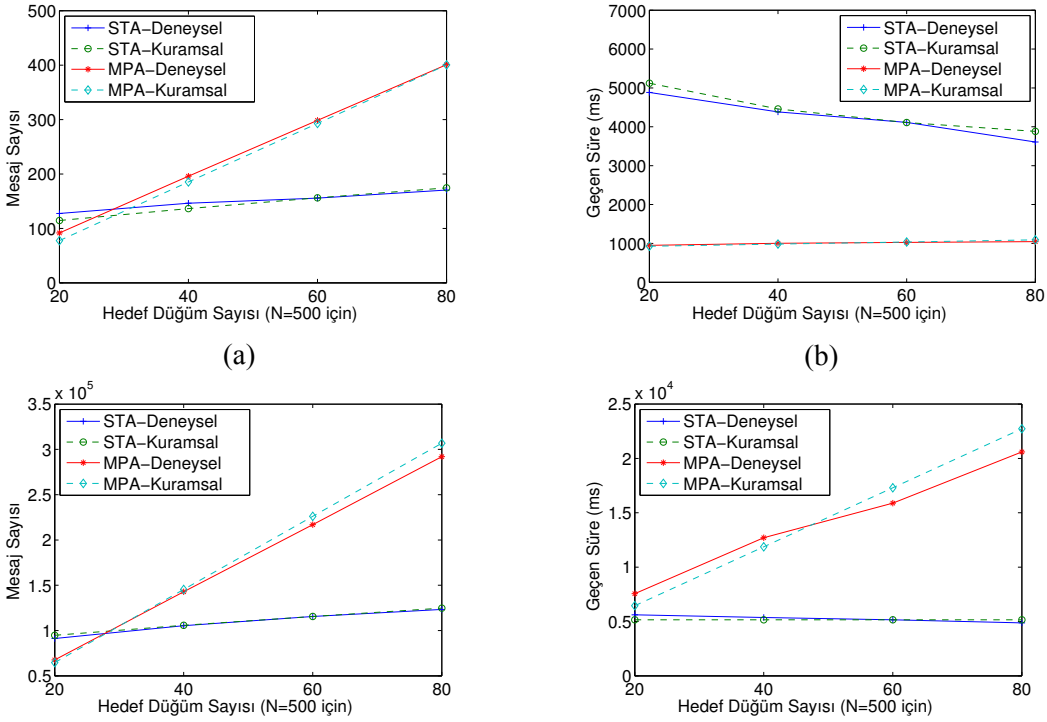
Şekil 4.6. STA ve MPA yaklaşımlarında 500 düğümlük bir ağda değişken miktarlarda veri güncellemesi işlemlerinde harcanan mesaj sayısı.



Şekil 4.7. STA ve MPA yaklaşımlarında 500 düğümlük bir ağda farklı miktarlarda veri güncellemesi işleminde 20 ve 40 hedef düğüm sayısı için harcanan süre.

Şekil 4.8 (a)'da verinin geri çatılması ve düğüm tamiri başlangıcı işlemlerinde kullanılan mesaj sayısı için elde edilmiş benzetim sonuçları ve uygun sabit ve katsayılar kullanılarak elde edilmiş kuramsal sonuçlar gösterilmektedir. Görüldüğü gibi, kuramsal sonuçlar ve benzetim sonuçları birbirleri ile uyuşmaktadır. Şekil 4.8 (b)'de verinin geri çatılması ve düğüm tamiri başlatımı işlemlerinde kullanılan zaman parametresi için elde edilmiş benzetim sonuçları ve uygun sabit ve katsayılar kullanılarak

elde edilmiş kuramsal sonuçlar gösterilmektedir. Ancak kuramsal sonuçlarda daha yakın bir sonuç elde edebilmek için Eşitlik (4.11) yerine Eşitlik (4.4) kullanılmıştır [1]. Bu kuramsal sonuçlar benzetim sonuçları ile birebir örtüşmese bile yine de uygun bir üst sınır oluşturmaktadır. Şekil 4.8 (c)'de veri güncellemesi işleminde kullanılan mesaj sayısı için elde edilmiş benzetim sonuçları ve uygun sabit ve katsayılar kullanılarak elde edilmiş kuramsal sonuçlar gösterilmektedir. Bu benzetim sonuçları kuramsal sonuçlara büyük ölçüde uymaktadır. Şekil 4.8 (d)'de veri güncellemesi işleminin zaman parametresi için elde edilen benzetim sonuçları ve kuramsal sonuçlar görülmektedir. Burada da kuramsal ve deneysel sonuçlar birbiriyle örtüşmektedir.



Şekil 4.8. STA ve MPA yaklaşımlarının verinin geri çatılmasının/düğüm tamirinin başlatılması ve veri güncelleme işlemlerinde kuramsal ve benzetim sonuçlarının karşılaştırılması a) Verinin geri çatılmasının/düğüm tamirinin başlatılması için kullanılan mesaj sayısı, b) Verinin geri çatılmasının/düğüm tamirinin başlatılması için kullanılan süre, c) Veri güncelleme işlemi için kullanılan mesaj sayısı, d) Veri güncelleme işlemi için kullanılan süre.

[1] Eşitlik (4.4)'te $f(P(i+1))$ fonksiyonunun $(i+1)$'inci döngüdeki düğüm çifti sayısının doğrusal bir fonksiyonu olduğu varsayılmıştır.

4.2 HMBR Kodlama Şeması İle İlgili Elde Edilen Bulgular

Bu alt bölümde e-MBR (Rashmi et al., 2011b), HSRC (Oggier and Datta, 2011b) ve HMBR (Haytaoglu and Dalkilic, 2013a) kodlama şemalarında düğüm tamiri ve verinin geri çatılması işlemlerinin karmaşıklık analizleri gerçekleştirilmiştir. Ayrıca, e-MBR, HSRC ve HMBR kodlama şemaları ile kodlanmış bir verinin geri çatılmasının başarım olasılıkları hesaplanarak karşılaştırılmıştır. Son olarak, e-MBR, HSRC ve HMBR kodlama şemaları ile düğüm tamirinin başarılı bir şekilde gerçekleştirilebilme olasılıkları incelenmiştir.

4.2.1 HMBR kodlarının hesapsal karmaşıklık analizi

e-MBR, HSRC ve HMBR kodlama şemalarının kullanabileceği en küçük boyutlu sonlu cisimler sırasıyla şunlardır: $2^{\lceil log_2 n \rceil}$, $2^{((\frac{k+1}{2}+(d-k))\lceil log_2 n \rceil)}$, $\sigma = max(2^d, (2^{(\lceil log_2 n \rceil + 1 - (\lceil log_2 n \rceil - \lfloor log_2 n \rfloor))}))$.

e-MBR kodlarında verinin geri çatılması işlemi $k \times k$'lık Φ_{VT} matrisinin tersini alma işlemini gerektirir, bu da $O(k^3)$ karmaşıklığına sahiptir. T matrisinin elde edilmesi işlemi ise $k \times k$'lık ve $k \times (d-k)$'lık iki matrisin çarpılmasını gerektirmektedir. Bu işlem $O(k^2(d-k))$ mertebesinde toplama ve çarpma karmaşıklığı gerektirmektedir. S matrisinin elde edilmesi işlemi ise $k \times (d-k)$'lık ve $(d-k) \times k$'lık iki matrisin çarpılması işlemini gerektirmekte ve bu işlem de $O(k^2(d-k))$'lık bir karmaşıklığa sahiptir. Ardından iki tane $k \times k$'lık matris üzerinde toplama işlemine ihtiyaç duyulmaktadır; bu da $O(k^2)$ seviyesinde bir karmaşıklık yaratır. S matrisinin elde edilme işleminin son adımında bir matris tersi alma işlemi ve matris çarpma işlemi bulunmaktadır ki ikisi de $O(k^3)$'lük karmaşıklığa sahiptir. Burada yapılan her işlem en az n boyutunda bir sonlu cisim üzerinden gerçekleştirilmektedir.

e-MBR kodlarında tamir etme işlemi iki tane vektör çarpımı işlemi gerektirir ve $O(d)$ mertebesinde bir karmaşıklığa sahiptir. Ardından yeni-gelen düğüm $d \times d$'lik bir matrisin tersini hesaplar ($O(d^3)$). Son olarak, tamir etme işlemi $d \times d$ ve $d \times 1$'lik iki matrisin çarpımı ile sona ermektedir, bu işlem de $O(d^2)$'dir. Burada da yapılan her işlem en az n boyutunda bir sonlu cisim üzerinden gerçekleştirilmektedir.

HSRC kodlarında verinin geri çatılması işlemi $O(k^3)$'lük bir karmaşıklığa sahiptir. Çünkü bu kodlama şeması $k \times k$'lık bir matrisin tersini alır ve $k \times k$'lık ve $k \times 1$'lik bir matris arasında bir çarpma işlemi gerçekleştirir. Bu işlemlerin hepsi minimumda $\mathbb{F}_{2^{((\frac{k+1}{2}+(d-k))\lceil log_2^n \rceil)}}$'lık bir sonlu cisim üzerinden gerçekleştirilmekte-

dir.

Bozulmuş bir düğümü tamir etmek için, HSRC kodlama şeması $O(\tau)$ tane toplama işlemi gerçekleştirir. τ en fazla k olabilir yani $O(\tau) = O(k)$'dır.

$AMBR_{DR}$ yöntemi ile verinin geri çatılmasında HMBR kodlama şeması, e-MBR kodlama şemasıyla aynı adımlara sahiptir. Tek farkı gerekli olan asgari sonlu cisim boyutudur. HMBR kodlama şemasında veri-toplayıcı düğüm, $(k \times k)$'lık bir matrisin tersini alır, bu işlem $O(k^3)$'lük toplama ve çarpma işlemi gerektirir. Ardından $k \times k$'lık ve $k \times (d-k)$'lık iki matrisi çarpar ($O(k^2(d-k))$). Benzer şekilde $k \times (d-k)$'lık ve $(d-k) \times k$ boyutlarında iki matrisi çarpma işlemi gerçekleştirir, bu işlemin karmaşıklığı $O((d-k)k^2)$'dir. Bu çarpma işleminin sonucu iki matris arasında $O(k \times k)$'lık bir toplama işlemi için kullanılır. Son olarak, bir matris tersini alma işlemi gerçekleştirilir bu da $O(k^3)$'lük çarpma ve toplama işlemi gerektirir. Tüm işlemler en az $\mathbb{F}_{max(2^d,(2^{\lceil log_2^n \rceil + 1 - (\lceil log_2^n \rceil - \lfloor log_2^n \rfloor)}))}$'lık bir sonlu cisim üzerinden gerçekleştirilir.

$AMSR_{DR}$ yöntemi ile verinin geri çatılması işleminde veri-toplayıcı düğüm, ilk $(d-k)$ adımda kullanılacak olan Φ_{VT}^{-1} matrisini hesaplamak için bir matris tersi alma işlemi gerçekleştirir. Bu matrisin tersini alma işlemi $O(k^3)$'lük bir karmaşıklığa sahiptir. M matrisinin son $d-k$ sütununu geri çatmak için $(k \times k)$'lık ve $(k \times 1)$'lik bir matris çarpma işlemi gerçekleştirilir bu da toplamda $O((d-k)k^2)$'lik bir toplama ve çarpma işlem yükü yaratmaktadır. Ardından, $p_j(x)$ polinomları kullanılarak $p'_j(x)$ sonuçlarının oluşturulması ($j \leq k$ için) $O(dk^2)$ seviyesinde toplama ve çarpma işlem karmaşıklığı yaratmaktadır. M matrisinin ilk k sütununun geri çatılması işlemi $O(k^4)$ seviyesinde toplama ve çarpma işlem yükü oluşturmaktadır. Yani, sonuç olarak $ASRC_{DR}$ yöntemi ile verinin geri çatılması işleminin $O(k^4 + dk^2)$ mertebesinde bir hesapsal karmaşıklığı bulunmaktadır.

$\mathbb{F}_{2^m}$ sonlu cismi üzerinde iki tane sembolü toplamak $O(m)$, çarpmak ise $O(m^2)$ tane $\mathbb{F}_2$ işlemi gerektirmektedir (Menezes et al., 1996). Ancak, günümüzde sonlu cisimlerin aritmetik işlemleri önceden kayıtlanmış tablolar üzerinden gerçekleştirildiği için bu işlemler ile ilgili bu karmaşıklıklar göz ardı edilmiştir. Sonlu cisimlerin boyutları bu tabloların kapladığı bellek alanlarını etkilemektedir.

Verinin geri çatılması işleminde $AMBR_{DR}$ yönteminin hesapsal karmaşıklığı e-MBR kodlama şemasındaki verinin geri çatılması yöntemi ile aynıdır. $AMBR_{DR}$ yönteminin hesapsal karmaşıklığı HSRC kodlama şemasınınki ile aynı ya da ondan daha yüksek olabilir. Bu koşulları sağlayabilecek $[n, k, d]$ değerleri kolaylıkla bu-

lunabilir. Ek olarak, $ASRC_{DR}$ yönteminin hesapsal karmaşıklığı e-MBR kodlama şemasında verinin geri çatılması yönteminin hesapsal karmaşıklığından daha yüksektir. Ayrıca $ASRC_{DR}$ yönteminin hesapsal karmaşıklığı HSRC kodlarından daha yüksektir.

HMBR kodlama şemasında düğüm tamir etme işlemleri iki farklı şekilde gerçekleştirilebildiği için, bu iki yol ayrı ayrı incelenmiştir. $AMBR_{NR}$ yönteminde yeni-gelen düğüm, d tane yardımcı düğüme bağlanır. Yardımcı düğümler $O(d)$ karmaşıklığına sahip vektör-vektör çarpma işlemi gerçekleştirir. Yeni-gelen düğüm E_{yrd} matrisinin tersini alma işlemini gerçekleştirir, bu işlem $O(d^3)$ çarpma ve toplama işlemi demektir. Son olarak $AMBR_{NR}$ yolu ile tamir etme işleminde $O(d^2)$ karmaşıklığına sahip bir matris çarpımı işlemi gerçekleştirilir.

$ASRC_{NR}$ yönteminde ise yeni-gelen düğüm τ düğüme bağlanır ve her birinden d tane sembol indirerek hepsini toplama işleminden geçirir. Bu işlem $O(\tau d)$'lik bir karmaşıklık getirir. İki tamir etme yoluyla tüm işlemler minimumda $\mathbb{F}_{max(2^d,(2^{\lceil log_2^n \rceil}+1-(\lceil log_2^n \rceil - \lfloor log_2^n \rfloor)))}$'lık bir sonlu cisim üzerinden gerçekleştirilir. Verinin geri çatılması ve düğüm tamiri işlemlerinin karmaşıklık analizlerinin özeti Çizelge 4.4 ve Çizelge 4.5'de bulunmaktadır.

HMBR kodlarının $ASRC_{NR}$ düğüm tamiri yöntemi, e-MBR kodlarındaki düğüm tamiri yöntemine göre her durumda daha düşük hesapsal karmaşıklığına sahiptir. $AMBR_{NR}$ yöntemi ise e-MBR kodlarındaki düğüm tamiri ile aynı hesapsal karmaşıklığa sahip olabilir. Ayrıca, $ASRC_{NR}$ yönteminin hesapsal karmaşıklığı HSRC kodlarındaki düğüm tamirinin hesapsal karmaşıklığından daha yüksektir. HSRC kodlama şemasındaki düğüm tamiri işlemi her durumda $AMBR_{NR}$ yönteminden daha düşük hesapsal karmaşıklığa sahiptir.

Çizelge 4.4. Verinin geri çatılması işleminin hesapsal karmaşıklık analizi özeti

Kodlama Şeması	**İşlem**	**Verinin Geri Çatılması**	
e-MBR	Çarpma	$O(k^3) + O(k^2(d-k))$	
	Toplama	$O(k^3) + O(k^2(d-k))$	
HSRC	Çarpma	$O(k^3)$	
	Toplama	$O(k^3)$	
		$AMBR_{DR}$	$ASRC_{DR}$
HMBR	Çarpma	$O(k^3) + O(k^2(d-k))$	$O((k^4 + dk^2))$
	Toplama	$O(k^3) + O(k^2(d-k))$	$O((k^4 + dk^2))$

Çizelge 4.5. Düğüm tamiri işlemlerinin hesapsal karmaşıklık analizi özeti

Kodlama Şeması	**İşlem**	**Düğüm Tamiri**	
e-MBR	Çarpma	$O(d^3)$	
	Toplama	$O(d^3)$	
HSRC	Çarpma	0	
	Toplama	$O(\tau)$	
		$AMBR_{NR}$	$ASRC_{NR}$
HMBR	Çarpma	$O(d^3)$	0
	Toplama	$O(d^3)$	$O(d\tau)$

4.2.2 HMBR kodlarında verinin geri çatılması olasılığı

Bu bölümde sırasıyla e-MBR, HSRC ve HMBR kodlama şemalarında verinin geri çatılmasının başarım olasılığı incelenmiştir. Tüm kodlama şemalarında $\mathbb{F}_{n+1}$ sonlu cisminde; $n = 2^z - 1$, $\sigma = n+1$ ve $z \geq d$ için $B = \frac{k(k+1)}{2} + k(d-k)$ adet sembol kodlanmaktadır. Analizlerde düğümlerin çalışır durumda olma olasılığını p_{saglam} temsil etmektedir ve bu olasılık her düğüm için eşit olarak kabul edilmiştir.

e-MBR kodlarında k tane çalışır durumdaki düğüm, verinin geri çatılması işlemini garanti etmektedir. Çünkü, Ψ matrisinin herhangi k satırı doğrusal bağımsızdır. Rassal değişken x, Oggier and Datta (2011b) tarafından da gösterildiği gibi çalışır durumdaki düğüm sayısını temsil etmektedir; bu değişken, n ve p_{saglam} parametreleri ile birlikte binom dağılıma sahiptir. Böylece e-MBR kodlarında verinin geri çatılması olasılığı, $p_{dr_{e-MBR}}$, aşağıda verilmiştir:

$$p_{dr_{e-MBR}} = \sum_{x=k}^{n} \binom{n}{x} {p_{saglam}}^{x} (1 - p_{saglam})^{(n-x)} \tag{4.12}$$

HSRC kodlarında verinin geri çatılabilmesi için, $\mathbb{F}_2$ sonlu cismi üzerinden doğrusal bağımsız polinom sonuçlarına sahip olan en az k tane düğüme ihtiyaç duyulmaktadır. Oggier and Datta (2011b) r kertesine sahip $(x \times z)$ boyutlu ikili (*binary*) matrislerin tüm permütasyonlarını saymak için aşağıda verilen $R(x, z, r)$ fonksiyonunu tanımlamışlardır:

$$R(x,z,r) = \begin{cases} 0, r = 0 \ \vee \ r > x \ \vee \ r = x \, , x > z \vee r < x \, , r > z \\ \prod_{i=0}^{r-1}(2^z - 2^i), \quad r = x \ , \ x \leqslant z \\ R(x-1,z,r-1)(2^z - 2^{r-1}) \\ \ + R(x-1,z,r)(2^r - x), \quad r < x, r \leqslant z \end{cases} \tag{4.13}$$

Eşitlik (4.13)'te r, matrisin kertesini; x çalışır durumda olan düğüm sayısını ve z ise bir polinom sonucundaki bit sayısını temsil etmektedir. Bu fonksiyon; x tane çalışır durumda düğüm olduğu varsayıldığında, bu x düğümün polinom sonuçları ikili formda olmak üzere üst üste konulduğunda oluşabilecek matrislerden r kertesine sahip matris permütasyonlarını saymaktadır. Sistemde x tane çalışır durumda düğüm olduğunu farzedelim, bu x düğümün polinom çıktılarından aşağıdaki şekilde bir $(x \times z)$ boyutunda P matrisi oluşturulmuş olsun:

$$P = \left[p(w_{[x_1]}) \quad p(w_{[x_2]}) \quad \ldots \quad p(w_{[x_x]}) \right]^t \tag{4.14}$$

Burada $p(w_{[x_i]})$'ler x_i düğümünün içinde depolanan polinom sonucunu temsil etmektedir ($i \in \{1, 2, \ldots, x\}$). $R(x,z,r)$ fonksiyonu r kertesine sahip tüm P matrislerinin sayısını hesaplamaktadır. $R(x,z,r)$ fonksiyonunun detaylı açıklaması (Oggier and Datta, 2011b) çalışmasında verilmiştir. Sonuç olarak, P matrisinin r kertesine sahip olma olasılığı aşağıda hesaplanmıştır:

$$\rho(x,z,r) = \frac{R(x,z,r)}{\sum_{j=0}^{z} R(x,z,j)} \tag{4.15}$$

Böylece $[n, \ k]$ HSRC kodlama şemasında verinin geri çatılması işleminin gerçekleştirilebilme olasılığı aşağıdaki şekilde verilmiştir (Oggier and Datta, 2011b):

$$p_{dr_{HSRC}} = \sum_{x=k}^{n} \sum_{r=k}^{z} \rho(x,z,r) \binom{n}{x} {p_{saglam}}^x (p_{saglam})^{n-x} \tag{4.16}$$

HMBR kodlarında iki farklı veri geri çatılması yöntemi olduğu için, bu yöntemlerin başarım olasılıkları aşağıda ayrı ayrı analiz edilmiştir. $AMBR_{DR}$ yönteminde, veri-toplayıcı düğüm Φ satırları doğrusal bağımsız olan k tane düğüme ihtiyaç duymaktadır. HMBR kodlarında hem E hem de Φ matrisi, Eşitlik (4.17)'de verilen Moore matrisi yapısını (Dickson, 1901) sağlamaktadır.

$$\begin{bmatrix} \lambda_1 & \lambda_1^2 & \lambda_1^{2^2} & \dots & \lambda_1^{2^{m-1}} \\ \lambda_2 & \lambda_2^2 & \lambda_2^{2^2} & \dots & \lambda_2^{2^{m-1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \lambda_m & \lambda_m^2 & \lambda_m^{2^2} & \dots & \lambda_m^{2^{m-1}} \end{bmatrix}, \tag{4.17}$$

$\lambda_i \in \mathbb{F}_{p^s}, i \in \{1, 2, \dots, m\}$, $s > 1, p \in \mathbb{P}$ için.

Moore, eğer, $\lambda_1, \lambda_2, \dots, \lambda_m$ değerleri $\mathbb{F}_p$ üzerinden doğrusal bağımsız ise Eşitlik (4.17)'deki matrisin satırlarının da doğrusal bağımsız olduğunu göstermiştir (Dickson, 1901). Bu yüzden, ilk koşul sağlanırsa, Moore matrisinin (Φ_{VT}'nin) satırları $\mathbb{F}_{2^s}$'e göre de doğrusal bağımsızdır ($s > 1$ için).

Teorem 4.17. *$AMBR_{DR}$ yönteminde, polinom girdileri $\mathbb{F}_2$'ye göre doğrusal bağımsız olan k tane çalışır durumda düğüm varsa, verinin geri çatılması işlemi gerçekleştirilebilir.*

İspat. *Veri-toplayıcı düğümün k tane düğüme bağlandığı varsayılsın ve bu düğümler $\{h_1, h_2, \dots, h_k\}$ ile ifade edilsin. Veri-toplayıcı düğüm, bağlandığı düğümlerin Φ matrisi satırlarını kullanarak Φ_{VT} matrisini oluşturur. Φ_{VT} matrisinin belirteci (determinant) sıfırdan farklı ise veri geri çatılabilmektedir. Φ_{VT}'nin belirteci Eşitlik (4.17)'ye göre $w_{[h_i]}$'ler $\mathbb{F}_2$'ye göre doğrusal bağımlı olduğu zaman sıfıra eşit olmaktadır. Böylece, veri-toplayıcı düğüm, polinom girdileri $\mathbb{F}_2$'ye göre doğrusal bağımsız olan k tane düğüm bulabilirse, verinin geri çatılması işlemini garantilemektedir.*

$AMBR_{DR}$ yönteminde verinin geri çatılması başarım olasılığı HSRC kodlama şemasındakine benzer bir şekilde hesaplanabilmektedir. Toplam düğüm sayısı n olduğunda, Φ_{VT} matrisinin satırları doğrusal bağımsız olan k tane düğümün permütasyonlarını aşağıda anlatıldığı gibi sayılsın. Veri-toplayıcı düğüm ilk düğüm için $n = 2^z - 1$ düğümden herhangi bir tanesini seçebilmektedir. İkinci düğüm için ise geriye kalan $2^z - 2$ düğümden herhangi bir tanesini seçebilir. Üçüncü seçimde geriye

$2^z - 4$ tane düğüm kalmıştır, çünkü kalan $2^z - 3$ düğümden bir tanesi diğer iki satırın $\mathbb{F}_2$'ye göre bir doğrusal kombinasyonudur. Bir sonraki seçim için düğüm sayısı, önceki adımlarda seçilmiş olan düğümlerin satırlarının bir doğrusal kombinasyonuna sahip düğümler dahil edilmeyecek şekilde sayılmaktadır (Eşitlik (4.13)'teki ikinci durum).

Çalışır durumda olan x düğümün kodlama matrisi satırlarının ilk k sütunu kullanılarak bir matris oluşturulduğunu düşünelim. Bu matrise Φ_{alive_x} matrisi denilsin. Eğer bu Φ_{alive_x} matrisinin kertesi k veya k'dan yüksek ise tam kerteye sahip bir Φ_{VT} matrisi oluşturulabilir. Eğer çalışır durumda olan düğüm sayısı x, r kertesinden yüksek ise r kertesine sahip olan olası tüm Φ_{alive_x} matrislerinin sayısı, Eşitlik (4.13)'teki üçüncü durumdan hesaplanabilir. Φ_{alive_x} matrisinin $r \leq x$ kertesine sahip tüm permütasyonları aşağıdaki gibi iki farklı şekilde oluşturulabilmektedir.

- $r-1$ kertesine sahip bir $\Phi_{alive_{x-1}}$'e sahip olan $x-1$ tane çalışır durumda düğüm olduğunu varsayalım. Bu $x-1$ düğüme bir tane düğüm ekleyerek r kertesine sahip olan Φ_{alive_x} matrisi oluşturmak için, bu yeni eklenen düğümün Φ satırının, önceki düğümlerin Φ satırlarına doğrusal bağımsız olması gerekmektedir. Bu koşulu sağlayabilecek $2^z - 2^{r-1}$ düğümden herhangi bir tanesi bu yeni eklenecek düğüm olabilir.
- İkinci alternatif olarak yine $x-1$ tane düğüm olduğu varsayılmaktadır. Ancak bu sefer, bu düğümlerin $\Phi_{alive_{x-1}}$ matrisinin kertesi r'dir. Bu düğüm grubuna yeni bir düğüm eklendiği takdirde Φ_{alive_x}'in kertesinin hala r olarak kalması için, bu yeni eklenecek düğümün Φ satırınının önceki düğümlerin Φ satırlarına doğrusal bağımlı olması gerekmektedir. Bu koşulu sağlayabilecek $2^r - x$ tane farklı düğüm vardır.

Sonuç olarak, $AMBR_{DR}$'da, doğrusal bağımsız Φ satırlarına (ya da $\mathbb{F}_2$'ye göre bağımsız k tane $w_{[h_i]}$'ye) sahip en az k tane çalışır durumda olan düğüm olması gerekmektedir. HSRC kodlama şemasına benzer olarak, $AMBR_{DR}$ yöntemi ile verinin geri çatılmasının olasılığı Eşitlik (4.16) kullanılarak hesaplanabilmektedir. Böylece, $[n, k, d]$ HMBR kodlarının $AMBR_{DR}$ yöntemi ile verinin geri çatılması olasılığı aşağıdaki şekilde hesaplanabilmektedir:

$$p_{dr_{HMBR}} = \sum_{x=k}^{n} \sum_{r=k}^{z} \rho(x, z, r) \binom{n}{x} {p_{saglam}}^{x} (1 - p_{saglam})^{n-x} \tag{4.18}$$

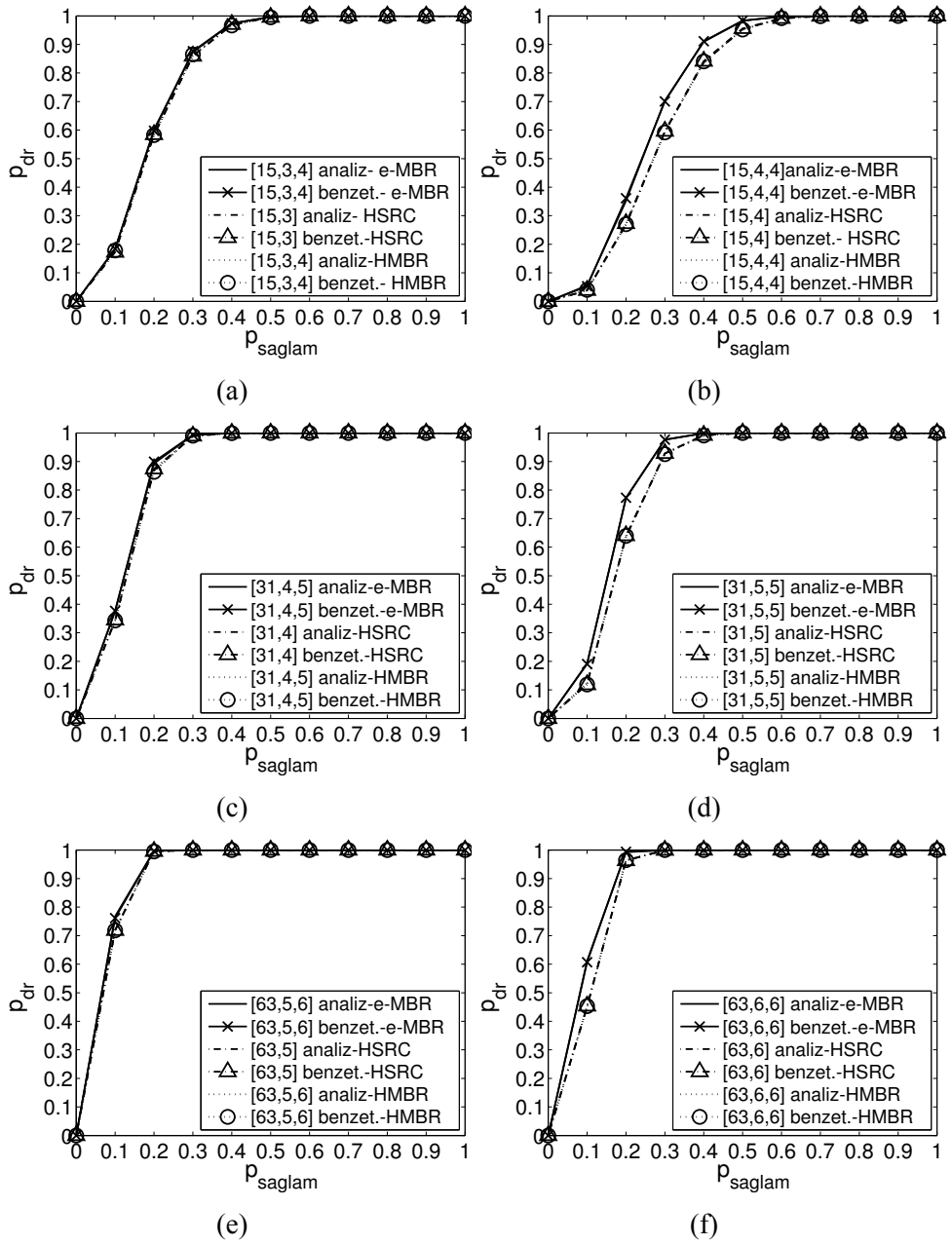
HSRC'ye benzer şekilde doğrusal bağımsız polinom girdisine ya da çıktısına sahip k düğümün bulunması $ASRC_{DR}$ yöntemi ile verinin geri çatılması işlemi için yeterlidir. $[n,\ k,\ d]$ HMBR kodları için $ASRC_{DR}$ yöntemi ile verinin geri çatılması olasılığı, $AMBR_{DR}$ ile geri çatılması olasılığına eşittir. Böylece, bu koşullar altında $ASRC_{DR}$'nin başarım olasılığı da Eşitlik (4.18) ile verilir.

Yukarıdaki olasılık analizlerinden görüldüğü gibi, $[n = 2^z - 1, k \geq 2]$ HSRC ve $[n = 2^z - 1, k \geq 2, d]$ HMBR kodlarında verinin geri çatılması olasılıkları, bu kodlama şemalarında kullanılan sonlu cisimlerin boyutları aynı ise eşittir.

Ayrıca, $ASRC_{DR}$ ve $AMBR_{DR}$ yöntemleri veriyi aynı koşullar altında (doğrusal bağımsız polinom girdi ya da çıktısına sahip k düğümün varlığı) geri çatabilmektedir. Yani $ASRC_{DR}$ ve $AMBR_{DR}$ yöntemlerinin başarım olasılıkları eştir:

$$p_{dr_{HMBR}} = p_{ASRC_{DR}} = p_{AMBR_{DR}} \tag{4.19}$$

Şekil 4.9'da HMBR, HSRC ve e-MBR kodlarında verinin geri çatılması başarım olasılıklarının kuramsal ve benzetim sonuçları görülmektedir. Buna göre, $[n,\ k]$ HSRC ve $[n,\ k,\ d]$ HMBR kodları aynı verinin geri çatılması başarım olasılığına sahiptir, $[n,\ k,\ d]$ e-MBR kodlarının başarım olasılığı ise bu iki koda göre biraz daha yüksektir. Ek olarak, k, d'den daha küçük olduğunda, bu üç kodlama şemasının veriyi geri çatması olasılığı neredeyse aynı olmaktadır. Çünkü, bu durumda HMBR ve HSRC için k tane doğrusal bağımsız Φ satırına sahip düğüm bulma olasılığı artmaktadır.



Şekil 4.9. $[n, k, d]$ e-MBR, $[n, k]$ HSRC ve $[n, k, d]$ HMBR kodlarında farklı $n,\ k,\ d$ değerleri için verinin geri çatılması olasılıkları a) $n = 15,\ k = 3,\ d = 4$ parametreleri için, b) $n = 15,\ k = 4,\ d = 4$ parametreleri için, c) $n = 31,\ k = 4,\ d = 5$ parametreleri için, d) $n = 31,\ k = 5,\ d = 5$ parametreleri için, e) $n = 63,\ k = 5,\ d = 6$ parametreleri için, f) $n = 63,\ k = 6,\ d = 6$ parametreleri için.

4.2.3 HMBR kodlarında düğüm tamiri olasılığı

Bu alt bölümde $[n,\ k,\ d]$ e-MBR, $[n,\ k]$ HSRC ve $[n,\ k,\ d]$ HMBR kodlarında düğüm tamiri başarım olasılığı $n = 2^z - 1$, $z \geq d$ ve $\sigma = 2^z$ boyutunda sonlu cisimler için analiz edilmiştir. Verinin geri çatılması işleminde olduğu gibi

burada da p_{saglam} düğümlerin çalışır durumda olma olasılığını göstermektedir ve bu olasılık her düğüm için aynıdır. Tüm kodlama şemaları $\mathbb{F}_{2^z}$ yani $\mathbb{F}_{n+1}$ sonlu cismini kullanmaktadır. Ek olarak, değişik $[n,\ k,\ d]$ ve $[n,\ k]$ kombinasyonlarında, bu üç kodlama şemasındaki düğüm tamiri başarım olasılıkları benzetim ile de ölçülmüş olup benzetim sonuçları kuramsal sonuçlarla karşılaştırılmıştır.

$[n,\ k,\ d]$ e-MBR kodları için çalışır durumdaki herhangi d düğüm, arızalanmış bir düğümü tamir etmek için yeterlidir (Rashmi et al., 2011b). $[n,\ k,\ d]$ e-MBR kodlarında düğüm tamiri olasılığı aşağıda verilmiştir:

$$p_{nr_{e-MBR}} = \sum_{i=d}^{n-1} \binom{n-1}{i} {p_{saglam}}^{i} (1 - p_{saglam})^{(n-i-1)} \tag{4.20}$$

$[n, k]$ HSRC kodlarında ise arızalanmış bir düğüm aşağıdaki koşulu sağlayan τ tane yardımcı düğüm ile tamir edilmektedir.

$$w_{[f]} = \sum_{i=1}^{\tau} w_{[h_i]} \tag{4.21}$$

Burada $w_{[f]}$, arızalanmış düğümün (düğüm f) polinom girdisi olmaktadır ve $w_{[h_i]}$ ise yardımcı düğüm i'nin polinom girdisidir. Düğüm f'in tamir edilmesi olasılığı τ'nun değerine göre değişim göstermektedir. Analizlerde, $\tau = 2$ olarak kabul edilmiştir, çünkü maksimum yerellik (en az yardımcı düğüm sayısı ile tamir) bu noktada sağlanmaktadır.

$n = 2^z - 1$ olduğu durumda, sonlu cisim boyutu $n+1$'dir ve her düğüm i, düğüm f'in tamir edilmesi için bir eşlenik düğüme sahiptir. Çünkü n düğüm arasından sadece bir düğümün polinom girdisi $(w[f] - w[h_i])$'ye eşittir.

İçinde düğüm f'i tamir edebilecek en az iki düğümün olduğu sağlam x düğümün permütasyonlarının, tüm x düğümlük permütasyonların sayısına oranı oranı aşağıda $\omega(n, x)$ ile gösterilmiştir:

$$\omega(n, x) = \frac{(P(n-1, x) - \sum_{i=1}^{x}(n - 1 - 2(i-1)))}{P(n-1, x)}, \quad x \leq n - 1 \tag{4.22}$$

Eşitlik (4.22)'de, ilk önce, içinde düğüm f'i tamir edebilecek herhangi iki düğüm bulundurmayan x tane düğümün tüm permütasyonlarının sayısı hesaplanmıştır. $n - 1$ tane düğümden herhangi bir tanesi ilk düğüm olarak seçilebilmektedir. İkinci düğüm olarak ise $n - 3$ tane düğümden herhangi biri seçilebilir, çünkü

$w[f]-w[h_1]$ girdisine sahip olan düğüm sayılmamalıdır. Üçüncü düğüm olarak ise $n-5$ tane düğümden biri seçilebilmektedir, çünkü geriye $n-3$ tane düğüm kalmıştır ancak bunlardan iki tanesi seçilen düğümlerin f düğümü için eşleniğidir. Bu şekilde devam ederek, geriye kalan $x-3$ seçimde, düğümler önceki seçilen düğümlerin eşlenikleri olmayacak şekilde sayılmaktadır. Diğer bir deyişle, i'inci seçimde, önceden seçilen $i-1$ düğümün eşlenikleri elenerek geriye kalan düğümler sayılmaktadır ($i \in \{1,2,...,x\}$). Bu işlem bittikten sonra, elde edilen permütasyon sayısı x düğümlük tüm permütasyonların sayısından çıkartılır ve sonuç, x düğümlük tüm permütasyonların sayısına bölünür.

Böylece $[n,\ k]$ HSRC kodlarında arızalanmış bir düğümün iki tane yardımcı düğüm $(\tau = 2)$ ile tamir edilebilme olasılığı aşağıdaki şekilde hesaplanabilmektedir:

$$p_{nr_{HSRC_{\tau=2}}} = \sum_{x=2}^{n-1} \omega(n,x) \binom{n-1}{x} {p_{saglam}}^{x} (1-p_{saglam})^{n-x-1} \tag{4.23}$$

Bu tez kapsamında geliştirilen HMBR kodları iki farklı düğüm tamiri yöntemi içermektedir: $AMBR_{NR}$ ve $ASRC_{NR}$. Bu iki yöntemin düğüm tamiri başarım olasılıkları farklıdır.

$AMBR_{NR}$, kodlama matrisinin satırları doğrusal bağımsız olan en az d tane çalışır durumda düğüme ihtiyaç duymaktadır. Yani, bu olasılık HMBR kodlarında verinin geri çatılması olasılığına benzemektedir. Dolayısı ile Eşitlik (4.18)'in bir modifikasyonu olarak, arızalanmış bir düğümün $AMBR_{NR}$ yöntemi ile tamir edilme olasılığı aşağıda verilmiştir:

$$p_{AMBR_{NR}} = \sum_{x=d}^{n-1} \sum_{r=d}^{z} \rho(x,z,r) \binom{n-1}{x} {p_{saglam}}^{x} (1-p_{saglam})^{n-x-1} \tag{4.24}$$

Diğer bir yandan $\tau = 2$ için, $p_{ASRC_{NR}}$; Eşitlik (4.23)'teki $p_{nr_{HSRC_{\tau=2}}}$ olasılığı ile aynı şekilde hesaplanabilmektedir:

$$p_{ASRC_{NR}} = \sum_{x=2}^{n-1} \omega(n,x) \binom{n-1}{x} {p_{saglam}}^{x} (1-p_{saglam})^{n-x-1} \tag{4.25}$$

Yukarıda $p_{AMBR_{NR}}$ ve $p_{ASRC_{NR}}$ olasılıkları gösterilmiştir. Ancak, HMBR kodlarında düğüm tamiri için $AMBR_{NR}$ veya $ASRC_{NR}$ yöntemlerinin ikisi de kullanılabildiğinden HMBR kodlarında düğüm tamiri başarım olasılığını hesaplamak için bu iki yöntemle düğüm tamirinin başarım olasılıklarının birleşimi hesaplanmalıdır.

$$\begin{aligned} p_{nr_{HMBR}} &= p_{AMBR_{NR}} \cup p_{ASRC_{NR}} \\ &= p_{AMBR_{NR}} + p_{ASRC_{NR} \cap AMBR'_{NR}} \end{aligned} \tag{4.26}$$

$$p_{nr_{HMBR}} = p_{ASRC_{NR}} + p_{AMBR_{NR} \cap ASRC_{NR}'} \tag{4.27}$$

$p_{ASRC_{NR}}$ ve $p_{AMBR_{NR}}$ olasılıklarının birleşimini hesaplamak için, arızalanmış bir düğümün tamiri için $ASRC_{NR}$ yöntemi kullanılamadığı durumlarda $AMBR_{NR}$ yönteminin başarım olasığı ve $AMBR_{NR}$ yönteminin gerçekleştirilemediği durumda bu düğümün $ASRC_{NR}$ yöntemi ile tamir edilebilmesinin olasığı aşağıdaki iki alt bölümde hesaplanmıştır.

4.2.4 $\tau = 2$ için $\mathbf{ASRC_{NR}}$ yöntemi uygulanamadığı durumlarda $\mathbf{AMBR_{NR}}$ yöntemi ile düğüm tamiri olasılığı

$\tau = 2$ için $ASRC_{NR}$ yönteminin uygulanamadığı durumlarda $AMBR_{NR}$ düğüm tamiri yönteminin başarım olasılığını bulmak için, Eşitlik (4.24) ile verilen $AMBR_{NR}$ yöntemi ile düğüm tamiri olasılığı formülünde $\rho(x,\ z,\ r)$ ifadesi değişikliğe uğramaktadır. Değişikliğe uğrayan bu ifade aşağıda verilen oldukça uzun bir analiz sonucu Eşitlik (4.35)'de verilmektedir.

Kodlama matrisi satırlarının kertesi r olan x tane çalışır durumda olan düğüm gruplarının tüm permütasyonlarının sayısı hesaplanmıştır. Düğüm f arızalandığı için bu düğüm bu permütasyonlara eklenememektedir. Bu permütasyonlar ikiye bölünmektedir: $w_{[f]}$'ye doğrusal bağımsız ($\mathbb{F}_2$'ye göre) polinom girdilerine sahip olan düğümleri içeren permütasyonlar ya da $w_{[f]}$'ye doğrusal bağımlı ($\mathbb{F}_2$'ye göre) polinom girdilerine sahip olan düğümleri barındıran permütasyonlar. Daha açık bir şekide, bu düğümlerin polinom girdileri kullanılarak $(x \times z)$ boyutunda ikili formda bir matris oluşturulmuş olsun. Bu matris polinom girdi matrisi olarak adlandırılmıştır. Eğer bu matrisin satırları $w_{[f]}$'ye doğrusal bağımsız ise, polinom girdileri $w_{[f]}$'ye doğrusal bağımsız denilmektedir. Ancak eğer, bu matrisin satırları $w_{[f]}$'ye doğrusal bağımlı ise, polinom girdileri $w_{[f]}$'ye doğrusal bağımlı denilmektedir.

Eğer çalışır durumda olan x düğümün polinom matrisinin kertesi x'e eşit ise, ilk gruptaki düğümlerin tüm permütasyonlarının sayısını hesaplamak için f düğümünün halihazırda bu gruptaki düğümlerden biri olduğu varsayılmaktadır. Yani, bu

gruptaki düğümlerin polinom girdileri zaten $w_{[f]}$'ye doğrusal bağımsız olacaktır. Bir başka ifade ile, bu permütasyonun içerisindeki düğümler düğüm f'i $ASRC_{NR}$ yöntemi ile tamir edemeyecektir. Polinom girdi matrisinin kertesi $r = x$ olan ve bu matrisinin satırları $w_{[f]}$'ye doğrusal bağımsız olan x tane düğümün tüm permütasyonlarının sayısı aşağıdaki fonksiyon ile hesaplanabilmektedir:

$$\gamma(x, z, r) = \prod_{i=1}^{x}(2^z - 2^i),\ r \leq z,\ x = r,\ r > 0 \tag{4.28}$$

Burada z bir polinom girdisinin temsil edildiği bit sayısıdır.

Çalışır durumda olan düğüm sayısı x, r kertesinden büyük olduğu zaman, polinom girdi matrisinin kertesi r olan ve bu matrisinin satırları $w_{[f]}$'ye doğrusal bağımsız olan x düğümlük tüm permütasyonların sayısının hesaplanması için Eşitlik (4.29) kullanılmaktadır. Bu durumda, bu permütasyonlar iki farklı şekilde oluşturulabilmektedir. İlk yöntemde, polinom girdi matrisinin kertesinin $r - 1$ olduğu ve bu matrisin satırlarının $w_{[f]}$'ye doğrusal bağımsız olan $x - 1$ düğüm olduğu farzedilmektedir. Burada polinom girdisi bu $x - 1$ düğümünkine doğrusal bağımsız olan ve yine doğrusal kombinasyonlar yoluyla önceki düğümlerle birlikte $w_{[f]}$'yi oluşturamayacak bir düğüm gerekmektedir. Bu koşulu sağlayan $2^z - 2^r$ tane düğüm bulunmaktadır. İkinci yöntemde ise polinom matrisinin kertesi r olan ve yine polinom girdileri $w_{[f]}$'ye doğrusal bağımsız olan $x - 1$ tane düğüm vardır. Bu kez, polinom girdisi önceki $x - 1$ düğümün girdilerine doğrusal bağımlı olan bir düğüm gerekmektedir. Bu düğümün girdisi doğal olarak öncekilerle birlikte $w_{[f]}$'yi oluşturamayacaktır. Kısacası, $x > r$ olduğu durumda, polinom girdi matrisinin kertesi r olan ve bu matrisin satırlarının vektör uzayının içinde $w_{[f]}$ bulunmayan tüm x düğümlük permütasyonların sayısı Eşitlik (4.29) ile hesaplanabilmektedir.

$$\begin{aligned}\gamma(x, z, r) = \gamma(x - 1, z, r - 1)(2^z - 2^r) + \gamma(x - 1, z, r)(2^r - x),\\ r \leq z, x > r,\ r > 0\end{aligned} \tag{4.29}$$

Diğer durumlar için $\gamma(x, z, r)$ fonksiyonu aşağıdaki gibi sıfırdır.

$$\gamma(x, z, r) = 0,\ r \leq 0 \text{ veya } r > x \text{ veya } r > z \tag{4.30}$$

$x = r$ olduğu durumda polinom girdi matrisinin satırları $w_{[f]}$'ye doğrusal bağımlı ve bu matrisinin kertesi r olan x düğümlük permütasyon grubunu saymak için aşağıdaki fonksiyon tanımlanmıştır:

$$\varphi(x,z,r) = \sum_{j=2}^{r}((\prod_{i=1}^{j-1}(2^z - 2^i))(2^{j-1}-1)(\prod_{i=j+1}^{r}(2^z - 2^{i-1}))), \quad x = r,\ r \leqslant z,\ r > 0 \tag{4.31}$$

Permütasyondaki düğümlerin girdileri ile $w_{[f]}$ arasındaki doğrusal bağımlılık ilişkisi, ikinci düğümün seçiminden r'inci düğümün seçimine kadar oluşabilmektedir. $\varphi(x,z,r)$ fonksiyonunda j bu doğrusal bağımlılık ilişkisinin başladığı seçimin indisini temsil etmektedir. Bu ilişki oluşmadan önce, polinom girdileri birbirlerine ve $w_{[f]}$'ye doğrusal bağımsız olan düğümler eklenmektedir. Bu düğümlerin olası tüm permütasyonlarının sayısı Eşitlik (4.31)'deki $\prod_{i=1}^{j-1}(2^z - 2^i)$ kısmında hesaplanmaktadır. $2^{j-1}-1$ tane farklı düğüm önceki adımlarda seçilmiş $j-1$ tane düğümün girdisi ile ile $w_{[f]}$ arasında bir doğrusal bağımlılık ilişkisi yaratabilmektedir. Bu doğrusal bağımlılık ilişkisi oluştuktan sonra eklenebilecek düğümler ($i \geq j+1$) sadece $i \mid i < j+1$'inci adımda seçilen düğümlerin girdilerine doğrusal bağımsız girdilere sahip olan düğümlerdir.

Bu noktadan sonra, polinom girdi matrisinin kertesi r olan ve bu matrisin satırları $w_{[f]}$'ye doğrusal bağımlı olan $x > r$ olmak üzere x tane düğümün tüm permütasyonlarının sayısının hesaplanması gösterilecektir. Bu durumda bu permütasyonlar; kerte, $w_{[f]}$ ile olan doğrusal bağımlılık ilişkisi gibi farklı özelliklere sahip $x-1$ düğümlük üç farklı permütasyon tipi tarafından oluşturulabilmektedir.

İlk permütasyon tipi, polinom girdi matrisinin kertesi $r-1$ olan ve bu matrisin satırları $w_{[f]}$'ye doğrusal bağımsız olan $x-1$ tane düğümden oluşmaktadır. Yani bu permütasyona, $x-1$ düğümün polinom girdi matrisinin satırlarına doğrusal bağımsız olan yeni bir düğüm eklenmelidir. Yanlız bu yeni düğüm önceki düğümlerin polinom girdileriyle $w_{[f]}$ arasında doğrusal bağımlılık ilişkisi yaratmalıdır. Bunu sağlayabilen $2^{r-1}-1$ tane düğüm vardır.

İkinci permütasyon tipi, polinom matrisinin kertesi $r-1$ olan ve polinom girdileri $w_{[f]}$'ye doğrusal bağımlı olan $x-1$ tane düğümden oluşmaktadır. İstenilen x düğümlük permütasyonu elde edebilmek için, önceki düğümlerin girdilerine doğrusal bağımsız bir girdiye sahip bir düğüm eklenmelidir. Sonuç olarak bu şekilde

$2^z - 2^{r-1}$ tane değişik düğüm bulunmaktadır.

Üçüncü permütasyon tipi polinom girdi matrisinin kertesi r olan ve bu girdileri $w_{[f]}$'ye doğrusal bağımlı olan $x-1$ tane düğümden oluşmaktadır. Yani, önceki $x-1$ düğümün polinom girdisine doğrusal bağımlı bir girdiye sahip bir düğüm eklenebilir (düğüm f hariç). Bu koşulu sağlayan $2^r - 2 - (x-1)$ tane farklı düğüm bulunmaktadır. Eğer $2^r - 1 < x$ ise, istenilen üçüncü permütasyon tipi oluşturulamaz. Bu durumda, Eşitlik (4.32)'deki ikinci durum kullanılmaktadır, aksi halde ilk durum kullanılmaktadır.

$$\varphi(x,z,r) = \begin{cases} \gamma(x-1,z,r-1)(2^{r-1}-1) \\ +\varphi(x-1,z,r-1)(2^z - 2^{r-1}) \\ +\varphi(x-1,z,r)(2^r - 2 - (x-1)), \\ \qquad\qquad x > r, r \leqslant z,\ r > 0,\ 2^r - 1 \geq x. \\ \gamma(x-1,z,r-1)(2^{r-1}-1) \\ +\varphi(x-1,z,r-1)(2^z - 2^{r-1}), \\ \qquad\qquad x > r,\ r \leqslant z,\ r > 0,\ 2^r - 1 < x. \\ 0, \qquad\qquad r > x \text{ or } z < r \text{ or } r \leq 0. \end{cases} \tag{4.32}$$

$p_{AMBR_{NR} \cap ASRC'_{NR}}$'nü hesaplamak için, polinom girdi matrisinin kertesi r olan ve f düğümünü $ASRC_{NR}$ yolu ile tamir edebilecek iki tane düğüm içermeyen x düğümün permütasyonları sayılmaktadır. Bu amaçla, φ fonksiyonu üzerinde birtakım değişiklikler yapılmıştır. $\gamma(x,z,r)$ fonksiyonu $w_{[f]}$'ye doğrusal bağımsız girdiler bulunduran düğümlerin permütasyonlarını saymaktadır. Bu yüzden, bu permütasyonlardaki herhangi iki düğüm, f düğümünü $ASRC_{NR}$ yolu ile tamir edememektedir. Ancak $\varphi(x,z,r)$ fonksiyonunda f düğümünü $ASRC_{NR}$ yolu ile tamir edebilicek iki tane düğümün bulunduğu permütasyonlar sayılabilmektedir. Bu yüzden $\varphi(x,z,r)$ fonksiyonu bu şekildeki ikilileri saymayacak şekilde yeniden düzenlenmiştir. $\varphi(x,z,r)$ fonksiyonun bu düzenlenmiş hali $\tilde{\varphi}(x,z,r)$ olarak adlandırılmıştır.

$x = r$ için $\varphi(x,z,r)$ fonksiyonundaki, $w_{[f]}$ ve önceki $j-1$ düğümün girdisi arasındaki doğrusal bağımlılığın oluştuğu j'inci seçim yeniden düzenlenmelidir. Bu

düzenleme j'inci seçimde önceden seçilmiş $j-1$ düğümün eşleniklerini saymayarak yapılabilmektedir. Bu düzenleme, Eşitlik (4.33)'teki $\tilde{\varphi}(x,z,r)$ fonksiyonun ikinci çarpanı olan $2^{j-1}-j$'de gösterilmektedir. $2^{j-1}-1$ tane farklı düğüm $w_{[f]}$ ve önceki $j-1$ tane farklı düğümün polinom girdisi arasında doğrusal bağımlılık ilişkisi sağlayabilmekte idi. Ancak bu düğümlerin $j-1$ tanesi, halihazırda seçilmiş $j-1$ düğümün f düğümünü tamir ederken eşleniğidir. Seçilmiş düğümlerin polinom girdileri ve bu düğümlerin $ASRC_{NR}$ yolu ile f düğümünü tamir etmek için eşleştiği düğümlerin polinom girdileri şöyledir: $\{(w_{[n_1]}, w_{[n_1]}+w_{[f]}), (w_{[n_2]}, w_{[n_2]}+w_{[f]}), \ldots, (w_{[n_{j-1}]}, w_{[n_{j-1}]}+w_{[f]})\}$, burada $w_{[n_l]}$ seçilmiş olan düğümün polinom girdisidir ve $l \in \{1,2,\ldots,j-1\}$. Böylece, $2^{j-1}-j$ düğümden herhangi bir tanesi, kendi girdisi ve önceki seçilen $j-1$ düğümün arasından herhangi iki veya daha çok düğümün girdisi arasındaki doğrusal bağımlılık ilişkisini sağlayabilmektedir. Bu ilk doğrusal bağımlılık ilişkisi oluştuktan sonra (j'inci seçimden sonra), geriye kalan seçimlerdeki düğümlerin polinom girdileri doğal olarak $w_{[f]}$ ve önceki düğümlerin polinom girdilerine doğrusal bağımsız olacaktır. Sonuç olarak, j'inci düğüm seçiminden sonra geriye kalan seçimlerdeki düğümlerin polinom girdileri $w_{[f]}$ ve halihazırda seçilmiş düğümlerin polinom girdilerinin vektör uzayında olamaz. Bu da demek oluyor ki, f düğümünü $ASRC_{NR}$ ile tamir edebilecek herhangi iki düğüm bu permütasyonların içerisinde yer alamaz. Yani, $i > j$ için i'inci düğüm seçiminde, $2^z - 2^{i-1}$ farklı düğüm seçeneği bulunmaktadır. $x = r$ ve $z \geq r > 0$ için $\tilde{\varphi}(x,z,r)$ fonksiyonu aşağıda verilmiştir:

$$\tilde{\varphi}(x,z,r) = \sum_{j=3}^{r}((\prod_{i=1}^{j-1}(2^z-2^i))(2^{j-1}-j)\prod_{i=j+1}^{r}(2^z-2^{i-1})), \quad x=r,\ z\geq r,\ r>0 \tag{4.33}$$

$x > r$ olduğu durumlarda, polinom girdi matrisinin kertesi r olan, ayrıca bu matrisin satırları $w_{[f]}$'ye doğrusal bağımlı olan ve f düğümünü $ASRC_{NR}$ yöntemi ile tamir edebilecek herhangi iki düğüm içermeyen x düğümün permütasyon sayısı aşağıdaki $\tilde{\varphi}(x,z,r)$ fonksiyonu ile hesaplanabilmektedir.

$$\tilde{\varphi}(x,z,r) = \begin{cases} \gamma(x-1,z,r-1)(2^{r-1}-(x-1)-1) \\ +\tilde{\varphi}(x-1,z,r-1)(2^z-2^{r-1}) \\ +\tilde{\varphi}(x-1,z,r)(2^r-2x), \\ \qquad\qquad x>r,\ z\geq r,\ 2^r\geq 2x. \\ \tilde{\varphi}(x-1,z,r-1)(2^z-2^{r-1}), \\ \qquad\qquad x>r,\ z\geq r,\ 2^r<2x. \\ 0, \qquad\qquad r>x \text{ or } z<r \text{ or } r\leq 0. \end{cases} \tag{4.34}$$

Eşitlik 4.34'te yer alan bu x düğümlük permütasyonlar, $x-1$ düğümlük üç farklı permütasyon tipi ile oluşturulabilmektedir.

İlk permütasyon tipi polinom matrisinin kertesi $r-1$ olan ve bu matrisin satırları $w_{[f]}$'ye doğrusal bağımsız olan $x-1$ tane düğümden oluşmaktadır. Yani, yeni eklenecek olan düğümün polinom girdisi permütasyondaki $x-1$ düğümün polinom girdisine doğrusal bağımsız ise istenilen permütasyon elde edilebilir. Ancak bu yeni düğümün polinom girdisinin seçilmiş $x-1$ düğümün polinom girdilerinin birden fazlası ile $w_{[f]}$ arasında doğrusal bağımlılık ilişkisi kurması gerekmektedir. Bu durumu sağlayan $2^{r-1}-x$ tane farklı düğüm bulunmaktadır ($2^{r-1}\geq x$ durumu için).

İkinci permütasyon tipi polinom matrisinin kertesi $r-1$ olan ve bu matrisinin satırları (üç veya daha fazlası) $w_{[f]}$'ye doğrusal bağımlı olan $x-1$ tane düğümden oluşmaktadır. Polinom girdi matrisinin satırlarına doğrusal bağımsız bir düğüm eklenmesi gerekmektedir. Doğal olarak, bu düğüm ikinci tip permütasyondaki $x-1$ düğümün herhangi birisi ile f düğümünü $ASRC_{NR}$ yolu ile tamir edemeyecektir. Sonuç olarak bu koşulu sağlayabilecek 2^z-2^{r-1} tane farklı düğüm vardır.

Üçüncü permütasyon tipi polinom girdi matrisinin kertesi r olan ve bu matrisinin satırları (üç veya daha fazlası) $w_{[f]}$'ye doğrusal bağımlı olan $x-1$ tane düğümden oluşmaktadır. Yani bu polinom matrisinin satırlarına doğrusal bağımlı bir polinom girdisi olan ancak seçilmiş $x-1$ düğümün polinom girdisinin eşleniği olmayan bir polinom girdisine sahip olan bir düğüm eklenebilir. $2^r\geq 2x$ iken bu koşulu sağlayan 2^r-2x tane düğüm vardır, çünkü seçilmiş düğümlerin polinom girdilerine doğrusal bağımlı girdilere sahip olan 2^r-x-1 tane düğüm vardır ve

bunların $x-1$ tanesi üçüncü tip permütasyondaki $x-1$ tane düğümün polinom girdilerinin eşleniğidir.

Şimdiye kadar, polinom girdi matrisinin kertesi r olan ve içerisindeki herhangi iki düğüm ile f düğümünü $ASRC_{NR}$ yöntemi kullanarak tamir edemeyecek x düğümün olası tüm permütasyonlarının sayısı hesaplandı. Böylece, arızalanmış bir düğümün iki yardımcı düğüm kullanılarak $ASRC_{NR}$ yöntemi ile tamir edilmesi olanaksız iken $AMBR_{NR}$ yöntemi ile tamir edilebilmesi olasılığı aşağıda formülize edilmiştir.

$$p_{AMBR_{NR}\cap ASRC'_{NR}} = \sum_{x=d}^{n-1}\sum_{r=d}^{z}\binom{n-1}{x}\frac{(\gamma(x,z,r)+\tilde{\varphi}(x,z,r))}{P(n-1,x)} \times {p_{saglam}}^{x}(1-p_{saglam})^{n-x-1}, \quad z \geq d \tag{4.35}$$

Şekil 4.10 (a), Şekil 4.10 (b) ve Şekil 4.10 (c)'de, farklı HMBR kodları için, $p_{AMBR_{NR}\cap ASRC'_{NR}}$ olasılıklarının kuramsal ve benzetim sonuçları görülmektedir. Arızalanmış bir düğümün, $ASRC_{NR}$ ($\tau = 2$) yöntemi ile tamir edilemediği durumda $AMBR_{NR}$ yolu ile tamir edilebilme olasılığı kayda değer büyüklüktedir. Bu olasılık p_{saglam}, $[0.2, 0.3]$ aralığında iken maksimum değerini alır. p_{saglam} yükseldikçe $p_{ASRC_{NR}}$ de yükseldiği için, $p_{AMBR_{NR}\cap ASRC'_{NR}}$ gittikçe düşmektedir, hatta sıfır değerini alabilmektedir. Sonuç olarak, p_{saglam}'ın düşük değerleri için $[n,\ k,\ d]$ HMBR kodlarında arızalanan bir düğüm, $ASRC_{NR}$ ($\tau = 2$) yöntemi ile tamir edilemiyorsa, $AMBR_{NR}$ yöntemi ile tamir edilebilmektedir.

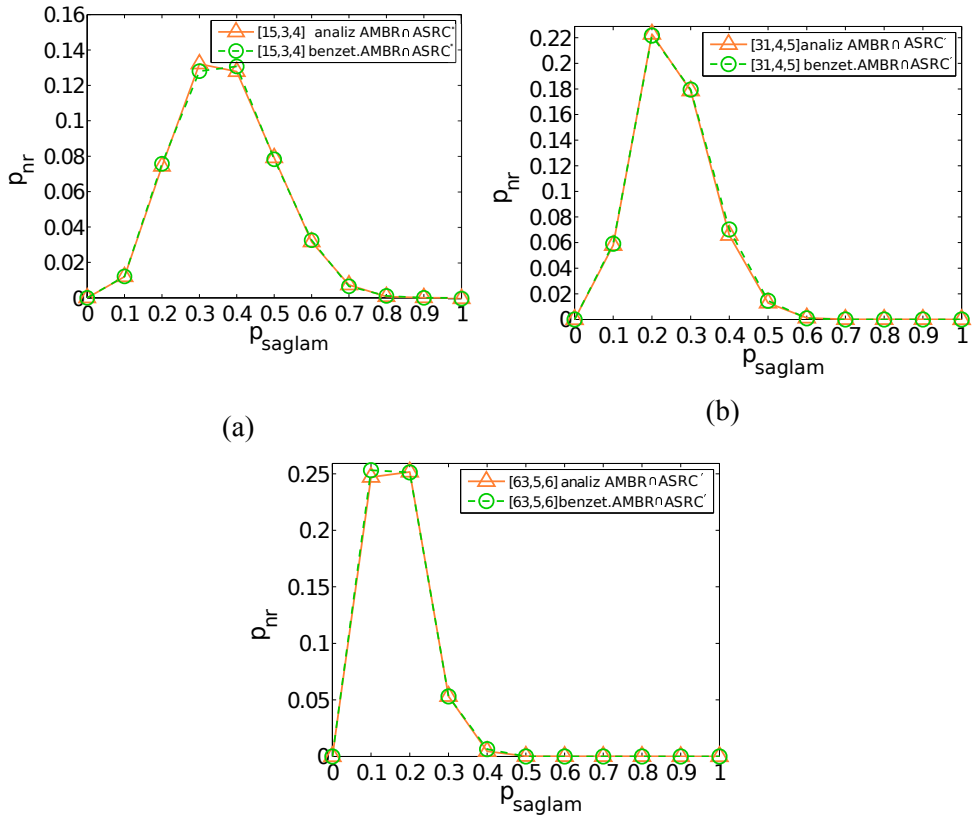
4.2.5 $\mathbf{AMBR_{NR}}$ yöntemi uygulanamadığı durumlarda $\boldsymbol{\tau = 2}$ için $\mathbf{ASRC_{NR}}$ yöntemi ile düğüm tamiri olasılığı

Bu alt bölümde, $p_{ASRC_{NR}\cap AMBR'_{NR}}$ olasılığı tümleyen kuralı ile formülize edilmiştir. $\varphi(x,z,r)$ fonksiyonu, polinom girdi matrisinin kertesi r olan ve bu matrisinin satırları $w_{[f]}$'ye doğrusal bağımlı olan x düğümün permütasyonlarının sayısını hesaplamaktadır. Diğer bir yandan, $\tilde{\varphi}(x,z,r)$ fonksiyonu, polinom girdi matrisinin kertesi r olan, ancak içerisinde arızalanmış bir düğümü $ASRC_{NR}$ yöntemi ile tamir edecek herhangi iki düğüm bulundurmayan ve polinom matris girdileri $w_{[f]}$'ye doğrusal bağımlı olan x düğümün permütasyonlarının sayısını vermektedir. Böylece, $(\varphi(x,z,r)-\tilde{\varphi}(x,z,r))$ farkı, polinom girdi matrisinin kertesi r olan ve arızalanmış bir düğümü $ASRC_{NR}$ yöntemi ile tamir edebilecek en az iki düğüm bulunduran x düğümlük tüm permütasyonların sayısını vermektedir. Ayrıca, polinom girdi matrisinin kertesi d'den düşük olan x düğümün hiçbir alt kümesi arızalanmış

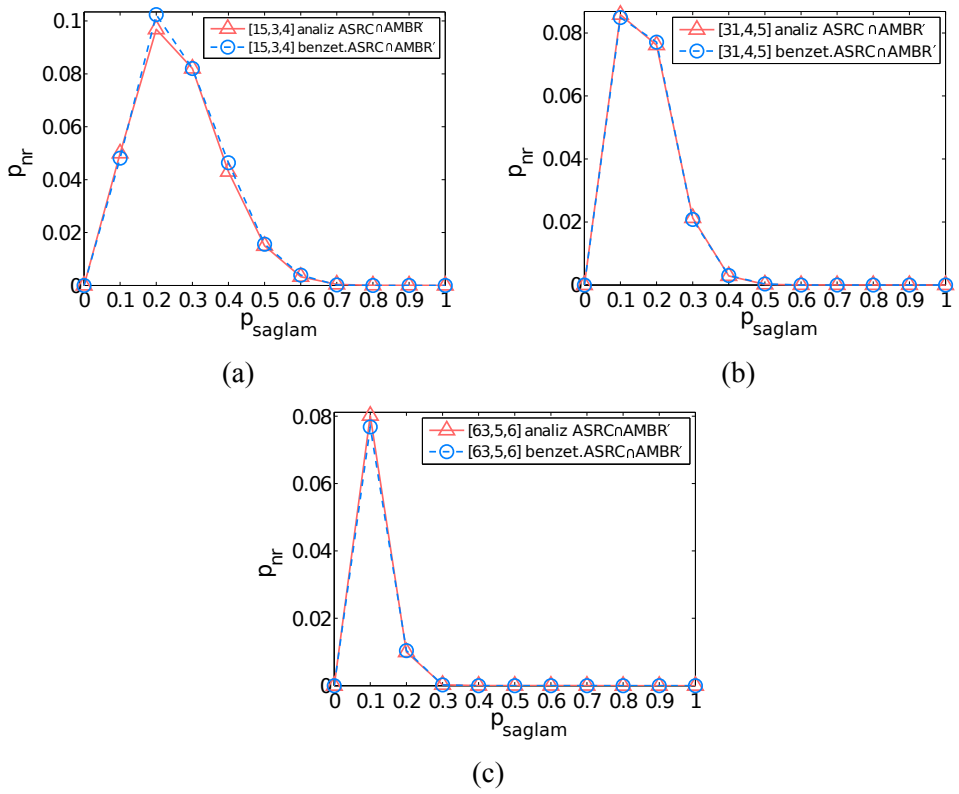
bir düğümü $AMBR_{NR}$ yöntemi ile tamir edemez. Böylece, $p_{ASRC_{NR}\cap AMBR'_{NR}}$ aşağıdaki şekilde hesaplanmaktadır:

$$p_{ASRC_{NR}\cap AMBR'_{NR}} = \sum_{x=2}^{n-1} \binom{n-1}{x} \sum_{r=2}^{d-1} \frac{(\varphi(x,z,r) - \tilde{\varphi}(x,z,r))}{P(n-1,x)} \times {p_{saglam}}^{x}(1-p_{saglam})^{n-x-1}, \quad z \geq d \tag{4.36}$$

Şekil 4.11 (a), Şekil 4.11 (b), Şekil 4.11 (c)'de, farklı HMBR kodlarında, $p_{ASRC_{NR}\cap AMBR'_{NR}}$ olasılığının kuramsal ve benzetim sonuçları görülmektedir. Bu sonuçlara göre $p_{ASRC_{NR}\cap AMBR'_{NR}}$ maksimum değerlerini, $[0.077, 0.12]$ arasında ve p_{saglam} 0.1 ya da 0.2 iken almaktadır. p_{saglam}'ın yüksek değerleri için $p_{AMBR_{NR}}$ 1'e yakınsar ve $p_{ASRC_{NR}\cap AMBR'_{NR}}$ gittikçe azalır. $ASRC_{NR}$ yöntemi $AMBR_{NR}$ yöntemi çalıştırılamadığı durumlarda düğüm tamirine çok büyük katkı sağlayamamakla birlikte katkı $\%8 - \%10$ aralığında gözardı edilemeyecek büyüklüktedir.

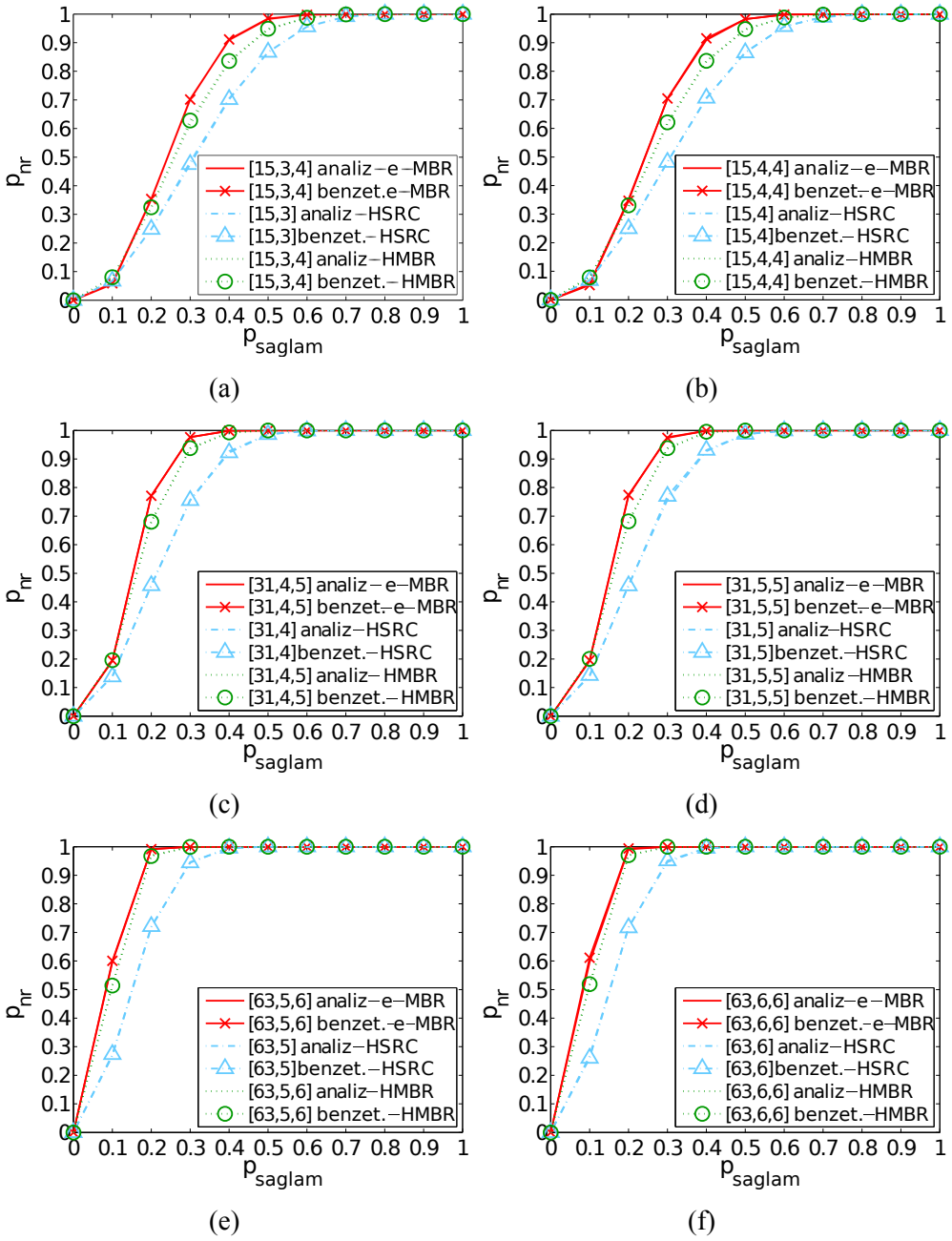


Şekil 4.10. $[n,k,d]$ HMBR kodlarında arızalanan bir düğümün $\tau = 2$ için $ASRC_{NR}$ düğüm tamiri yöntemi ile tamir edilemediği durumlarda $AMBR_{NR}$ yöntemi ile tamir edilebilme olasılığı a) $n = 15,\ k = 3,\ d = 4$ parametreleri için, b) $n = 31,\ k = 4,\ d = 5$ parametreleri için, c) $n = 63,\ k = 5,\ d = 6$ parametreleri için.



Şekil 4.11. $[n, k, d]$ HMBR kodlarında arızalanan bir düğümün $AMBR_{NR}$ düğüm tamiri yöntemi ile tamir edilemediği durumlarda $\tau = 2$ için $ASRC_{NR}$ yöntemi ile tamir edilebilme olasılığı a) $n = 15,\ k = 3,\ d = 4$ parametreleri için, b) $n = 31,\ k = 4,\ d = 5$ parametreleri için, c) $n = 63,\ k = 5,\ d = 6$ parametreleri için.

Şekil 4.12'de HMBR kodları, e-MBR kodları ve HSRC kodlarının düğüm tamiri başarım olasılıklarının kuramsal ve deneysel sonuçları verilmiştir (her şemada sonlu cisim olarak $\sigma = n+1$ kullanılmıştır). Bu sonuçlar HMBR kodlarının düğüm tamiri başarım olasılığının HSRC kodlarından biraz daha yüksek ve e-MBR kodlarından biraz daha düşük olduğunu göstermektedir. Sabit n, d ve σ değerleri için k parametresinin değişimi düğüm tamiri olasılıklarını değiştirmemektedir. HMBR kodlarının e-MBR kodlarına göre biraz daha düşük düğüm tamiri başarım olasılığına sahip olmasına karşın, bu kodların düğüm tamirinde farklı ihtiyaçlara yönelik birden fazla düğüm tamiri yöntemini desteklemesi hesapsal karmaşıklık ve I/O parametreleri açılarından avantaj sağlayabilmektedir.



Şekil 4.12. $[n, k, d]$ e-MBR, $[n, k]$ HSRC, $[n, k, d]$ HMBR kodlarında düğüm tamiri başarım olasılığı a) $n = 15$, $k = 3$, $d = 4$ parametreleri için, b) $n = 15$, $k = 4$, $d = 4$ parametreleri için, c) $n = 31$, $k = 4$, $d = 5$ parametreleri için, d) $n = 31$, $k = 5$, $d = 5$ parametreleri için, e) $n = 63$, $k = 5$, $d = 6$ parametreleri için, f) $n = 63$, $k = 6$, $d = 6$ parametreleri için.

4.3 HMSR Kodlama Şeması İle İlgili Elde Edilen Bulgular

Bu bölümde HMSR kodlarında verinin geri çatılması ve düğüm tamiri işlemlerinin hesapsal karmaşıklığı incelenmiştir. Ayrıca, HMSR kodlarında verinin başarılı bir şekilde geri çatılmasının ve başarılı bir şekilde düğüm tamirinin olasılığı hesaplanmıştır.

4.3.1 HMSR kodlarının hesaplama karmaşıklık analizi

Bu bölümde HMSR kodlarının hesaplama karmaşıklığı analizi yapılmıştır. Ayrıca, bu kodlama şemasının hesaplama karmaşıklığı e-MSR ve HSRC kodlama şemalarının hesaplama karmaşıklığıyla karşılaştırılmıştır.

$[n, k, d = 2k - 2]$ e-MSR kodlama şemasında bir düğümün tamiri için başta her yardımcı düğüm $\psi_{t_j} M$ ve ϕ_f vektörlerini çarpmaktadır. Bu çarpım $O(\alpha)$ mertebesinde toplama ve çarpma işlemi gerektirmektedir. Bunun ardından $(d \times d)$'lik Ψ_{tamir} matrisinin tersini alma işlemi gerçekleştirilir ki bu da $O(d^3)$ toplama ve çarpma karmaşıklığına sahiptir. Bu işlemden sonra, $(d \times d)$ boyutlu Ψ_{tamir}^{-1} ile $(d \times 1)$ boyutlu $\Psi_{tamir} M \phi_f$ matrisleri çarpılır, bu matris çarpımı $O(d^2)$ karmaşıklığına sahiptir. Tamir işleminde artık son olarak λ_f ve $\phi_f^t S_2$ çarpılır ve $\phi_f^t S_1$ ile toplanır bütün bunların karmaşıklığı $O(\alpha)$'dır.

e-MSR kodlarında verinin geri çatılması işleminde $(k \times \alpha)$'lık $\Psi_{VT} M$ matrisi ile $(\alpha \times k)$'lık Φ_{VT} matrisi çarpılmaktadır bu işlem ise $O(k^2\alpha)$'lık bir hesaplama karmaşıklığına sahiptir. Eşitlik (2.15)'teki iki bilinmeyenli iki denklemi çözmek c sabir bir değer olmak üzere $O(c\alpha)$ zaman karmaşıklığına sahiptir. P ve Q matrisleri için böyle $\frac{\alpha(\alpha-1)}{2}$ tane denklem çözülür. Yani bu işlemler $O(\alpha^3)$ toplama ve çarpma karmaşıklığına sahiptir. Ardından, Eşitlik (2.17)'den Eşitlik (2.18)'e geçebilmek için $(\alpha \times \alpha)$ boyutundaki bir matrisin tersi $O(\alpha^3)$'lük bir toplama ve çarpma karmaşıklığı ile alınır, ardından yine benzer şekilde Eşitlik (2.19)'dan S_1'i elde edebilmek için $O(\alpha^3) = O(k^3)$'lük bir karmaşıklık ile ikinci bir matrisin tersi alınır. Bu işlemler S_2 matrisini elde etmek için de gerçekleştirilir ancak hesaplama karmaşıklığı S_1'i elde etmekle aynı olacağı için toplam karmaşıklık değişmez.

HMSR kodlama şemasında iki farklı tamir etme yöntemi vardır. Bunlardan ilki olan $AMSR_{NR}$ yöntemine baktığımızda, bu yöntemin e-MSR ile aynı seviyede toplama ve çarpma karmaşıklığı bulunmaktadır, bu yöntemler arasındaki fark kullanılan sonlu cisim boyutundan kaynaklanmaktadır. $AMSR_{NR}$ tamir etme yönteminde ilk olarak kodlama matrisi satırları birbirinden bağımsız olan d tane yardımcı

düğüme bağlanılmaktadır. Burada bu d adet yardımcı düğümün her biri kendi depoladıkları $(1 \times \alpha)$ boyutlu $e_{t_i}M$ ile $(\alpha \times 1)$ boyutlu ϕ_f'i çarpar, bu çarpım $O(\alpha)$ karmaşıklığına sahiptir. Ardından, yeni-gelen düğüm $(d \times d)$'lik E_{yrd} matrisinin tersini $O(d^3)$'lük bir hesaplama karmaşıklığı ile alır. Ardından, ${E_{yrd}}^{-1}$ ile $E_{yrd}M$'i çarpar. Bu işlem $O(d^2)$ karmaşıklığına sahiptir. Bu işlemden sonra, yeni-gelen düğüm λ_f ile $\phi_f^t S_2$ ile çarpıp $\phi_f^t S_1$ ile toplar, bu işlem toplamda $O(\alpha)$ karmaşıklığına sahiptir.

HMSR kodlama şemasının $ASRC_{NR}$ düğüm tamiri yönteminde ise, toplamda en az dört yardımcı düğüme bağlanılır ve bu düğümlerden toplamda 4α sembol indirilir. Sonra, yeni-gelen düğüm bu kod satırlarını ikişerli olarak eşlenikleriyle gruplandırıp Eşitlik (3.63), Eşitlik (3.64), Eşitlik (3.65) ve Eşitlik (3.66)'yı kullanarak $\phi_{t_i}^t S_1$ ve $\phi_{t_i}^t S_2$'leri çözer. Burada $(\alpha \times 1)$'lık bir vektörle bir sembol çarpma işlemi gerçekleştirilir bu da $O(\alpha)$'lık bir hesaplama karmaşıklığı yaratır.

HMSR kodlama şemasının verinin geri çatılması işlemine bakacak olursak buradaki tek yöntem de e-MSR kodlama şemasındaki verinin geri çatılması işleminin bir adaptasyonu şeklindedir. Verinin geri çatılması işleminde $(k \times \alpha)$'lık $E_{VT}M$ matrisi ile $(\alpha \times k)$'lık Φ_{VT} matrisi çarpılmaktadır, bu işlem de $O(k^2\alpha)$'lık bir hesaplama karmaşıklığına sahiptir. İki bilinmeyenli iki denklemi çözmek $O(c\alpha)$'dır. P ve Q matrisleri için $\frac{\alpha(\alpha-1)}{2}$ tane eleman $P_{ij} + \lambda_i Q_{ij}$ ve $P_{ji} + \lambda_j Q_{ji}$ denklemleri kullanılarak çözülür. Yani bu işlemler $O(\alpha^3)$ toplama ve çarpma karmaşıklığına sahiptir. Sonra, Eşitlik (3.72)'deki matrisin sağ tarafındaki çarpan matrisinin tersi $O(\alpha^3)$'lük bir karmaşıklık ile alınır. Ardından, Eşitlik (3.74)'daki matrisin sol tarafındaki çarpan matrisinin tersi $O(\alpha^3) = O(k^3)$'lük bir karmaşıklık ile alınıp aynı ifade ile çarpılır.

Bozulmuş bir düğümü tamir etmek için, HSRC kodlama şeması $O(\tau)$ tane sonlu cisim toplama işlemi gerçekleştirir. τ en fazla k olabilir yani $O(\tau) = O(k)$'dır. HSRC kodları verinin geri çatılması işlemi için $O(k^3)$'lük bir karmaşıklığa sahiptir. Çünkü bu kodlama şeması $(k \times k)$'lık bir matrisin tersini alır ve $(k \times k)$'lık ve $(k \times 1)$'lik iki matris arasında bir çarpma işlemi gerçekleştirir. HSRC'de bu işlemlerin hepsi en azından $\mathbb{F}_{2^{((k-1)\lceil log_2^{n(d-k+1)}\rceil)}}$'lık bir sonlu cisim üzerinden gerçekleştirilmektedir.

Bu bölümde bahsedilen e-MSR, HSRC ve HMSR kodlarında kullanılması gereken sonlu cisim boyutları Çizelge 4.6'da göserilmiştir. Ayrıca, düğüm tamiri ve verinin geri çatılması işlemlerinin hesapsal karmaşıklıklarının özetleri sırasıyla Çizelge 4.7 ve Çizelge 4.8'de verilmiştir.

Çizelge 4.6. Kodlama şemalarında kullanılması gereken sonlu cisim boyutu

Kodlama Şeması	**Sonlu Cisim Boyutu**
e-MSR	$2^{\lceil logn(d-k+1)\rceil}$
HSRC	$2^{(k-1)\lceil logn(d-k+1)\rceil}$
HMSR	$q = max(2^d, (2^{\lceil log_2 n(d-k+1)\rceil} + 1 - (\lceil logn(d-k+1)\rceil - \lfloor logn(d-k+1)\rfloor)$

Çizelge 4.7. Düğüm tamiri işlemlerinin hesapsal karmaşıklık analizi özeti

Kodlama Şeması	**İşlem**	**Düğüm Tamiri**	
e-MSR (Rashmi et al., 2011b)	Çarpma	$O(d^3)$	
	Toplama	$O(d^3)$	
HSRC (Oggier and Datta, 2011b)	Çarpma	0	
	Toplama	$O(\tau)$	
		$AMSR_{NR}$	$ASRC_{NR}$
HMSR	Çarpma	$O(d^3)$	0
	Toplama	$O(d^3)$	$O(\alpha)$

Çizelge 4.8. Verinin geri çatılması işleminin hesapsal karmaşıklık analizi özeti

Kodlama Şeması	**İşlem**	**Verinin Geri Çatılması**
e-MSR	Çarpma	$O(k^3)$
	Toplama	$O(k^3)$
HSRC	Çarpma	$O(k^3)$
	Toplama	$O(k^3)$
HMSR	Çarpma	$O(k^3)$
	Toplama	$O(k^3)$

4.3.2 HMSR kodlarında verinin geri çatılması olasılığı

HMSR kodlarında bir verinin geri çatılması işleminin başarılı bir şekilde gerçekleştirilmesi için çalışır durumdaki düğümler arasında öyle k tane düğüm olmalıdır ki bu k düğümün ϕ_i^t vektörlerinin her α tanelik alt kümesi doğrusal bağımsız olmalıdır. Yani bu k vektörün hepsinin doğrusal bağımsız olması gerekmemektedir. Burada, i'inci düğümün ϕ_i^t vektörünün ilk elemanı w_i ile ifade edilmektedir.

Moore matrisinden (Goss, 1998) biliyoruz ki herhangi w_x, w_y, w_z, $\ldots$, w_s $\in \mathbb{F}_{q^l}$ sembolleri $\mathbb{F}_q$'de doğrusal bağımsız ise bu değerlerin $q^{l^j}, j \in N$ üzeri katlarının oluşturduğu vektörleri (Moore matirisinin satırlarını oluşturacak şekilde) $\mathbb{F}_{q^l}$ üzerinden doğrusal bağımsızdır. Yani ilk koşul sağlandığında ϕ_i^t vektörleri de doğrusal bağımsızdır. Diğer bir deyişle, seçilen düğümlerin ilgili kodlama satırlarının ilk elemanından oluşan sembol kümesinin her α'lık alt kümesi $\mathbb{F}_2$ üzerinden doğrusal bağımsız ise orijinal veri geri çatılabilir demektir. Sistemde bu koşulu sağlayan herhangi bir k düğümlük grup bulunabilirse orijinal veri geri çatılabilmektedir. p_{saglam} bir düğümün çalışır durumda olma olasılığı olarak atanmıştır ve bu olasılık tüm düğümler için aynıdır. Aşağıda, HMSR kodu kullanan bir sistemde verinin geri çatılması olasılığının alt sınırı hesaplanmıştır.

$$p_{dr_{HMSR}} \geq \sum_{x=k}^{n} \sum_{r=\alpha}^{z} \frac{\vartheta(n,x,z,r,k)}{P(n,x)} \binom{n}{x} {p_{saglam}}^{x} (1-p_{saglam})^{n-x}) \qquad (4.37)$$

Burada z bir sembolun temsil edildiği bit sayısıdır. $\frac{\vartheta(n,x,z,r,k)}{P(n,x)}$, x adet çalışan düğümü bulunan ve HMSR kodlama kullanan bir sistemde veri geri çatılması işlemi

için gerekli olan koşulların bulunması olasılığının alt sınırını hesaplamaktadır.

$$\vartheta(n,x,z,r,k) = \begin{cases} \prod_{i=0}^{x-1}(n-2^{i+1}+2),\ x=(r+1),\ r=\alpha. \\ R_2(n,x-1,z,r,k), \\ \qquad\qquad x \leq n/4,\ x>(r+1),\ r=\alpha, r \leqslant z \\ R_2(n,x,z,r,k), \\ \qquad\qquad x>r,\ r \leqslant z,\ r>\alpha\ || \\ \qquad\qquad x>n/4, x \geq (r+1), r \geq \alpha, r \leq z\ || \\ \qquad\qquad x=r, r=k, r \leq z \\ 0, \qquad\qquad x<r,\ r>0. \end{cases} \tag{4.38}$$

Eşitlik (4.38)'de $\vartheta(n,x,z,r,k)$ fonksiyonu, içerisinde veri geri çatılması için gerekli koşulu sağlayan en az bir grup k düğüm bulunduran x düğümlük permütasyon sayısının alt sınırını vermektedir. Bu olasılığı bulmak için önce sistemde sadece x adet çalışır düğüm bulunurken, bu x düğümün içinde bulunan kaç farklı k düğümlük kombinasyonun tüm α'lık alt kümelerinin vektörlerinin $r=\alpha$ kertesine sahip olduğu sayılacaktır. Bunu hesaplayan fonksiyona $R_2(n,x,z,r,k)$ adı verilmiştir. Burada, n toplam düğüm sayısı, x çalışır durumda olan düğüm sayısı, z bit sayısı, r ise kertedir.

Çalışır durumda olan düğüm sayısı k'ya eşit ve bu düğümlerin ilgili Φ matrisi satırlarının kertesi α olan permütasyonların sayısı $\prod_{i=0}^{x-1}(n-2^{i+1}+2)$ ile hesaplanmaktadır.

Çalışır durumda olan düğüm sayısı $x > \frac{n}{4}, x \geq k$ ve ilgili Φ matrisi satırlarının kertesi α olan her düğüm kümesi, HMSR kodlama şemasının gerektirdiği veri geri çatılması işlemi için gerekli koşulları sağlamaktadır. Bu durum Eşitlik (4.38)'in üçüncü durumunda görülmektedir.

Eğer $x=r$ ve $r \neq k$ ise, Φ matrisinin kertesi r olan x tane düğüm kombinasyonu sayısı aşağıda hesaplanmıştır.

$$R_2(n,x,z,r=x,k) = \prod_{j=1}^{r}(n-2*2^{(j-1)}+2) \tag{4.39}$$

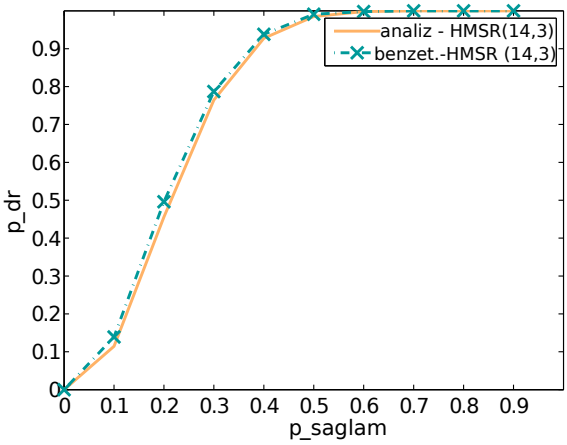
Eğer $x = r$ ve $r = k$ ise α vektörlük tüm alt kümelerinin kertesi α olan k düğümlük kombinasyonlar hesaplanmıştır.

$$R_2(n, x, z, r = x = k, k) = (n - 2 * 2^{\alpha} + 2) \prod_{j=1}^{r-1} (n - 2 * 2^{(j-1)} + 2) \quad (4.40)$$

Eğer $r \leqslant x$ ise Φ matrisinin kertesi $r - 1$ olan bir düğüm kümesine doğrusal bağımsız vektöre sahip bir düğüm ekleme permütasyonlarının ve Φ matrisinin kertesi r olan $x-1$ düğümden oluşan bir kümeye doğrusal bağımlı vektöre sahip bir düğüm ekleme permütasyonlarının sayıları hesaplanır.

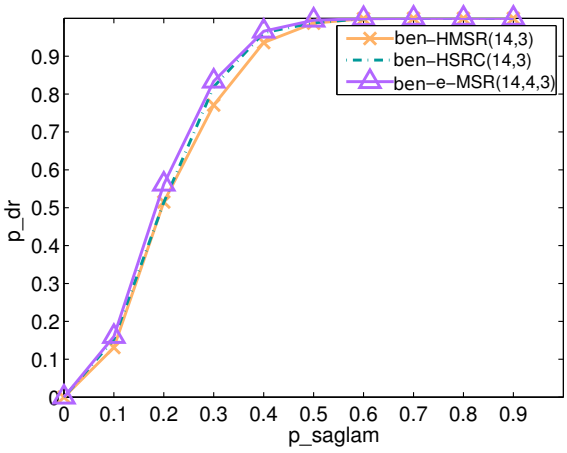
$$\begin{aligned} R_2(n, x, z, r, k) = {} & R_2(n, x-1, z, r-1, k) * (n - 2 * 2^{(r-1)} + 2) \\ & + R_2(n, x-1, z, r, k) * (2^{(r+1)} - x - 1),\ r \leqslant x \end{aligned} \quad (4.41)$$

Şekil 4.13'te HMSR kodlama şemasında, düğümlerin çeşitli p_{saglam} çalışma olasılıkları altında veri geri çatılması başarım olasılıkları kuramsal olarak hesaplanmıştır ve benzetim sonuçlarıyla desteklenmiştir.



Şekil 4.13. HMSR kodlarında verinin geri çatılması olasığı analiz ve benzetim sonuçları.

Bu tezde ortaya konulan HMSR kodlama şeması ile mevcut HSRC ve e-MSR kodlarında veri geri çatılması olasığı benzetim sonuçları Şekil 4.14'te verilmiş olup e -MSR kodları en yüksek başarım olasığına sahiptir. e-MSR kodlarından sonra ise HSRC kodları gelmektedir. HMSR kodları verinin geri çatılması işleminde en düşük başarım olasılıklarına sahip olup bu olasılıklar diğer iki kodlama şemasının olasılıklarına yakındır. HMSR kodlarının verinin geri çatılması işlemi başarım olasılıklarının diğer kodların başarım olasılıklarına en fazla $\%7,8$ uzaklıkta sonuç verdiği görülmektedir.



Şekil 4.14. HMSR, HSRC ve e-MSR kodlarında verinin geri çatılması başarım olasığı benzetim sonuçları.

4.3.3 HMSR kodlarında düğüm tamiri olasılığı

HMSR kodlarının da $AMSR_{NR}$ ve $ASRC_{NR}$ olmak üzere iki farklı düğüm tamiri yöntemi olduğu için iki yöntemdeki düğüm tamiri başarım olasılıkları aşağıdaki alt bölümlerde ayrı ayrı incelenmiştir.

4.3.3.1 AMSR_{NR} yöntemi ile düğüm tamiri olasılığı

HMSR kodlama şemasının düğüm tamiri olasılığının bulunması için önce kodlama matrisi olan E'nin herhangi bir d satırlık alt kümesinin tersinir olma olasılığının incelenmesi gerekmektedir. Düğüm tamirinde yeni-gelen düğümün bağlandığı d farklı düğümün ilgili satırlarından oluşan matris $(d \times d)$ boyutlu E_{yrd} olarak adlandırılmıştır. E_{yrd} matrisi aşağıdaki gibi dört alt kare matris ile gösterilebilmektedir. E_{yrd} matrisinin tersinir olması için bu matrisin belirtecinin (*determinant*) sıfırdan büyük olması gerekmektedir (Karpovsky et al., 2008).

$$E_{yrd} = \begin{bmatrix} A & \Lambda_{\alpha_1} A \\ B & \Lambda_{\alpha_2} B \end{bmatrix} \tag{4.42}$$

Burada A matrisi E_{yrd} matrisinin ilk α satır ve sütununun bulunduğu alt matristir. $\Lambda_{\alpha_1} A$ matrisi ise E_{yrd} matrisinin $\alpha + 1, d$ arası sütünlarının (bu sütunlar dahil olmak üzere) ilk α satırından oluşmaktadır. B matrisi ise E_{yrd} matrisinin $\alpha + 1$ ve d satırlarının arasındaki ilk α sütunun bulunduğu kısmı temsil etmektedir. Benzer şekilde, $\Lambda_{\alpha_2} B$ matrisi de E_{yrd} matrisinin $\alpha + 1$ ve d satırlarının ikinci α sütunun olduğu alt matristir.

Sırasıyla $(t\times t)$, $(t\times m)$, $(m\times t)$, ve $(m\times m)$ boyutlarına sahip $J,\ K,\ L,\ N$ matrislerinin verildiği düşünülsün ($t > 0,\ m > 0$). Eğer J matrisi tersinir bir matris ise aşağıdaki eşitlikten yararlanılarak $\begin{pmatrix} J & K \\ L & N \end{pmatrix}$ matrisinin belirteci hesaplanabilir (Brookes, 2011).

$$det\begin{pmatrix} J & K \\ L & N \end{pmatrix} = det(J)det(N - LJ^{-1}K) \tag{4.43}$$

Eşitlik (4.43) kullanılarak A matrisinin tersinir olduğu varsayıldığında E_{yrd} matrisinin belirteci de aşağıdaki gibi hesaplanabilir:

$$det\begin{pmatrix} A & \Lambda_{\alpha_1}A \\ B & \Lambda_{\alpha_2}B \end{pmatrix} = det(A)det(\Lambda_{\alpha_2}B - BA^{-1}\Lambda_{\alpha_1}A) \tag{4.44}$$

Sonlu cisim aritmetiği kullanıldığı için Eşitlik (4.44)'deki belirtecin sıfırdan farklı olması E_{yrd} matrisinin tersinir olduğu anlamına gelmektedir. Başlangıçta, A matrisinin tersinir olduğu varsayılmıştır. A matrisi tersinir olarak kabul edildiği için A^{-1} matrisi de tersinirdir. A^{-1} matrisinin tersinir olduğu için eşitlik (4.44)'deki $det(\Lambda_{\alpha_2}B - BA^{-1}\Lambda_{\alpha_1}A)$ fonksiyonu ile çarpılmasında bir sakınca yoktur.

$$det\begin{pmatrix} A & \Lambda_{\alpha_1}A \\ B & \Lambda_{\alpha_2}B \end{pmatrix} det(A^{-1}) = det(A)det(\Lambda_{\alpha_2}B - BA^{-1}\Lambda_{\alpha_1}A)det(A^{-1}) \tag{4.45}$$

$$det\begin{pmatrix} A & \Lambda_{\alpha_1}A \\ B & \Lambda_{\alpha_2}B \end{pmatrix} det(A^{-1}) = det(A)det(\Lambda_{\alpha_2}BA^{-1} - BA^{-1}\Lambda_{\alpha_1}AA^{-1}) \tag{4.46}$$

$$det\left(E_{yrd}\right) det(A^{-1}) = det(A)det(\Lambda_{\alpha_2}BA^{-1} - BA^{-1}\Lambda_{\alpha_1}) \tag{4.47}$$

Eşitlik (4.47)'ye bakıldığında E_{yrd} matrisinin tersinir olması, A ve A^{-1} matrislerinin tersinir olduğu varsayıldığından $det(\Lambda_{\alpha_2}BA^{-1} - BA^{-1}\Lambda_{\alpha_1})$ değerinin sıfırdan farklı olmasına bağlıdır. Burada BA^{-1} matrisinin yerine, χ yazılsın. Bundan sonra artık aşağıdaki belirtecin sıfırdan farklı olma olasılığı üzerinde çalışılacaktır.

$$det(\Lambda_{\alpha_2}\chi - \chi\Lambda_{\alpha_1}) \tag{4.48}$$

χ matrisi $(\alpha \times \alpha)$ boyutlu bir matristir. χ, Λ_{α_1} ve Λ_{α_2} matrisleri açık formu göste-

rilirse aşağıdaki belirtec ifadesi oluşmaktadır:

$$
\begin{aligned}
det(\Lambda_{\alpha_2} &\begin{pmatrix} x_1 & x_2 & \dots & x_\alpha \\ x_{\alpha+1} & x_{\alpha+2} & \dots & x_{2\alpha} \\ \vdots & \vdots & \ddots & \vdots \\ x_{\alpha(\alpha-1)+1} & x_{\alpha(\alpha-1)+2} & \dots & x_{\alpha^2} \end{pmatrix} \\
&- \begin{pmatrix} x_1 & x_2 & \dots & x_\alpha \\ x_{\alpha+1} & x_{\alpha+2} & \dots & x_{2\alpha} \\ \vdots & \vdots & \ddots & \vdots \\ x_{\alpha(\alpha-1)+1} & x_{\alpha(\alpha-1)+2} & \dots & x_{\alpha^2} \end{pmatrix} \Lambda_{\alpha_1})
\end{aligned}
\tag{4.49}
$$

$$
\begin{aligned}
det(&\begin{pmatrix} x_1(\lambda_{11}) & x_2(\lambda_{11}) & \dots & x_\alpha(\lambda_{11}) \\ x_{\alpha+1}(\lambda_{12}) & x_{\alpha+2}(\lambda_{12}) & \dots & x_{2\alpha}(\lambda_{12}) \\ \vdots & \vdots & \ddots & \vdots \\ x_{\alpha(\alpha-1)+1}(\lambda_{1\alpha}) & x_{\alpha(\alpha-1)+2}(\lambda_{1\alpha}) & \dots & x_{\alpha^2}(\lambda_{1\alpha}) \end{pmatrix} \\
&- \begin{pmatrix} x_1(\lambda_{21}) & x_2(\lambda_{22}) & \dots & x_\alpha(\lambda_{2\alpha}) \\ x_{\alpha+1}(\lambda_{21}) & x_{\alpha+2}(\lambda_{22}) & \dots & x_{2\alpha}(\lambda_{2\alpha}) \\ \vdots & \vdots & \ddots & \vdots \\ x_{\alpha(\alpha-1)+1}(\lambda_{21}) & x_{\alpha(\alpha-1)+2}(\lambda_{23}) & \dots & x_{\alpha^2}(\lambda_{2\alpha}) \end{pmatrix})
\end{aligned}
\tag{4.50}
$$

$$
det(\begin{pmatrix} x_1(\lambda_{11}-\lambda_{21}) & x_2(\lambda_{11}-\lambda_{22}) & \dots & x_\alpha(\lambda_{11}-\lambda_{2\alpha}) \\ x_{\alpha+1}(\lambda_{12}-\lambda_{21}) & x_{\alpha+2}(\lambda_{12}-\lambda_{22}) & \dots & x_{2\alpha}(\lambda_{12}-\lambda_{2\alpha}) \\ \vdots & \vdots & \ddots & \vdots \\ x_{\alpha(\alpha-1)+1}(\lambda_{1\alpha}-\lambda_{21}) & x_{\alpha(\alpha-1)+2}(\lambda_{1\alpha}-\lambda_{22}) & \dots & x_{\alpha^2}(\lambda_{1\alpha}-\lambda_{2\alpha}) \end{pmatrix})
\tag{4.51}
$$

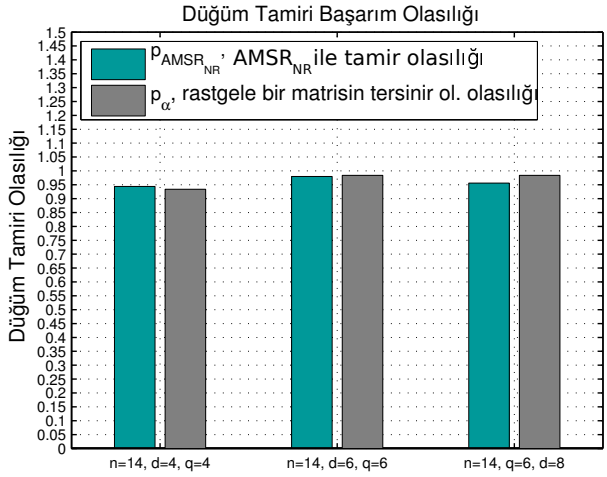
Eşitlik (4.51)'deki tüm λ_{1i} ve λ_{2j}, $i,j \in \{1,2,\dots,\alpha\}$ rastgele seçildiği için $\lambda_{1i}-\lambda_{2j}$ değerleri de sonlu cisim kümesi içerisinden rastgele seçilmiş olacağından χ matrisinde herhangi bir düzen olsa bile $\Lambda_{\alpha_2}\chi-\chi\Lambda_{\alpha_1}$ matrisi en azından kısmen rastgele oluşmaktadır. Tersinir bir A matrisi bulunduğunda, $\Lambda_{\alpha_2}\chi-\chi\Lambda_{\alpha_1}$ matrisinin belirteci de sıfırdan farklı ise E_{yrd} matrisinin belirteci de sıfırdan farklı olmalıdır,

bu durumda E_{yrd} matrisi tersinir olacağı için tamir başarım olasılığı 1 olmaktadır. $AMSR_{NR}$ yöntemi ile düğüm tamiri olasılığının analitik analizi tamamlanmamıştır. Yapılan çalışmalarda $\Lambda_{\alpha_2}\chi - \chi\Lambda_{\alpha_1}$ matrisinin rastgele olduğu ispatlanabilirse anılan olasılığın Eşitlik (4.52)'de verilen P_α'nın kullanılmasıyla bulunabileceği saptanmıştır. Dolayısıyla $AMSR_{NR}$ başarım olasılığı ile ilgili matrisin tersinir olma olasılığı arasında doğrudan bir ilişki olduğu görülmüştür. Bu hipotezi test etmek için de $AMSR_{NR}$ düğüm tamiri başarım olasılığı ile $(\alpha \times \alpha)$ boyutunda rastgele bir matrisin tersinir olma olasılığı yapılan bir benzetimle karşılaştırılıp bu değerlerin birbirleriyle büyük ölçüde yakın olduğu görülmüştür.

$(\alpha \times \alpha)$ boyutunda, elemanları $\mathbb{F}_{q^h}$ ($q \in \mathbb{P}$) kümesi içerisinden eş dağılıma göre seçilmiş olan bir matrisin tersinir olması olasılığı, P_α, aşağıda verilmiştir:

$$P_\alpha = (1 - \frac{1}{q})(1 - \frac{1}{q^2}) \ldots (1 - \frac{1}{q^\alpha}) \text{ (Charlap et al., 1990)} \qquad (4.52)$$

Şekil 4.15'te rastgele seçilen k düğüm ile $AMSR_{NR}$ yönteminde düğüm tamiri başarım olasılığı ile $(\alpha \times \alpha)$ boyutlu rastgele matrislerin tersinir olma olasılığının karşılaştırılması görülmektedir.



Şekil 4.15. $AMSR_{NR}$ yönteminde düğüm tamiri başarım olasılığı ile $(\alpha \times \alpha)$ boyutlu rastgele oluşturulan bir matrisin tersinir olma olasılığının karşılaştırılması.

4.3.3.2 $ASRC_{NR}$ yöntemi ile düğüm tamiri olasılığı

Bu bölümde $ASRC_{NR}$ yöntemi ile düğüm tamiri işleminin başarım olasılığı analiz edilmiştir. Bu yöntemin analizi $n = 2(2^\alpha - 1)$ değeri için gerçekleştirilmiştir. Toplamda n tane düğümün olduğu bir sistemde çalışır durumda x düğüm

bulunuyorsa bunlar herhangi bir $C(n,x)$'lik bir kombinasyonla seçilebilmektedir. $ASRC_{NR}$ yöntemi ile tamirin mümkün olmadığı kombinasyonların tüm kombinasyonlardan çıkartılmasıyla $ASRC_{NR}$ yöntemi ile düğüm tamiri işleminin başarım olasılığı elde edilebilir. İçerisinde $ASRC_{NR}$ yöntemi ile tamir olasılığı barındırmayan x düğümlük kombinasyonlar iki şekilde hesaplanabilmektedir: İlk kombinasyonda eşlenik kodlama satırlarına sahip herhangi iki düğüm bulunmamaktadır. Bu durum sadece $x \leq n/2-1$ olduğunda gerçekleşebilir ve bu durumu oluşturabilen x düğümlük kombinasyonlar aşağıdaki gibi sayılabilir:

$$\vartheta(x) = \prod_{i=0}^{x-1} \frac{(n-2-(2(i-1)))}{x!}, \;\; x \leq n/2-1 \tag{4.53}$$

İkinci kombinasyon türünde ise birbiriyle eşlenik kodlama satırlarına sahip olan ancak arızalanan düğümü tamir etmek için gerekli eşlenik grupların bulunmadığı düğümler sayılmaktadır. Bu kombinasyon türü de aşağıdaki şekilde sayılabilir:

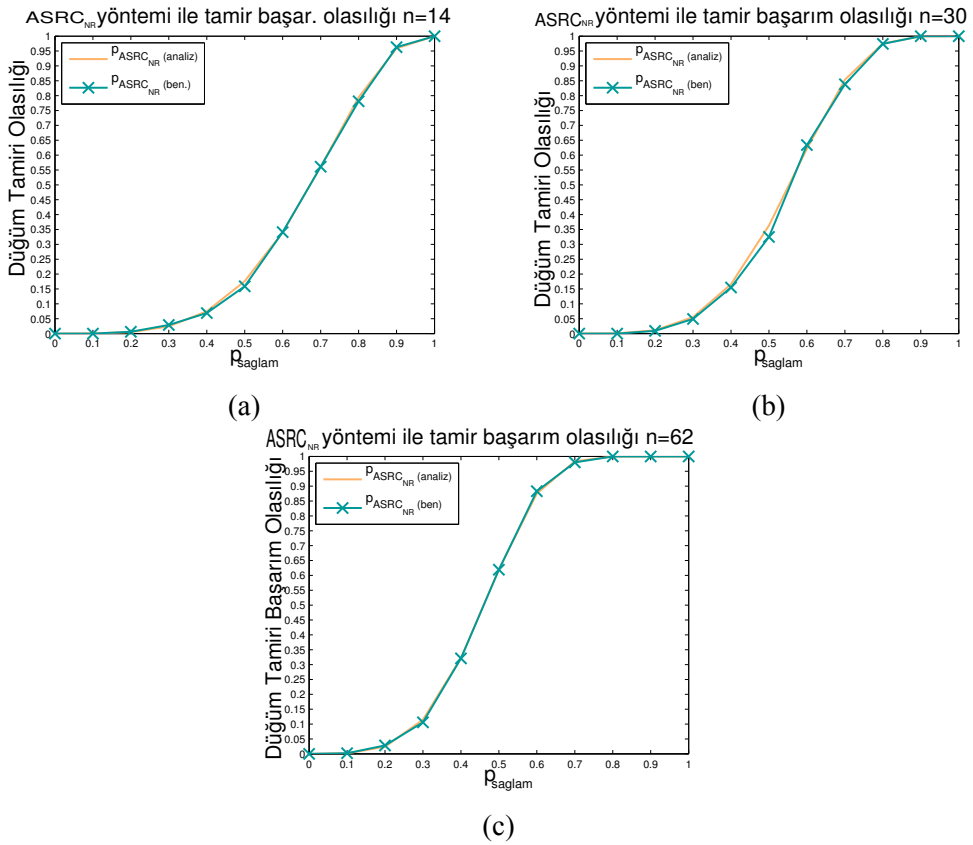
$$\begin{aligned}\zeta(x) = \sum_{i=1}^{\frac{x}{2}} & \frac{(\prod_{j=1}^{i}((n/2-1)-2(j-1)))}{i!} \\ & \times \frac{(\prod_{z=2i+1}^{x} n-2-2i-2(z-2i-1)))}{(x-2i)!}\end{aligned} \tag{4.54}$$

Eşitlik (4.54)'te $2i$ değeri o döngüde sayılan kombinasyonun içerisinde eşleniği bulunan düğüm sayısıdır. İçerisinde j bulunan çarpım ise $2i$ tane eşleniğe sahip düğüm kombinasyonlarını hesaplamak için kullanılmaktadır. z ise aynı döngüde eşleniğe sahip olmayan düğümlerin kombinasyonlarını bulmak için kullanılmaktadır.

$$p = \frac{C(n,x) - (\vartheta(x) + \zeta(x))}{C(n,x)} \tag{4.55}$$

Eşitlik (4.55)'te ise sistemde çalışır durumda x düğüm bulunuyorken $ASRC_{NR}$ yöntemi ile düğüm tamiri başarım olasılığı hesaplanmıştır.

Bir düğümün $ASRC_{NR}$ yöntemi ile farklı erişilebilirlik olasılıklarına göre tamir edilebilme olasılığının kuramsal ve benzetim sonuçları Şekil 4.16'da görülmektedir. Görüldüğü gibi analiz ve benzetim sonuçları büyük ölçüde örtüşmektedir.



Şekil 4.16. HMSR kodlarında arızalanmış düğümün $\tau = 2$ için $ASRC_{NR}$ yöntemi ile tamir edilme olasılığı a) $n = 14$ için, b) $n = 30$ için, c) $n = 62$ için.

4.4 Kümeleme Tabanlı Dağıtık Depolama Sistemi İle İlgili Elde Edilen Bulgular

Bu alt bölümde kümeleme tabanlı dağıtık depolama sistemi çalışması hakkında elde edilen bulgular incelenmiştir. Bu kapsamda alt bölüm 3.4'de tasarlanan kümeleme tabanlı olan ve olmayan dağıtık depolama sistemleri ağ benzetim ortamında gerçekleştirilmiştir.

Dağıtık depolama sistemlerinde kodlama şemalarının getirdiği birtakım maliyetler bulunmaktadır. Bunlar arasında kodlama şemasının yerine getirdiği işlemlerde kullanılan CPU süresi, bu işlemlerin tamamlanması için geçen toplam süre ve bu işlemlerin ağda oluşturduğu trafik bulunmaktadır. Kodlama şemasının yerine getirdiği işlemler, orijinal verinin kodlanarak genişletilmesini, kodlanmış verilerin kullanılarak orijinal verinin geri çatılmasını, arızalanan bir düğümde bulunan kodlanmış veri parçasının yeniden oluşturulmasını yani düğüm tamirini kapsamaktadır.

Bazı kodlama şemalarında düğüm tamiri ve verinin geri çatılması işlemleri

için yeterli sayıda çalışır durumda düğüm bulunması yeterli iken bazı kodlama şemalarında yeterli sayıdaki her düğüm istenilen işlemi yerine getiremez. Bu durum orijinal verinin dayanıklılığını etkilediği için bir maliyet faktörü olarak görülebilmektedir.

Bu çalışma kapsamında, ilk üç alt bölümde öncelikle HSRC (Oggier and Datta, 2011b), e-MBR (Rashmi et al., 2011b) ve HMBR (Haytaoglu and Dalkilic, 2013a) kodlama şemaları ile çeşitli büyüklüklerdeki verilerin kodlanmasında geçen süreler, ardından düğüm tamiri ve verinin geri çatımı işlemleri için kullanılan toplam CPU süreleri ölçülmüştür. Ölçülen süreler ağ işlemleri için harcanan süreleri içermemektedir. Bu alt bölümde gerçekleştirilen tüm testlerde kodlama şeması parametreleri olarak $n = 15,\ k = 3$ ve $d = 4$ alınmıştır.

Kümeleme tabanlı dağıtık depolama sistemlerinde düğüm tamiri ve verinin geri çatımı için gerekli olan toplam süreler ve mesaj sayıları alt bölüm 4.4.4 ve alt bölüm 4.4.5'te incelenmiştir. Ölçülen sürelerin içerisinde ağ iletişimi için gereken süreler de bulunmaktadır.

Kümeleme tabanlı olmayan dağıtık depolama sisteminde her bir grubun düğüm tamiri ve verinin geri çatımı işlemleri için kullandığı süre ve mesaj sayıları ise alt bölüm 4.4.6 ve alt bölüm 4.4.7'de verilmiştir. Burada da ölçülen sürelerin içerisinde ağ iletişimi için gereken süreler de bulunmaktadır.

Kümeleme tabanlı olan ve olmayan dağıtık depolama sistemlerinde verinin geri çatımı işleminden çok düğüm tamiri işleminin gerçekleştirildiği varsayılmıştır. Bunun bir örneği de Facebook'un kullandığı veri ambarlarıdır (Rashmi et al., 2013a).

Yukarıda sözü geçen işlemler için kullanılan süreleri ve mesaj sayılarını ölçmek için bu sistemler ns-3 (ns-3, 2015) ağ benzetim programı ile gerçekleştirilmiştir. Bunun için her testte 45 düğümden oluşan farklı bir ağ Waxman (Waxman, 1988) topolojisi ile BRITE aracı kullanılarak (Medina et al., 2000) üretilmiştir. Ağdaki düğümler fiziksel yakınlıkları ve ağ özellikleri açısından 15 düğümlük üç farklı kategoriye ayrılmıştır. İlk kategoride kendi aralarında bağlı bir çizge oluşturan 15 düğümün arasındaki hatların bant genişliği 12 Mbps-120 Mbps arasında eş dağılımla atanmıştır. Bu düğümler arasındaki hatların yayılım gecikmesi ise 80 ms olarak atanmıştır. Bağlı bir çizge oluşturan ikinci kategorideki 15 düğümün arasındaki hatların bant genişlikleri 9 Mbps-90 Mbps arasından eş dağılımla atanmıştır. Bu kategorideki hatların yayılım gecikmesi ise 50 ms olarak verilmiştir. Üçüncü kategorideki

düğümler de kendi arasında bağlı bir çizge oluşturmaktadır. Bu düğümler arasındaki hatların bant genişliği ise 10 Mbps-100Mbps arasından eş dağılımla seçilmektedir. Bu hatların yayılım gecikmesi ise 50 ms'dir.

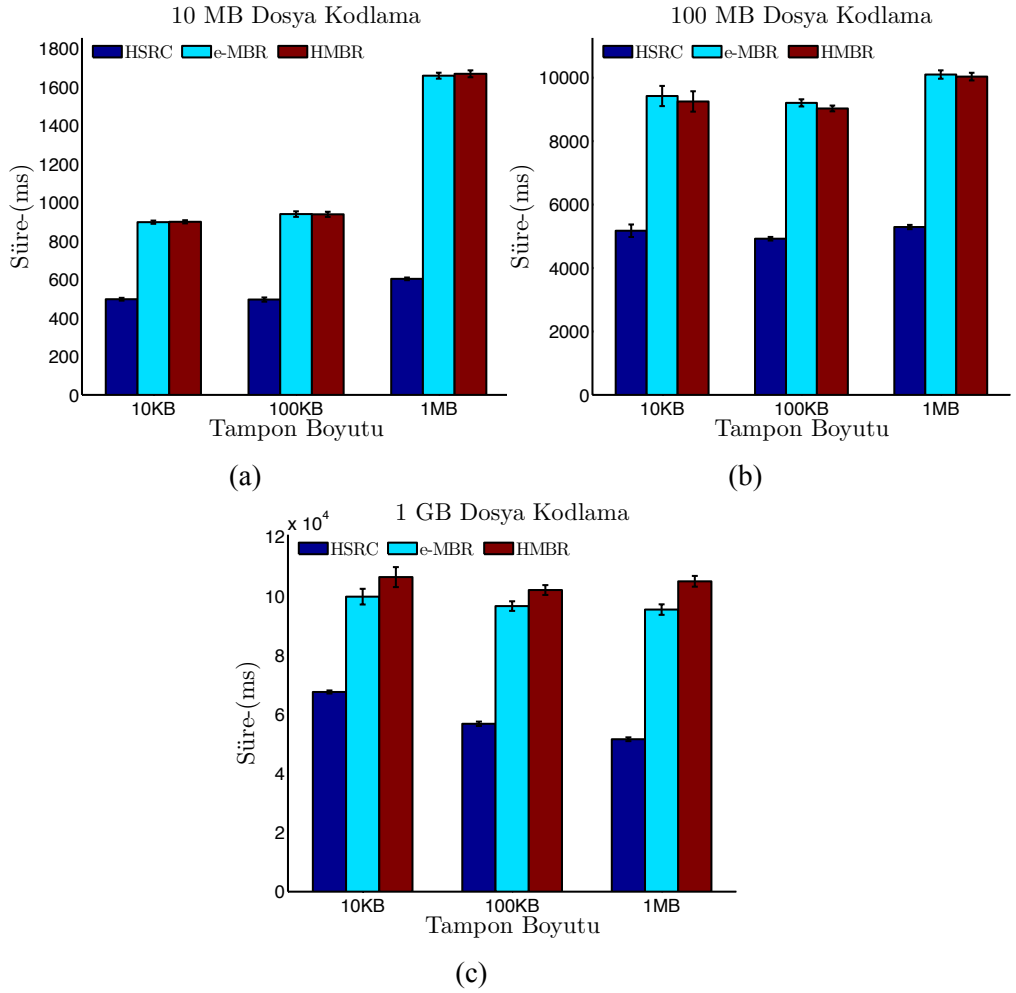
Gerçekleştirilen testlerde düğüm tamiri işlemi için ağda her bir küme ya da grubun içerisinde rastgele bir düğüm arızalı olarak seçilir. Yeni-gelen düğümler yeterli koşulları sağlayan rastgele düğümlere bağlanarak düğüm tamiri işlemini gerçekleştirmektedir. Verinin geri çatılması işlemi de aynı şekilde yeterli koşulları sağlayan rastgele düğümler kullanılarak gerçekleştirilmektedir. Kümeleme tabanlı olan ve olmayan dağıtık depolama sistemlerinde eş zamanlı gerçekleşen üç farklı düğüm tamiri/verinin geri çatılması işlemi incelenmiştir.

Kümeleme tabanlı olan ve olmayan dağıtık depolama sistemlerinin maliyet analizi ise alt bölüm 4.4.8'de verilmiştir. Buradaki sonuçların, sistem tasarımcılarının kendi platformlarına en uygun sistemi seçebilmesinde faydalı olabileceği düşünülmektedir.

4.4.1 Kodlama şemalarında orijinal verinin kodlanması süreleri

Bu alt bölümde HSRC, e-MBR ve HMBR kodlama şemalarında 10 MB, 100 MB ve 1 GB büyüklüklerindeki dosyaların yedekli bir şekilde depolanması için kodlanması sırasında geçen süreler incelenmiştir. Bunun için kodlama işleminde çeşitli tampon büyüklükleri kullanılmıştır. Bu tampon büyüklükleri dosyalardan bir seferde okunması gereken veri hacmini etkilemektedir.

Şekil 4.17'de kodlama şemalarının kodlama işlemi için kullandığı süreler görülmektedir. Buna göre HSRC kodlama şeması en düşük süreyi kullanırken, e-MBR ve HMBR kodlama şemaları birbirine yakın süreler kullanmaktadır. HSRC kodlama şeması ek olarak daha az kodlanmış veri üretirken, e-MBR ve HMBR kodları mesaj matrisinin oluşturulması için fazladan IO işlemleri yapmaktadır. Dolasıyla sonuçlar beklendiği gibidir. 10 MB büyüklüğündeki dosyanın kodlanması işlemi için farklı tampon boyutlarının kodlama sürelerinde etkisi daha çok görülmektedir. Bunun nedeninin kodlama şemaları tarafından kodlama işlemi öncesi gerçekleştirilen doldurma (padding) olduğu düşünülmektedir. 10 ve 100 MB büyüklüğündeki dosyalar için 1MB tampon boyu, 1 GB büyüklüğündeki dosyalar için ise 10 KB tampon boyunun diğer tampon büyüklüklerine göre verimsiz olduğu görülmüştür.



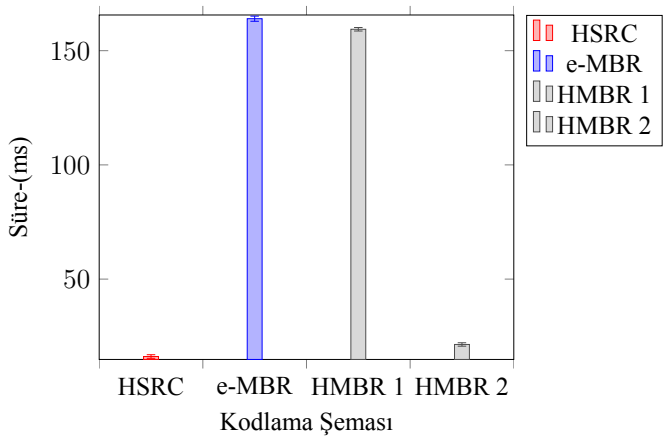
Şekil 4.17. Kodlama şemalarında kodlama işlemi için geçen süre a) 10 MB dosyanın kodlanması için geçen süre, b) 100 MB dosyanın kodlanması için geçen süre, c) 1 GB dosyanın kodlanması için geçen süre.

4.4.2 Kodlama şemalarında düğüm tamiri işlemlerinin süreleri

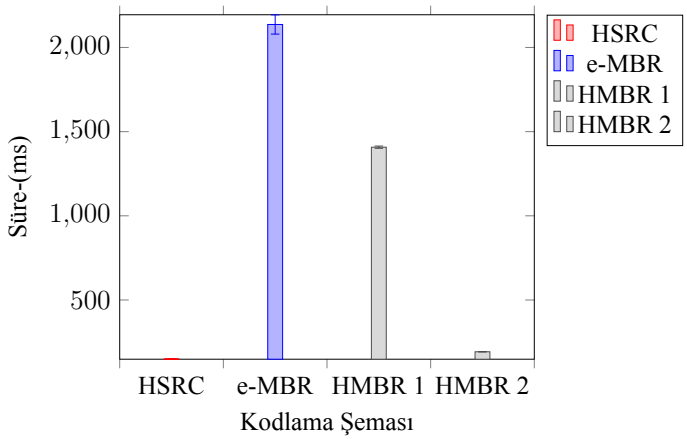
Bu alt bölümde HSRC (Oggier and Datta, 2011b), e-MBR (Rashmi et al., 2011b) ve HMBR (Haytaoglu and Dalkilic, 2013a) kodlama şemalarında düğüm tamiri işlemlerinin gerektirdiği toplam CPU süreleri incelenmiştir. Bunun için sırasıyla 10 MB, 100 MB ve 1 GB büyüklüklerindeki dosyalar ilgili kodlama şemaları kullanılarak kodlanmıştır. Bu kodlama işleminde $n = 15$, $k = 3$ ve $d = 4$ olarak alınmıştır.

Şekil 4.18, Şekil 4.19 ve Şekil 4.20'de sırasıyla orijinal veri büyüklükleri 10 MB, 100 MB ve 1 GB olan depolama sistemlerinde 100 KB büyüklüğünde tampon bellek kullanıldığında düğüm tamiri işlemleri için kullanılması gereken süreler görülmektedir. HSRC kodlama şeması tüm dosya büyüklükleri için düğüm tamiri işlemini en kısa zamanda gerçekleştirirken, e-MBR ve HMBR kodlama şemasının

birinci düğüm tamiri yöntemi birbirlerine yakın zamanlarda düğüm tamiri işlemini tamamlamaktadır. HSRC kodlama şemasının bir adaptasyonu olan HMBR kodlama şemasının ikinci düğüm tamiri yöntemi ise HSRC kodlama şemasının düğüm tamirinde kullandığı sürelere yakın, ancak HSRC kodlama şemasından daha çok ek veri depoladığı için daha yüksek süreler gerektirmektedir. HMBR kodlama şema-



Şekil 4.18. 10 MB dosya tamiri için kullanılan toplam CPU süreleri.

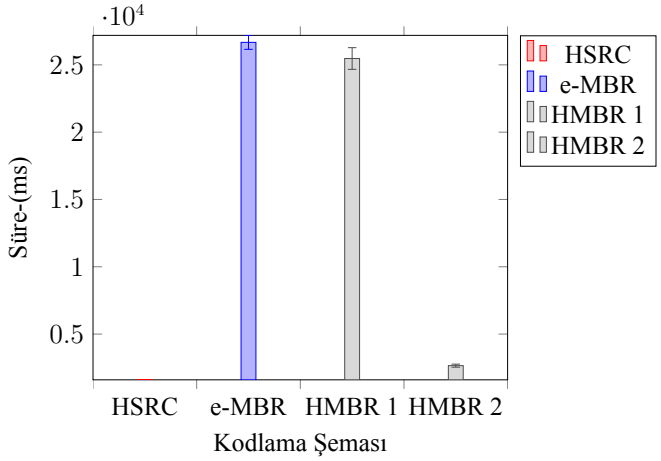


Şekil 4.19. 100 MB dosya tamiri için kullanılan toplam CPU süreleri.

sının ikinci düğüm tamiri yöntemi ise e-MBR kodlama şemasında kullanılan CPU süresinden da az süre kullanmaktadır.

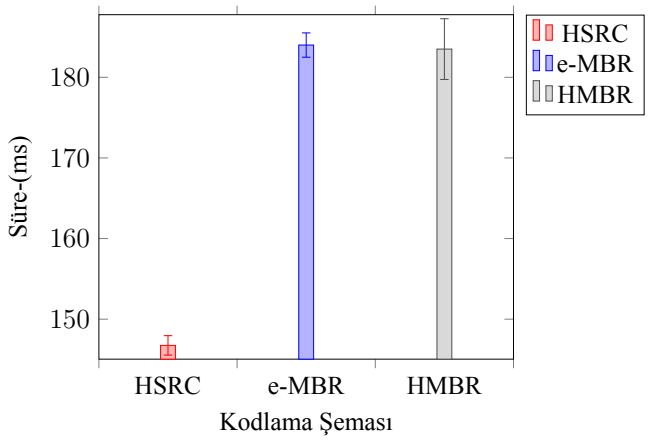
4.4.3 Kodlama şemalarında verinin geri çatımı işlemlerinin süreleri

Bu alt bölümde 10 MB, 100 MB ve 1 GB büyüklüklerindeki orijinal verilerin HSRC, e-MBR ve HMBR kodlama şemalarında geri çatılması için kullanılan CPU süreleri incelenmiştir. Kodlama şeması konfigürasyonu $n = 15$, $k = 3$ ve $d = 4$

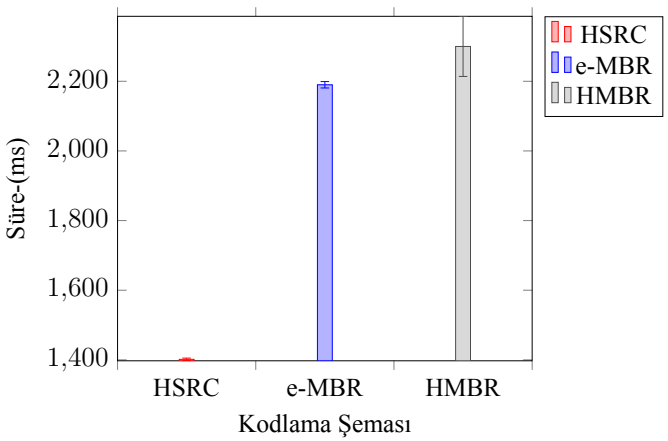


Şekil 4.20. 1 GB dosya tamiri için kullanılan toplam CPU süreleri.

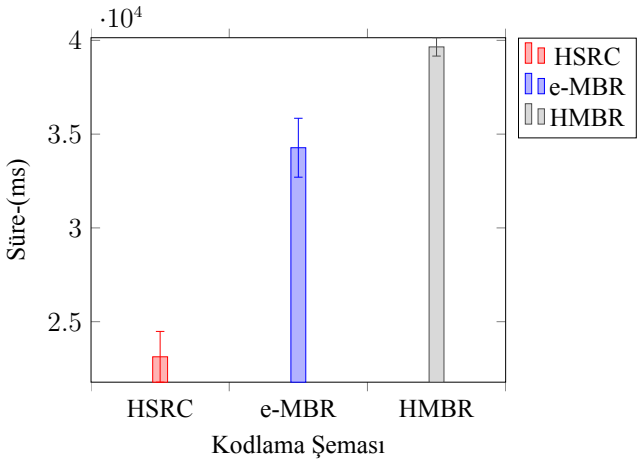
olacak şekilde ayarlanmıştır. Şekil 4.21, Şekil 4.22 ve Şekil 4.23'te ilgili kodama şemalarının verinin geri çatımı için kullandığı CPU süreleri gösterilmiştir. Buna göre tüm dosya büyüklükleri için HSRC kodlama şeması, verinin geri çatımı işlemi için en az süreyi kullanmaktadır. e-MBR ve HMBR kodlama şemaları verinin geri çatılması işleminde önceden kodladığı mesaj matrisi yapısından dolayı yüksek IO işlemi gerektirdiğinden bu alt bölümdeki sonuçlar da beklendiği çıkmıştır.



Şekil 4.21. 10 MB dosyanın geri çatımı için kullanılan toplam CPU süreleri.



Şekil 4.22. 100 MB dosyanın geri çatımı için kullanılan toplam CPU süreleri.



Şekil 4.23. 1 GB dosyanın geri çatımı için kullanılan toplam CPU süreleri.

4.4.4 Kümeleme tabanlı dağıtık depolama sistemlerinde düğüm tamiri

Bu alt bölümde kümeleme tabanlı dağıtık depolama sistemlerinde $n = 15$, $k = 3$ ve $d = 4$ parametreleri için 10 MB, 100 MB ve 1 GB büyüklüklerindeki dosyaların yedekli bir şekilde depolanması sonucunda bu sistemlerde düğüm tamiri işleminde kullanılan süreler ve mesaj sayıları incelenmiştir. Ölçülen sürenin içerisinde yerel işlemlerin yanı sıra ağ iletişimi için gereken süreler de bulunmaktadır. Ayrıca bir kümenin içerisinden diğer kümelerin paketleri de geçebildiğinden bir kümenin işlemi diğer bir kümenin işlem süresini etkilemektedir.

Her bir kümenin kullandığı kodlama şemasında yeni-gelen düğüm yeterli sayıda yardımcı düğümden veri parçalarını indirir. Bu parçaların her biri lime adı verilen daha küçük parçalara bölünmüştür. Yeni-gelen düğümün düğüm tamiri işlemine

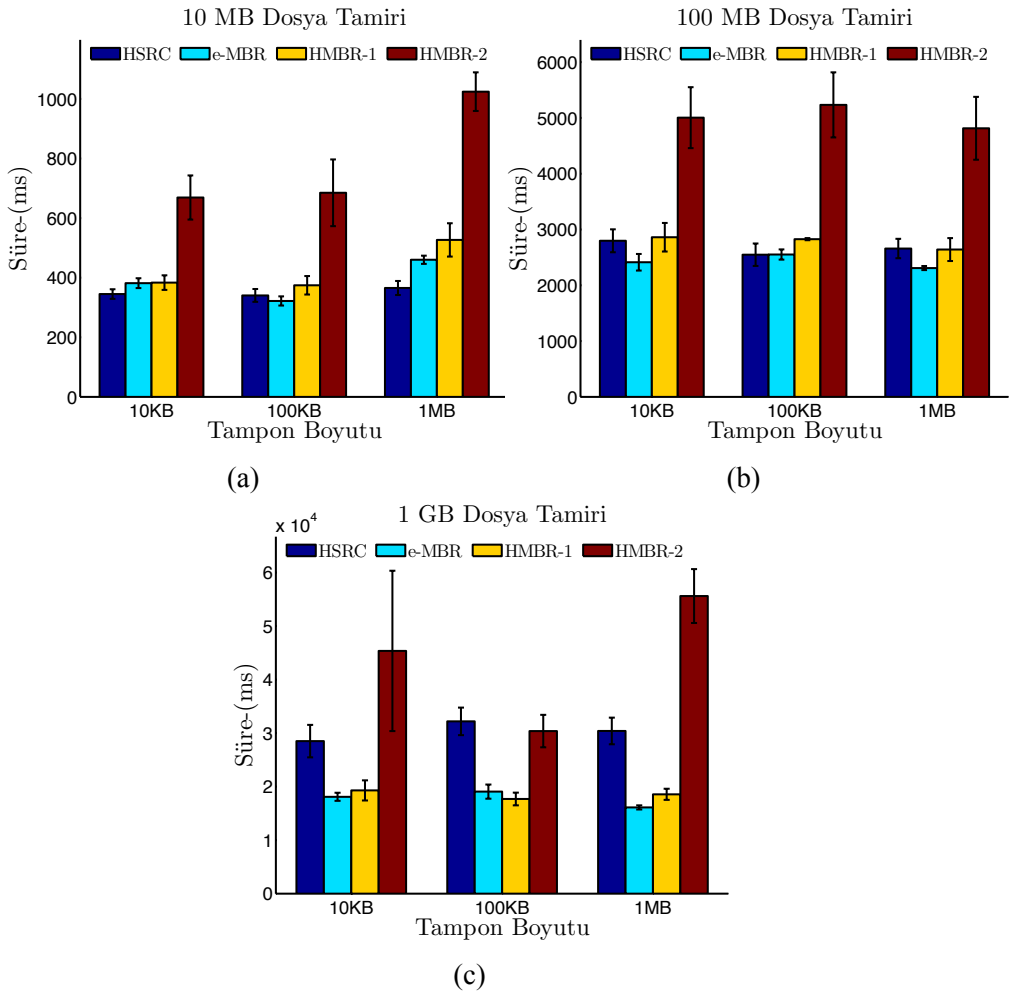
başlaması için tüm parçaları almayı beklemesine gerek yoktur. Bağlandığı tüm yardımcı düğümlerden aynı numaraya sahip limeleri alması düğüm tamiri işlemine başlaması için yeterlidir. Dolayısıyla, bir düğümün tamiri ağ iletişimi ile paralel olarak gerçekleştirilebilmektedir. Bir düğüm tamiri işleminin tamamlanması için gerekli olan süre aşağıdaki eşitlikte verilmiştir.

$$t_{tamir} = t_{ilet} + t_{lokal} \tag{4.56}$$

Eşitlik (4.56)'da t_{ilet} yeni-gelen düğümün düğüm tamirine başlaması ile son limeyi alması arasında geçen süredir. t_{lokal} ise son lime alındıktan sonra henüz çözülmemiş limelerin çözülmesi için gerekli olan CPU süresidir.

Benzetimde oluşturulan ağlar üç kategoriye bölünmüştü. İlk kategoriye giren düğümler HSRC kümesini, ikinci kategoriye giren düğümler e-MBR kümesini, üçüncü kategoriye giren düğümler ise HMBR kümesini oluşturmaktadır. Kümeleme tabanlı dağıtık depolama sisteminde üçüncü kümenin ağ trafiği açısından düşük maliyete sahip olduğu varsayıldığı için bu kümeye ikinci düğüm tamiri yöntemi ile yüksek mesaj maliyetine sahip olan HMBR kodlama şeması atanmıştır. Birinci kümede ise CPU kullanım süresinin maliyetli olduğu varsayıldığı için HSRC kodlama şeması atanmıştır. İkinci kümeye de e-MBR kodlama şeması atanmıştır.

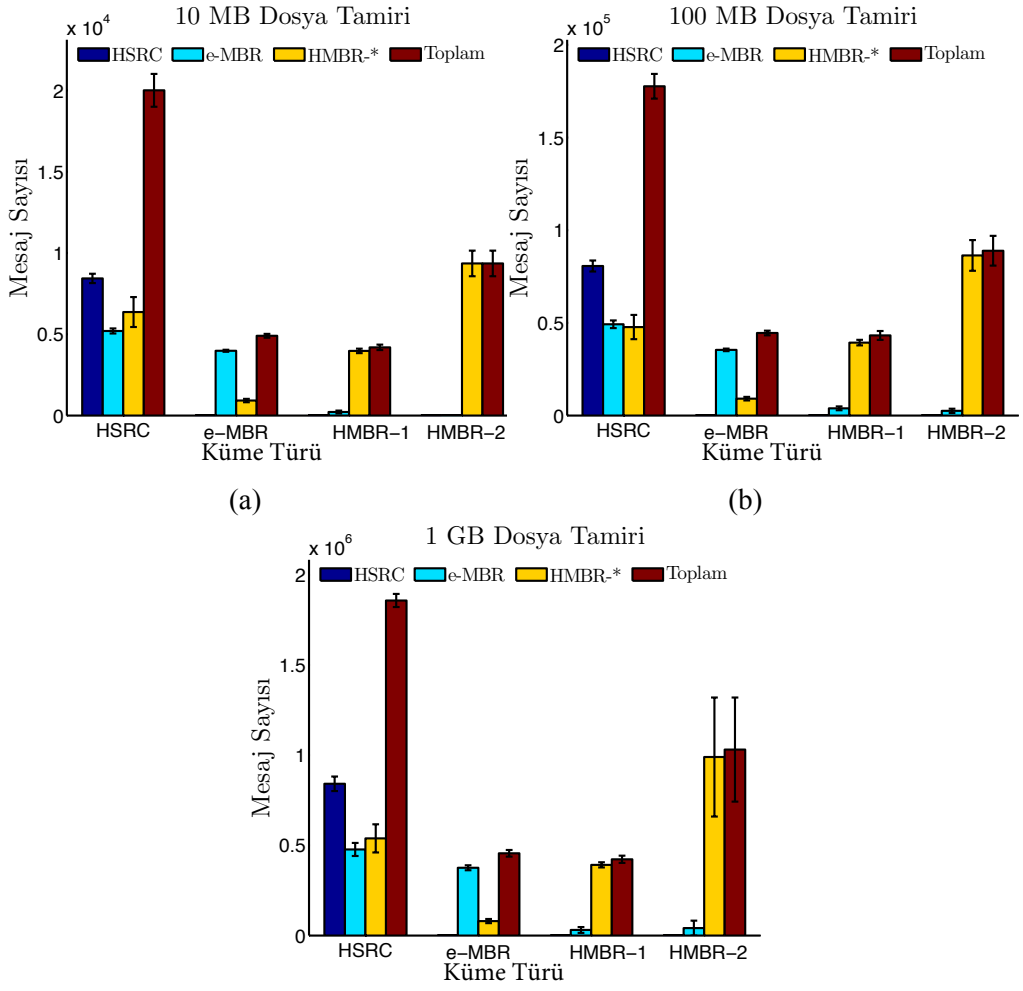
Şekil 4.24 (a), Şekil 4.24 (b) ve Şekil 4.24 (c)'de sırasıyla 10 MB, 100 MB ve 1GB büyüklüklerindeki dosyaların yedekli bir şekilde depolandığı sistemlerde düğüm tamiri işlemlerinde kümeler tarafından kullanılan süreler görülmektedir. Düğüm tamiri işlemleri çeşitli tampon bellek büyüklükleri kullanılarak gerçekleştirilmiştir. Bu büyüklükler bir seferde bir dosyadan okunan veri miktarını belirlemektedir. Genel olarak tüm benzetimlerde HMBR kodlama şemasının ikinci düğüm tamiri yöntemi ($ASRC_{NR}$) en yüksek süreyi gerektirmektedir. Bunun nedeni ise bu yöntemde yeni-gelen düğümün diğer düğüm tamiri yöntemlerine göre daha çok veri indirmesi gerekmesidir. HMBR kümesinin, HMBR kodlama şemasının birinci düğüm tamiri yöntemini ($AMBR_{NR}$) kullandığı durumlarda düğüm tamiri için harcanan süre ile e-MBR kümesinde harcanan süre birbirine yakındır. HSRC kümesinde ise 100 MB ve 1GB büyüklüklerindeki dosyaların kodlandığı sistemlerde düğüm tamiri için genellikle e-MBR ve HMBR kodlama şemasının birinci düğüm tamiri yönteminden daha yüksek süre kullanılmaktadır. 10 MB dosyanın yedekli olarak depolandığı bir sistem için bu durum değişmektedir. 10 MB dosyanın depolandığı sistemlerde asıl süreyi, HSRC, e-MBR ve HMBR kodlama şemasının birinci yöntemi için, ağ işlemleri yerine yerel işlemler belirlediğinden HSRC kümesi bu durumda düğüm tamiri için düşük süreler kullanmaktadır. Burada 10 MB büyüklüğünde bir dosyanın depolandığı sistemde 1 MB büyüklüğündeki tampon boyutu süre açısından HMBR



Şekil 4.24. Kümeleme tabanlı dağıtık depolama sisteminde düğüm tamiri için geçen süre a) 10 MB dosya tamiri için kullanılan süre, b) 100 MB dosya tamiri için kullanılan süre, c) 1 GB dosya tamiri için kullanılan süre.

kümesi için verimsiz olmuştur.

Şekil 4.25'de düğüm tamiri işlemi için kümelerde kullanılan mesaj sayıları görülmektedir. Genel olarak en düşük mesaj kullanımı e-MBR ve HMBR-1 kümeleri içerisinde gerçekleştirilmiştir. HMBR-1 ve HMBR-2 sırasıyla HMBR kümelerinde birinci ve ikinci düğüm tamiri yöntemi kullanıldığı durumları ifade etmektedir. HMBR-* ise düğüm tamiri yöntemi ayrımı yapılmaksızın HMBR kümesi içerisinden geçen düğüm tamiri yöntemlerince kullanılan tüm mesajların sayısını ifade etmektedir.



Şekil 4.25. Kümeleme tabanlı dağıtık depolama sisteminde düğüm tamiri için kullanılan mesaj sayıları a) 10 MB dosya tamiri için kullanılan mesaj sayısı, b) 100 MB dosya tamiri için kullanılan mesaj sayısı, c) 1 GB dosya tamiri için kullanılan mesaj sayısı.

Mesaj sayısı parametresi üzerinde tampon boyutunun bir etkisi bulunmamaktadır. Bu nedenle farklı tampon boyutları ve mesaj sayısının ilişkilendirildiği bir analiz bulunmamaktadır. HSRC kümesinde genel olarak en yüksek mesaj kullanımı olmasına rağmen bu mesajların büyük kısmını diğer kümelerden gelen mesajlar oluşturmaktadır.

4.4.5 Kümeleme tabanlı dağıtık depolama sistemlerinde verinin geri çatımı

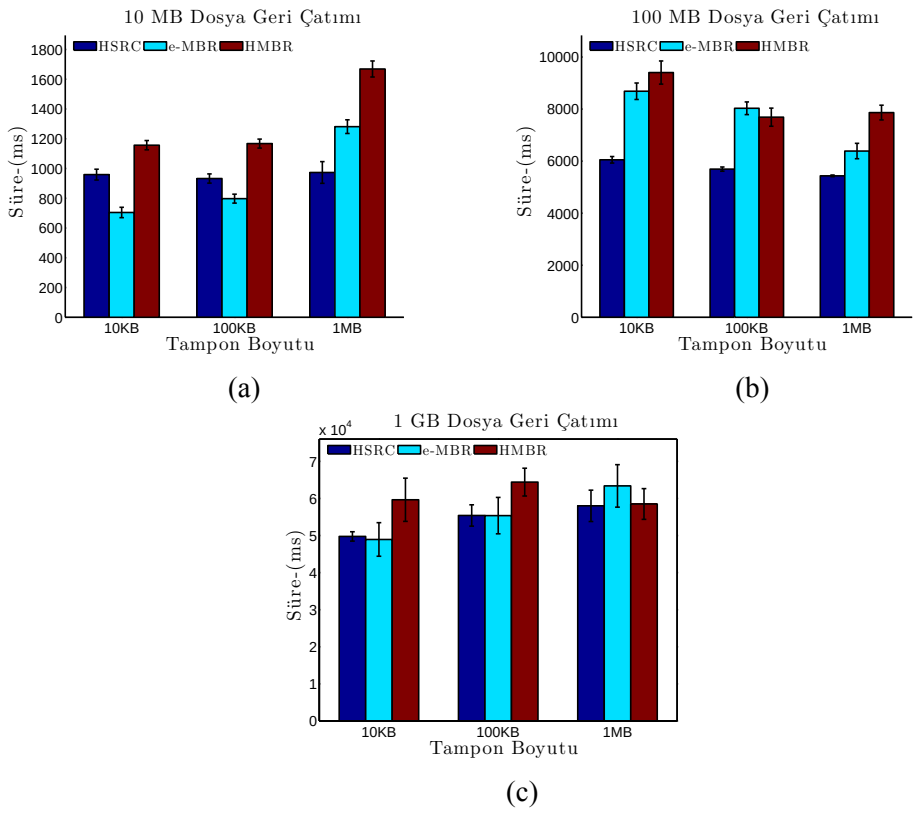
Bu alt bölümde $n = 15$, $k = 3$ ve $d = 4$ parametreleri için kümeleme tabanlı dağıtık depolama sisteminde verinin geri çatımı için kullanılan süreler ve mesaj sa-

yıları görülmektedir (Şekil 4.26 (a), Şekil 4.26 (b) ve Şekil 4.26 (c)). Ölçülen sürenin içerisinde yerel işlemlerin yanı sıra ağ iletişimi için gereken süreler de bulunmaktadır. Ayrıca bir kümenin içerisinden diğer kümenin paketleri de geçebildiğinden kümelerin kendi işlemleri birbirlerinin işlem sürelerini etkilemektedir. Kümelerde gerçekleştirilen verinin geri çatımı işlemi süreleri aşağıdaki eşitlikle hesaplanmaktadır.

$$t_{geri} = t_{ilet} + t_{lokal} \tag{4.57}$$

Eşitlik (4.57)'de t_{ilet} verinin geri çatımı işleminin başlatılması ve veri-toplayıcı düğümün son aldığı lime arasında geçen süreyi, t_{lokal} ise son alınan limeden itibaren yerel olarak gerçekleştirilmesi gereken geri çatım işlemleri için geçen süredir.

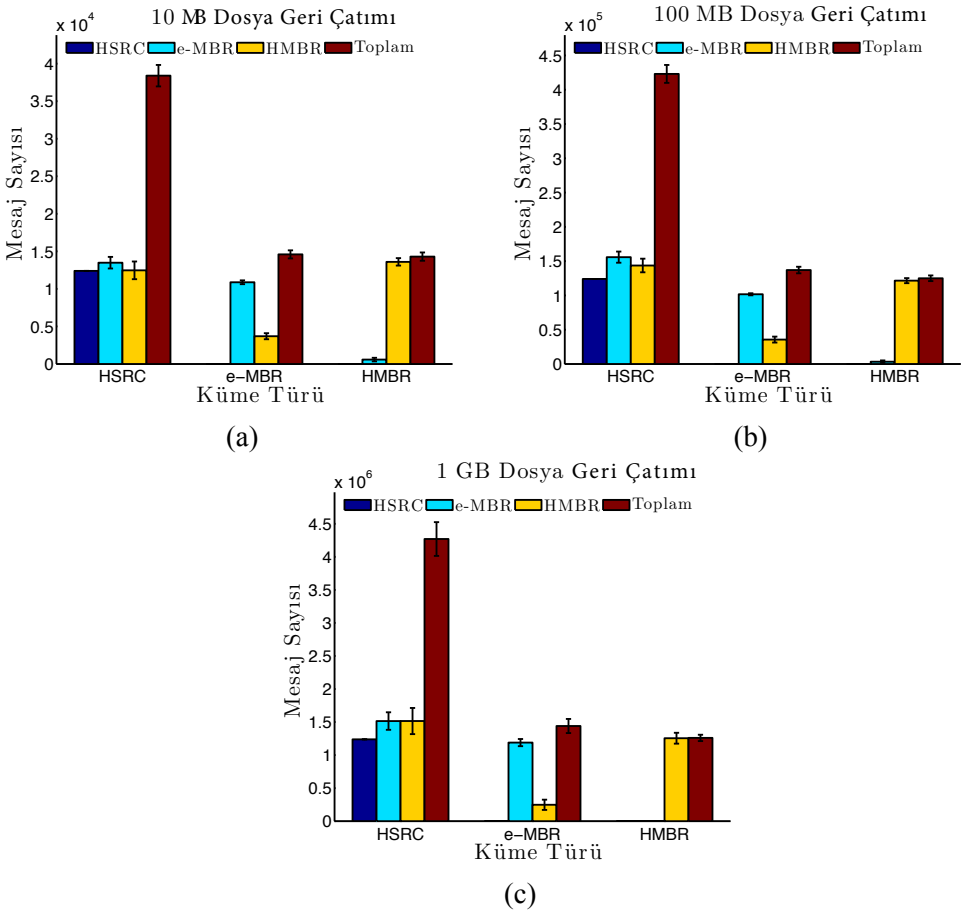
HSRC kümesi 100 MB ve 1GB büyüklüğündeki dosyalar için genel olarak diğer kümelere göre verinin geri çatımı işlemi için daha az süre kullanmaktadır. HSRC kodlama şeması verinin geri çatımı işlemi için diğer kodlama şemalarına göre daha az işlemci süresi ve bant genişliği gerektirmesine rağmen, HSRC kümesinde verinin geri çatımı işleminin harcadığı sürenin diğer kümelerde verinin geri çatımı işlemi süresine yakın olmasının nedeni HSRC kümesinin içerisinden diğer kümelerin paketlerinin geçişinin olmasıdır. HMBR kümesi ise verinin geri çatımı işlemi için genel olarak en yüksek süre kullanımına sahiptir.



Şekil 4.26. Kümeleme tabanlı dağıtık depolama sisteminde verinin geri çatılması için kümelerde kullanılan süre a) 10 MB dosyanın geri çatılması için kullanılan süre, b) 100 MB dosyanın geri çatılması için kullanılan süre, c) 1 GB dosyanın geri çatılması için kullanılan süre.

10 KB tampon boyutu 10 MB ve 1 GB dosyanın geri çatılmasında geçen süreler açısından daha verimli olmuştur. 100 MB dosyanın geri çatımı için ise 1 MB tampon boyutunda kullanılan süreler daha düşük olmaktadır.

Şekil 4.27 (a), Şekil 4.27 (b) ve Şekil 4.27 (c)'de $n = 15$, $k = 3$ ve $d = 4$ parametreleri için HSRC, e-MBR ve HMBR kümelerinde sırasıyla 10 MB, 100 MB ve 1 GB büyüklüklerindeki dosyaların geri çatımı için kullanılan mesaj sayıları görülmektedir. En yüksek mesaj harcaması HSRC kümesinde olup, bunun nedeni bu kümenin içerisinden diğer kümelerin geri çatım işlemleri için kullanılan mesajların geçmesidir. Bunun dışında toplam mesaj sayısı, e-MBR ve HMBR kümelerinde birbirlerine yakındır. Kodlama şeması bazında ise ağdaki en yüksek mesajı, verinin geri çatımında belirli özelliklere sahip $k = 3$ düğüme bağlanılması gereken HMBR kodlama şeması kullanmıştır.



Şekil 4.27. Kümeleme tabanlı dağıtık depolama sisteminde verinin geri çatılması için kümelerde kullanılan mesaj sayısı a) 10 MB dosyanın geri çatılması için kullanılan mesaj sayısı, b) 100 MB dosyanın geri çatılması için kullanılan mesaj sayısı, c) 1 GB dosyanın geri çatılması için kullanılan mesaj sayısı.

4.4.6 Kümeleme tabanlı olmayan dağıtık depolama sistemlerinde düğüm tamiri

Bu alt bölümde kümeleme tabanlı olmayan dağıtık depolama sistemlerinde düğüm tamiri işlemlerinin harcadığı süre ve mesaj sayısı kümeleme tabanlı dağıtık depolama sistemleri ile karşılaştırılmıştır. Kümeleme tabanlı olmayan dağıtık depolama sistemlerinde ağdaki düğümler rastgele bir şekilde üç farklı gruba bölünmüştür. Sistemdeki her grup aynı kodlama şemasını kullanmaktadır. Dolayısıyla, üç farklı kümeleme tabanlı olmayan dağıtık depolama sistemi incelenmiştir. Bunlar; üç farklı grubun HSRC kodlama şemasını kullandığı, üç farklı grubun e-MBR kodlama şemasını kullandığı ve üç farklı grubun da HMBR kodlama şemasını kullandığı sistemlerdir. Bu sistemlerde ve kümeleme tabanlı dağıtık depolama sisteminde 100 KB tampon boyutu için düğüm tamirinde kullanılan süreler ve mesaj sayıları gösterilmiştir. Ayrıca, grup ve küme başına düşen süre ve mesaj kullanımları incelenmiştir. Kodlama şeması parametreleri olarak yine $n = 15$, $k = 3$ ve $d = 4$ kullanılmıştır.

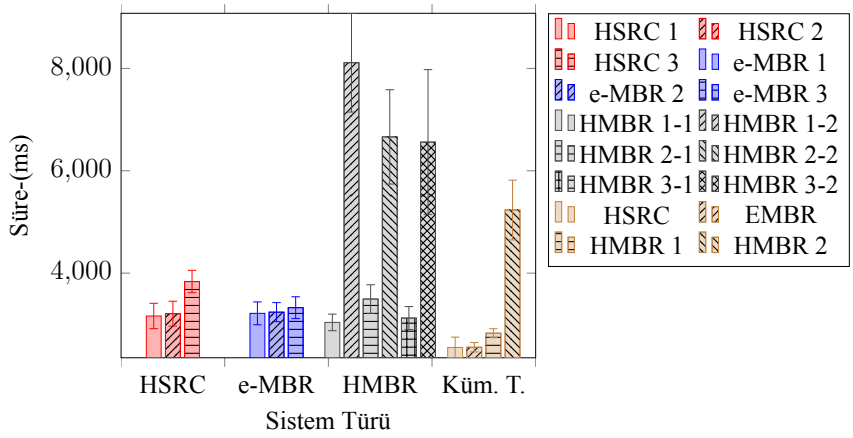
10 MB ve 1GB büyüklüğünde bir dosyanın yedekli depolandığı sistemler hakkında elde edilen sonuçlar Ek 1'de verilmiştir.

Ek 1.1'de 10 MB büyüklüğünde dosyaların depolandığı sistemlerde düğüm tamiri için kullanılan süreler ölçülmüştür. Şekil açıklamalarında bulunan HMBR 1-1, HMBR kodlama şeması kullanan kümeleme tabanlı olmayan sistemde birinci grubun birinci düğüm tamiri yöntemini kullandığı durumu temsil etmektedir. Benzer şekilde HMBR 3-2 ise sadece HMBR kodlama şeması kullanılan bir sistemde üçüncü grubun ikinci düğüm tamiri yöntemini kullandığı durumu temsil etmektedir. HSRC kodlama şemasını kullanan sistemin diğer sistemlere göre daha az süre kullandığı görülmektedir. HMBR kodlama şemasını kullanan sistemde ise ikinci düğüm tamiri yöntemi kullanıldığında diğer sistemlere göre daha yüksek süreler kullandığı görülmektedir. Benzer şekilde kümeleme tabanlı dağıtık depolama sisteminde de HMBR kümesinin ikinci yönteminin diğer yöntemlere göre çok yüksek süre kullanarak düğüm tamirini gerçekleştirmesi dikkat çekmektedir. Ek 1.2'de ise dört farklı sistemde 10 MB büyüklüğünde bir dosyanın kodlandığı durumda düğüm tamiri için grup/küme başına kullanılması gereken ortalama süreler gösterilmektedir. Ortalamalara göre en düşük süreyi kümeleme tabanlı dağıtık depolama sistemi kullanmaktadır. Kümeleme tabanlı dağıtık depolama sistemini sırasıyla HSRC, e-MBR ve HMBR kodlama şemalarının kullanıldığı kümeleme tabanlı olmayan sistemler takip etmektedir.

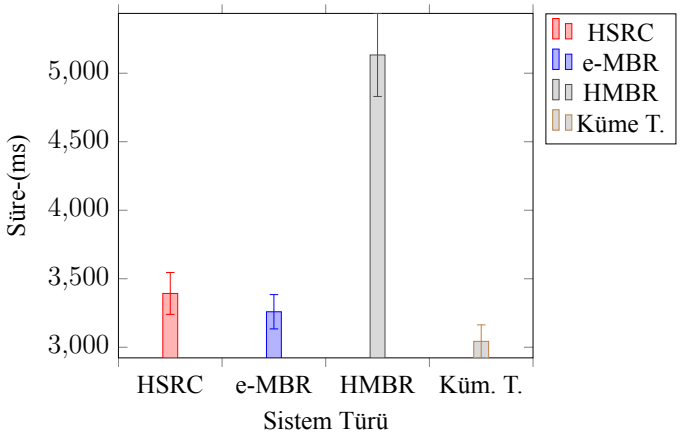
Şekil 4.28'de 100 MB büyüklüğünde dosyaların depolandığı sistemlerde düğüm tamiri işlemlerinin gerektirdiği süreler görülmektedir. Buna göre, kümeleme tabanlı ve e-MBR kodlama şemasını kullanan sistem en düşük süreleri gerektirirken HMBR kodlama şemasını kullanan depolama sistemi ikinci düğüm tamiri yöntemi nedeniyle diğerlerine kıyasla yüksek süreler gerektirmektedir. Şekil 4.29'da ise 100 MB büyüklüğünde bir dosyanın depolandığı bir sistemde düğüm tamiri için bir grup/küme'de kullanılması gerekli olan süreler görülmektedir. En düşük süreyi kümeleme tabanlı dağıtık depolama sistemi kullanmaktadır.

Ek 1.3 ve Ek 1.4'te 1GB büyüklüğünde bir dosyanın yedekli bir şekilde depolandığı sistemlerde düğüm tamiri için grup ve kümelerde kullanılan süreler ve bunların ortalamaları verilmiştir. 100 MB büyüklüğündeki dosyaların depolandığı sistemlere benzer şekilde ortalamada en düşük süreyi kümeleme tabanlı dağıtık depolama sistemi kullanmış olup ardından e-MBR, HSRC ve HMBR kodlama şemalarını kullanan sistemler gelmektedir.

Ek 1.5'de depolama sistemlerinin 10 MB büyüklüğünde bir dosyanın kodlan-



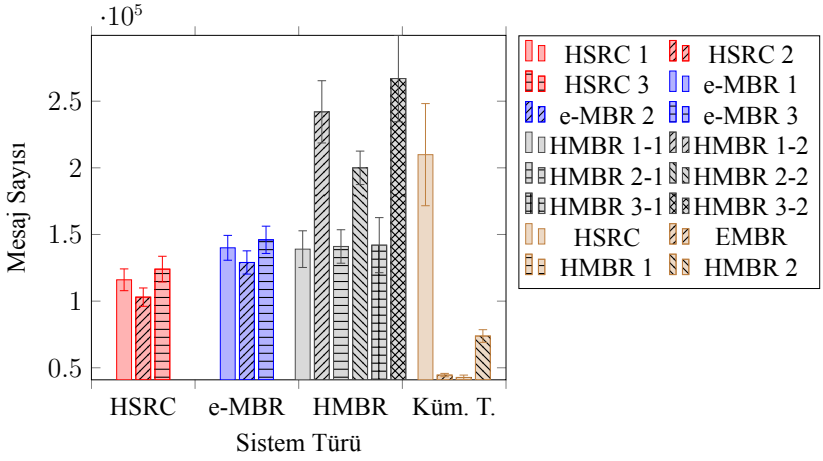
Şekil 4.28. 100MB dosya tamiri için geçen süreler.



Şekil 4.29. 100 MB dosya tamiri için geçen sürelerin ortalamaları.

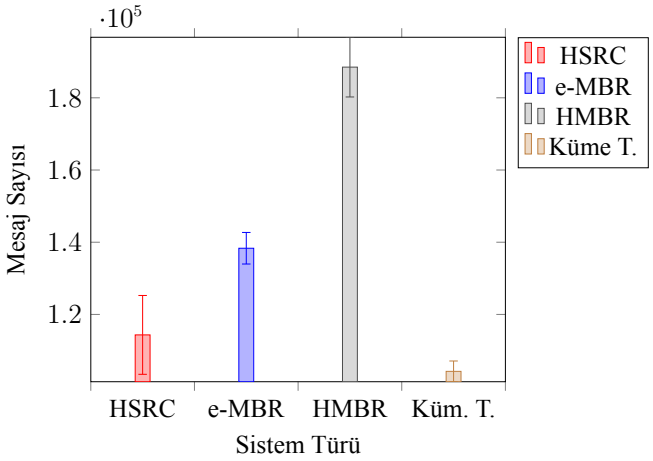
mış parçasının tamir edilmesi sırasında kullandığı mesaj sayıları verilmiştir. Burada, gruplar ve kümeler diğer grupların veya kümelerin de mesajlarını kendi üzerlerinden geçirdiği için tek bir grubun ya da kümenin kullandığı mesaj sayısı sadece kendi düğüm tamiri ile ilgili mesajları kapsamamaktadır. Örneğin kümeleme tabanlı dağıtık depolama sisteminde HSRC kümesinin içerisinden çok sayıda e-MBR ve HMBR kümesinin mesajları da geçmektedir. Ek 1.6'da depolama sistemlerinin yedekli bir şekilde depolanan 10 MB'lık bir dosyanın parçasının tamiri için kullandıkları mesaj sayılarının ortalamaları görülmektedir. Buna göre en düşük mesaj sayısı kümeleme tabanlı sistemde oluşurken, kümeleme tabanlı olmayan sistemler arasında en düşük mesaj sayısı HSRC grubuna aittir. 10 MB büyüklüğündeki dosyanın kodlanmış parçasının tamiri için kullanılan mesaj sayısı hakkında elde edilen sonuçlar süre için elde edilen sonuçlarla doğru orantılıdır.

Şekil 4.30'da 100 MB büyüklüğünde bir dosyanın depolandığı sistemlerde dosya tamiri için kullanılan mesaj sayıları görülmektedir. Şekil 4.31'de 100 MB'lık



Şekil 4.30. 100 MB dosya tamiri için kullanılan mesaj sayısı.

bir dosyanın kodlanmış parçasının tamir edilmesi için sistemlerde kullanılan ortalama mesaj sayısı görülmektedir. e-MBR ve HMBR kodlarında düğüm tamiri için bağlanılan düğümlerin sayısı daha çok olduğu için mesajların geçtiği hat sayısı da yüksek olmaktadır. Kümeleme tabanlı dağıtık depolama sistemi mesaj sayısı kullanımı açısından ortalamada en düşük değere sahip olmasına karşın küme bazında bakılacak olursa HSRC kümesinde oldukça yüksek mesaj kullanımına sahiptir.



Şekil 4.31. 100 MB dosya tamiri için kullanılan mesajların ortalamaları.

Ek 1.7'de 1 GB büyüklüğünde bir dosyanın yedekli olarak depolandığı sistemlerde düğüm tamiri için kullanılan mesaj sayıları görülmektedir. Diğer sonuçlara benzer şekilde en yüksek mesaj sayıları sadece HMBR kodlama şemasının kullanıldığı sistemde HMBR kodlama şemasının ikinci yönteminin kullanıldığı durumlarda ortaya çıkmıştır. Ek 1.8'de 1 GB dosyaların yedekli bir şekilde depolandığı sistemlerde düğüm tamiri için grup ve küme başına kullanılan mesaj sayılarının or-

talamaları verilmiştir. Bu sefer kümeleme tabanlı dağıtık depolama sistemi, HSRC kodlama şeması kullanılan sisteme göre biraz daha yüksek mesaj kullanımı gerçekleştirmektedir. Bunun nedeni kümeleme tabanlı dağıtık depolama sisteminde indirilen parçaların büyüklüğüne bağlı olarak daha yüksek bant genişliği kullanan TCP bağlantılarının ağ tıkanıklığı oluşturmasıdır.

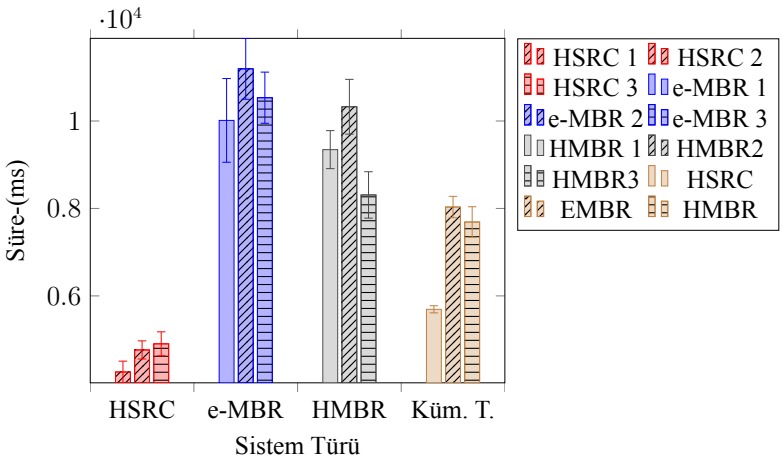
4.4.7 Kümeleme tabanlı olmayan dağıtık depolama sistemlerinde verinin geri çatımı

Bu alt bölümde $n = 15$, $k = 3$ ve $d = 4$ parametreleri için 10 MB, 100 MB ve 1 GB büyüklüğündeki dosyaların yedekli bir şekilde depolandığı sistemlerde 100 KB tampon boyutunda verinin geri çatımı işlemi için kullanılan süre ve mesaj sayıları incelenmiştir.

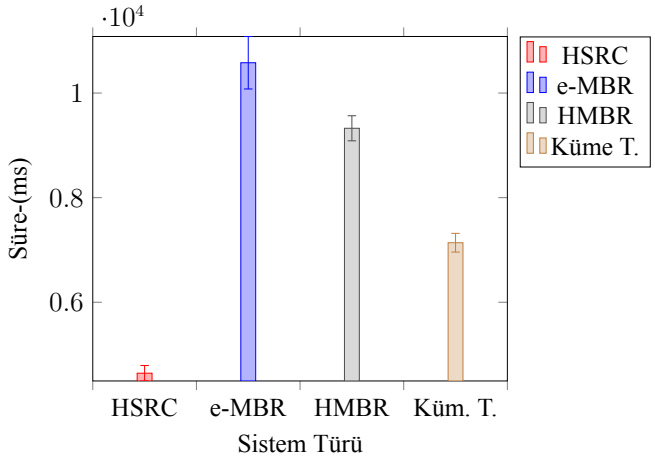
Ek 1.9'da 10 MB büyüklüğündeki bir dosyanın kümeleme tabanlı olan ve olmayan sistemlerde geri çatılması için kullanılan süreler görülmektedir. Sadece e-MBR ve HMBR kodlama şemalarının kullanıldığı sistemlerde kullanılan süreler birbirlerine çok yakın olup en düşük süreleri kümeleme tabanlı dağıtık depolama sistemi kullanmaktadır. HSRC kodlama şemasının kullanıldığı gruplar verinin geri çatılması işlemi için HMBR ve e-MBR kodlama şemalarının kullanıldığı gruplarla aynı sayıda düğümden daha düşük büyüklüklerde kodlanmış veri indirdiği için bu grupların kullandığı süre de daha az olmaktadır. Ek 1.10'da 10 MB dosyanın geri çatımı için grup veya küme başına kullanılan ortalama süreler görülmektedir. Kümeleme tabanlı dağıtık depolama sistemi bu işlem için ortalamada en düşük süreyi kullanmasına rağmen kendi kümeleri tarafından kullanılan süreler arasında farklılıklar bulunmaktadır.

Şekil 4.32'de 100 MB büyüklüğündeki bir dosyanın geri çatımı için kümeleme tabanlı olan ve olmayan sistemlerde kullanılan süreler görülmektedir. Burada ise en düşük süreler HSRC grupları tarafından kullanılmıştır. Şekil 4.33'te ise 100 MB'lık bir dosyanın geri çatılması için grup ve küme başına düşen süre harcamaları görülmektedir. Sadece HSRC kodlama şemasının kullanıldığı sistemler 100 MB büyüklüğünde bir dosyanın geri çatılması işlemi için diğer sistemlere göre çok daha düşük süreler kullanmaktadır.

Ek 1.11'de 1 GB büyüklüğünde bir dosyanın geri çatımı için kümeleme tabanlı olan ve olmayan sistemlerde kullanılan süreler görülmektedir. Ek 1.12'de ise 1 GB büyüklüğünde bir dosyanın geri çatımı için grup ve küme başına düşen süreler görülmektedir. 10 MB'lık dosya dışında test edilen diğer tüm dosya büyüklüklerinde



Şekil 4.32. 100 MB dosyanın geri çatımı için kullanılan süreler.



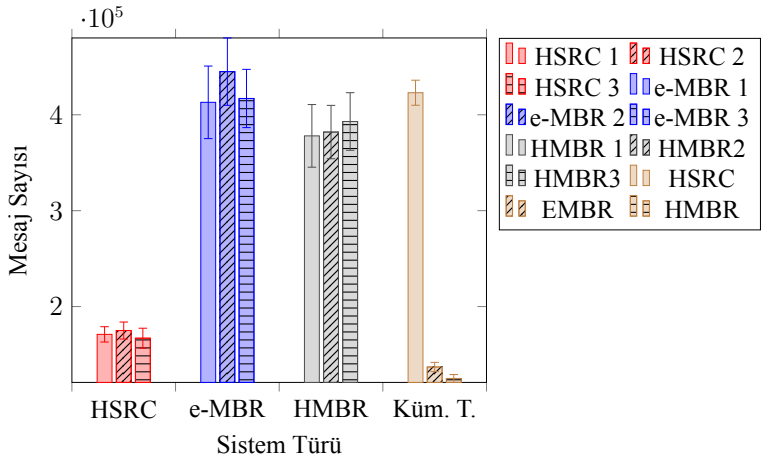
Şekil 4.33. 100 MB dosyanın geri çatımı için kullanılan sürelerin ortalaması.

verinin geri çatımı için en az süre HSRC kodlama şemasının kullanıldığı kümeleme tabanlı olmayan dağıtık depolama sistemi tarafından kullanılmıştır.

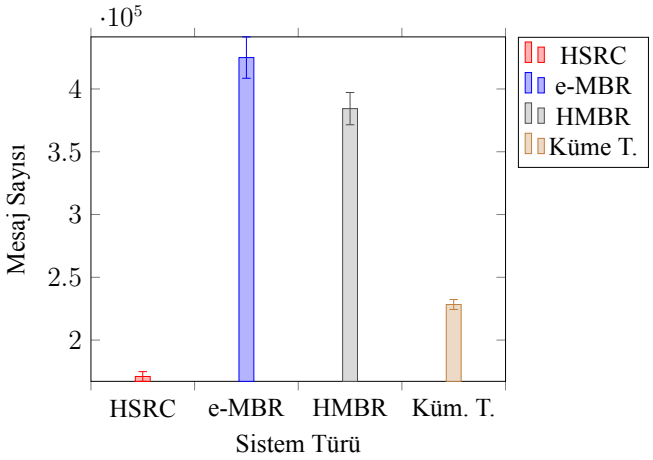
Ek 1.13'te 10 MB'lık bir dosyanın geri çatımı için sistemler tarafından kullanılan mesaj sayıları görülmektedir. HSRC kodlama şemasının diğer kodlama şemalarıyla aynı sayıda düğüme bağlanıp daha az veri indirmesi gerektiği için HSRC kodlama şeması kullanılan kümeleme tabanlı olmayan dağıtık depolama sisteminde daha az mesaj kullanımı gerçekleşmiştir. Ek 1.14'te 10 MB büyüklüğünde bir dosyanın geri çatılması için gruplar ve kümeler tarafından kullanılan mesaj sayılarının ortalamaları bulunmaktadır. En düşük mesaj HSRC kodlama şeması kullanılan dağıtık depolama sistemi tarafından kullanılırken ardından sırasıyla kümeleme tabanlı, e-MBR ve HMBR kodlama şemasının kullanıldığı sistemler gelmektedir.

Şekil 4.34'te ve Şekil 4.35'de sırasıyla 100 MB büyüklüğünde bir dosyanın geri çatımı için kullanılan mesajlar ve bu mesajların grup ve küme başına düşen or-

talamaları görülmektedir. Sonuçlar 10 MB'lık dosya geri çatımı işlemine benzerdir.



Şekil 4.34. 100 MB dosyanın geri çatımı için kullanılan mesajlar.



Şekil 4.35. 100 MB dosyanın geri çatımı için kullanılan mesajların ortalaması.

Ek 1.15 ve Ek 16'da 1 GB büyüklüğünde bir dosyanın geri çatımı için kullanılan mesaj sayıları ve bunların ortalamaları bulunmaktadır. Önceki sonuçlara benzer şekilde HSRC kodlama şeması kulanılan sistem en düşük sayıda mesaj kullanmakta, bunun ardından kümeleme tabanlı dağıtık depolama sisteminin kullandığı mesaj sayısı gelmektedir. Sadece e-MBR ve HMBR kodlama şemalarının kullanıldığı kümeleme tabanlı olmayan dağıtık depolama sistemleri ise birbirlerine yakın sayıda mesaj kullanımı gerçekleştirmiştir.

Verinin geri çatılması işlemi için genel olarak HSRC kodlama şemasının kul-

lanıldığı kümeleme tabanlı olmayan dağıtık depolama sistemleri en iyi süre ve mesaj kullanımı göstermiştir. Ancak bu kodlama şemasında verinin geri çatılmasının herhangi k düğüm ile gerçekleştirilememesi bu kodlama şemasının e-MBR kodlama şemasına göre bir dezavantajıdır.

4.4.8 Kümeleme tabanlı olan ve olmayan dağıtık depolama sistemlerinin maliyet analizi

Bu alt bölümde kümeleme tabanlı olan ve olmayan dağıtık depolama sistemlerinin maliyet analizi gerçekleştirilmiştir. Maliyet analizinde, maliyet faktörleri olarak ağ trafiği, toplam süre ve CPU kullanım süreleri hesaba katılmıştır. Maliyet hesaplamaları düğüm tamiri ve verinin geri çatımı işlemleri için ayrı ayrı gerçekleştirilmiştir.

Maliyet fonksiyonlarında kümeler ve gruplar için tanımlanmış maliyet faktörleri bulunmaktadır. Bu maliyet faktörleri temel olarak işlemlerin aldığı toplam süre, ağ trafiği ve işlemlerin kullandığı CPU süresi kaynakları için belirlenmiştir. Grupların maliyet parametrelerinin değerleri kümeler için atanan maliyet parametrelerinin ortalaması şeklindedir. $t_1,\ t_2$ ve t_3 sırasıyla birinci, ikinci ve üçüncü kümelerin, harcanan toplam süre için belirlenmiş maliyet değerleridir. Ayrıca, HMBR kodlama şemasının kullanıldığı üçüncü kümede toplam zaman maliyeti için an içinde değişen iki farklı zaman maliyeti değeri bulunmaktadır ($t_{3,1},\ t_{3,2}$). Bu değerlerin ortalaması t_3'ü vermektedir. t_{ort} ise grupların her biri için kullanılan zaman parametresi maliyeti değeridir ve gruplar rastgele seçildiği için bu değer t_1, t_2 ve t_3'ün ortalaması şeklindedir.

Sadece HMBR kodlama şemasının kullanıldığı kümeleme tabanlı olmayan dağıtık depolama sistemi için birinci grupta $t_{g1,1}$, $t_{g1,2}$, ikinci grupta $t_{g2,1}$, $t_{g2,2}$ ve üçüncü grupta ise $t_{g3,1}$ ve $t_{g3,2}$ olmak üzere her bir grup için farklı zamanlarda farklı zaman maliyetleri bulunmaktadır. Ancak her grubun ortalama zaman maliyeti değeri t_{ort} değerine eşittir.

β_1, β_2 ve β_3 ise kümeleme tabanlı dağıtık depolama sisteminde kümelerin içerisinden geçen ağ trafiği için belirlenen maliyet parametreleridir. β_{ort} ise kümeler için belirlenmiş olan $\beta_1,\ \beta_2$ ve β_3 maliyet parametrelerinin bir ortalamasıdır. Bu maliyet faktörü simgesi, kümeleme tabanlı olmayan dağıtık depolama sistemlerinde bant genişliği maliyet parametresinin her bir grup için belirlenmiş değerini temsil etmektedir. $\beta_{3,1}$ ve $\beta_{3,2}$ ise kümeleme tabanlı dağıtık depolama sisteminde üçüncü kümenin zaman içerisinde değişen ağ trafiği yükünün maliyet parametresinin aldığı

değerlerdir. Zaman faktöründe olduğu gibi $\beta_{3,1}$ ve $\beta_{3,2}$ değerlerinin ortalaması β_3'ü vermektedir.

$\beta_{g1,1}$, $\beta_{g1,2}$ ise sadece HMBR kodlama şemasının kullanıldığı kümeleme tabanlı olmayan depolama sistemlerinde ilk grupta zaman içerisinde değişen ağ trafiği yükü maliyet faktörünün aldığı değerlerdir. $\beta_{g2,1}$, $\beta_{g2,2}$, $\beta_{g3,1}$ ve $\beta_{g3,2}$ benzer şekilde bu sistemdeki ağ trafiği yükü maliyet parametresinin ilgili gruplar için zaman içerisinde aldığı değerlerdir. Her gruba ait ağ trafiği maliyet faktörlerinin ortalaması β_{ort} değerine eşittir.

t_{cpu1}, t_{cpu2} ve t_{cpu3} kümeleme tabanlı dağıtık depolama sisteminde kümeler için belirlenmiş CPU işlem süresinin maliyet değerleridir. Rastgele bir şekilde gruplara ayrılan kümeleme tabanlı olmayan dağıtık depolama sistemlerinde her bir grup için CPU işlemi maliyeti olarak $t_{cpu_{ort}}$ değeri atanmış olup, bu değer t_{cpu1}, t_{cpu2} ve t_{cpu3} değerlerinin ortalamasıdır. Diğer maliyet faktörleri için olduğu gibi sadece HMBR kodlama şemasını kullanan kümeleme tabanlı olmayan dağıtık depolama sistemlerinde CPU işlem süresinin zaman içerisindeki önemi değişmektedir. Bu değişim gruplar için $t_{gcpu1,1}$,$t_{gcpu1,2}$, $t_{gcpu2,1}$, $t_{gcpu2,2}$, $t_{gcpu3,1}$ ve $t_{gcpu3,2}$ olarak belirlenmiş olup gruplara ait maliyet faktörlerinin ortalaması $t_{cpu_{ort}}$'u vermektedir.

Çizelge 4.9'da kümeler ve gruplar için maliyet faktörlerinin örnek değerleri görülmektedir. Çizelge 4.9'dan anlaşılmaktadır ki birinci ve üçüncü kümeler için toplam zaman maliyeti aynı olup bu maliyet ikinci kümede daha yüksektir. Ağdaki mesaj yükü maliyet faktörlerine bakıldığında ise birinci ve ikinci kümede bu maliyetler üçüncü kümeye göre daha yüksektir. Bu nedenle mesaj sayısı açısından yüksek maliyete sahip olan HMBR kodları üçüncü kümeye atanmıştı. Düğüm tamiri işleminin CPU üzerinde aldığı süre açısından en yüksek maliyete sahip küme olan birinci kümeye ise CPU işlem süresi diğer kodlama şemalarına nazaran daha düşük olan HSRC kodlama şeması atanmıştı. Düğüm tamiri işlemlerinin daha çok gerçekleştirildiği bir sistem için toplam zaman maliyetinin daha yüksek olduğu kümeye düğüm tamirindeki hız performansından dolayı e-MBR kodlama şeması atanmıştı.

Aşağıdaki eşitliklerde sırasıyla sadece HSRC kullanılan, sadece e-MBR kullanılan ve sadece HMBR kullanılan kümeleme tabanlı olmayan dağıtık depolama sistemlerinin, ayrıca kümeleme tabanlı dağıtık depolama sisteminin düğüm tamiri işleminin maliyet fonksiyonları bulunmaktadır. Maliyet fonksiyonları 10 MB, 100 MB ve 1 GB büyüklüklerindeki dosyaların yedekli olarak depolandığı sistemler için ayrı ayrı oluşturulmuştur. Maliyet fonksiyonlarında kullanılan simgeler Çizelge 4.9'da verilmiştir. Fonksiyonlarda bulunan katsayılar ise benzetimlerde ilgili mali-

Çizelge 4.9. Maliyet faktörlerinin aldığı değerler

Parametre	**Değeri**	**Parametre**	**Değeri**
t_1	1	$t_{g1,1}$	1.82
t_2	1.5	$t_{g1,2}$	0.5
t_3	1	$t_{g2,1}$	1.42
$t_{3,1}$	1.6	$t_{g2,2}$	0.9
$t_{3,2}$	0.4	$t_{g3,1}$	1.9
t_{ort}	1.16	$t_{g3,2}$	0.42
β_1	1	$\beta_{g1,1}$	1
β_2	0.9	$\beta_{g1,2}$	0.6
β_3	0.5	$\beta_{g2,1}$	1.1
$\beta_{3,1}$	0.7	$\beta_{g2,2}$	0.5
$\beta_{3,2}$	0.3	$\beta_{g3,1}$	0.9
β_{ort}	0.8	$\beta_{g3,2}$	0.7
t_{cpu1}	1.4	$t_{gcpu1,1}$	0.2
t_{cpu2}	0.3	$t_{gcpu1,2}$	1.42
t_{cpu3}	0.75	$t_{gcpu2,1}$	0.7
$t_{cpu_{ort}}$	0.81	$t_{gcpu2,2}$	0.92
$t_{cpu3,1}$	0.2	$t_{gcpu3,1}$	0.5
$t_{cpu3,2}$	1.3	$t_{gcpu3,2}$	1.12

yet faktörü hakkında elde edilen sonuçların normalize edilmesi yoluyla hesaplanmıştır. Normalize etme işlemi her bir faktör için sistemlerin oluşturduğu benzetim değerlerinin $[0, 1]$ aralığına ölçeklendirilmesi şeklinde gerçekleştirilmiştir.

$$\begin{aligned} c_{HSRC_{NR}}(10\text{ MB}) =& t_{ort}0.1896 + t_{ort}0.1708 + t_{ort}0.1625 + \beta_{ort}0.2974 \\ &+ \beta_{ort}0.3020 + \beta_{ort}0.3255 + 3t_{cpu_{ort}}0.0 \end{aligned} \tag{4.58}$$

$$\begin{aligned} c_{EMBR_{NR}}(10\text{ MB}) =& t_{ort}0.1792 + t_{ort}0.2479 + t_{ort}0.3438 + \beta_{ort}0.4988 \\ &+ \beta_{ort}0.5082 + \beta_{ort}0.4660 + 3t_{cpu_{ort}}1 \end{aligned} \tag{4.59}$$

$$
\begin{aligned}
c_{HMBR_{NR}}(10\text{ MB}) =&(t_{g1,1}0.2812 + t_{g1,2}0.7396)/2 + (t_{g2,1}0.1521 + t_{g2,2}0.5208)/2 \\
&+ (t_{g3,1}0.4917 + t_{g3,2}1)/2 \\
&+ (\beta_{g1,1}0.3536 + \beta_{g1,2}1)/2 \\
&+ (\beta_{g2,1}0.5128 + \beta_{g2,2}0.9859)/2 \\
&+ (\beta_{g3,1}0.5550 + \beta_{g3,2}0.8548)/2 \\
&+ (t_{gcpu1,1}0.9662 + t_{gcpu1,2}0.0338)/2 \\
&+ (t_{gcpu2,1}0.9662 + t_{gcpu2,2}0.0338)/2 \\
&+ (t_{gcpu3,1}0.9662 + t_{gcpu3,2}0.0338)/2
\end{aligned} \tag{4.60}
$$

$$
\begin{aligned}
c_{Kume_{NR}}(10\text{ MB}) =&t_1 0.0396 + t_2 0 + (t_{3,1}0.1104 + t_{3,2}0.7583)/2 + \beta_1 0.7293 \\
&+ \beta_2 0.0290 + (\beta_{3,1}0 + \beta_{3,2}0.1791)/2 + t_{cpu1}0 \\
&+ t_{cpu2}1 + (t_{cpu3,1}0.9662 + t_{cpu3,2}0.0338)/2
\end{aligned} \tag{4.61}
$$

$$
\begin{aligned}
c_{HSRC_{NR}}(100\text{ MB}) =&t_{ort}0.1104 + t_{ort}0.1186 + t_{ort}0.2315 + \beta_{ort}0.3267 \\
&+ \beta_{ort}0.2687 + \beta_{ort}0.3623 + 3t_{cpu_{ort}}0.0
\end{aligned} \tag{4.62}
$$

$$
\begin{aligned}
c_{EMBR_{NR}}(100\text{ MB}) =&t_{ort}0.1197 + t_{ort}0.1238 + t_{ort}0.1217 + \beta_{ort}0.4337 \\
&+ \beta_{ort}0.3846 + \beta_{ort}0.4604 + 3t_{cpu_{ort}}1
\end{aligned} \tag{4.63}
$$

$$
\begin{aligned}
c_{HMBR_{NR}}(100\text{ MB}) =&(t_{g1,1}0.0879 + t_{g1,2}1)/2 + (t_{g2,1}0.1698 + t_{g2,2}0.7394)/2 \\
&+ (t_{g3,1}0.1032 + t_{g3,2}0.7211)/2 + (\beta_{g1,1}0.4292 \\
&+ \beta_{g1,2}0.8885)/2 + (\beta_{g2,1}0.4381 \\
&+ \beta_{g2,2}0.7012)/2 + (\beta_{g3,1}0.4426 \\
&+ \beta_{g3,2}1)/2 \\
&+ (t_{gcpu1,1}0.6331 + t_{gcpu1,2}0.0216)/2 \\
&+ (t_{gcpu2,1}0.6331 + t_{gcpu2,2}0.0216)/2 \\
&+ (t_{gcpu3,1}0.6331 + t_{gcpu3,2}0.0216)/2
\end{aligned} \tag{4.64}
$$

$$
\begin{aligned}
c_{Kume_{NR}}(100\text{ MB}) =&t_1 0 + t_2 0.00071891 + (t_{3,1}0.0500 + t_{3,2}0.4826)/2 \\
&+ \beta_1 0.7453 + \beta_2 0.0080 + (\beta_{3,1}0 + \beta_{3,2}0.1609)/2 \\
&+ t_{cpu1}0.0 + t_{cpu2}1.0 + (t_{cpu3,1}0.6331 + t_{cpu3,2}0.0216)/2
\end{aligned} \tag{4.65}
$$

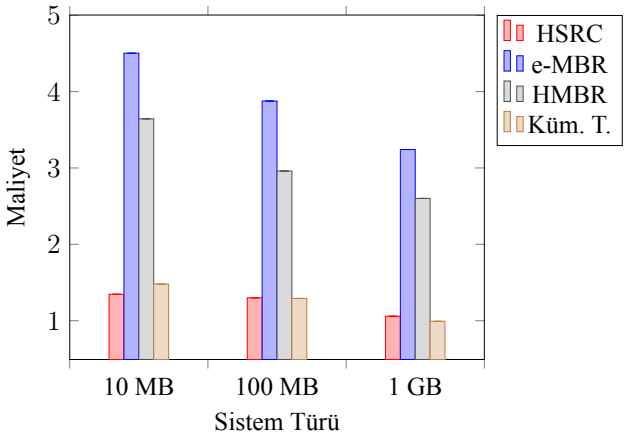
$$c_{HSRC_{NR}}(1\text{ GB}) = t_{ort}0.1982 + t_{ort}0.1931 + t_{ort}0.3631 + \beta_{ort}0.0411 + \beta_{ort}0.1268 + \beta_{ort}0.0619 + 3t_{cpu_{ort}}0.0 \quad (4.66)$$

$$c_{EMBR_{NR}}(1\text{ GB}) = t_{ort}0.1673 + t_{ort}0.1181 + t_{ort}0.1292 + \beta_{ort}0.1388 + \beta_{ort}0.1597 + \beta_{ort}0.1149 + 3t_{cpu_{ort}}1.0 \quad (4.67)$$

$$\begin{aligned} c_{HMBR_{NR}}(1\text{ GB}) = &(t_{g1,1}0.1010 + t_{g1,2}0.5716)/2 + (t_{g2,1}0.0784 \\ &+ t_{g2,2}1)/2 + (t_{g3,1}0.1239 + t_{g3,2}0.7330)/2 \\ &+ (\beta_{g1,1}0.1208 \\ &+ \beta_{g1,2}0.3701)/2 \\ &+ (\beta_{g2,1}0.0484 + \beta_{g2,2}0.2104)/2 \\ &+ (\beta_{g3,1}1 + \beta_{g3,2}0.4283)/2 \\ &+ (t_{gcpu1,1}0.9521 + t_{gcpu1,2}0.0417)/2 \\ &+ (t_{gcpu2,1}0.9521 + t_{gcpu2,2}0.0417)/2 \\ &+ (t_{gcpu3,1}0.9521 + t_{gcpu3,2}0.0417)/2 \end{aligned} \quad (4.68)$$

$$\begin{aligned} c_{Kume_{NR}}(1\text{ GB}) = &t_1 0.2493 + t_2 0 + (t_{3,1}0.0727 + t_{3,2}0.2149)/2 \\ &+ \beta_1 0.2152 + \beta_2 0 + (\beta_{3,1}0.0011 + \beta_{3,2}0.0422)/2 \\ &+ t_{cpu1}0.0 + t_{cpu2}1.0 + (t_{cpu3,1}0.9521 + t_{cpu3,2}0.0381)/2 \end{aligned} \quad (4.69)$$

Çizelge 4.9'da verilen değerlere göre 10 MB, 100 MB ve 1 GB büyüklüğünde bir dosyanın yedekli bir şekilde depolandığı sistemlerdeki düğüm tamiri maliyetleri Şekil 4.36'da verilmiştir. Şekil 4.36'ya göre 10 MB dışında en düşük maliyete sahip



Şekil 4.36. 10 MB, 100 MB ve 1 GB dosya tamiri için sistemlerin maliyetleri.

sistem kümeleme tabanlı dağıtık depolama sistemidir. Ardından ona çok yakın maliyetlere sahip olan HSRC kodlama şemasının kullanıldığı kümeleme tabanlı olmayan dağıtık depolama sistemi gelmektedir. Bu sistem CPU işlem süresi açısından en

düşük maliyete sahip sistemdir. Ancak bu kodlama şemasının düğüm tamiri başarım olasılığı kümeleme tabanlı dağıtık depolama sisteminde kullanılan e-MBR ve HMBR kodlama şemasına göre daha düşüktür. Bu durumun kuramsal sonuçları alt bölüm 4.2.3'te verilmiştir. Kümeleme tabanlı dağıtık depolama sisteminin maliyet sonuçları üzerinde kümeleme işleminin de etkisinin olduğu görülmektedir. Birinci küme içerisinden diğer kümelerle ilgili bir çok mesaj geçişi olduğu için ağ trafiği açısından yüksek maliyet faktörüne sahip bu kümede düşük ağ yükü oluşturan bir kodlama şeması kullanılmasına rağmen bu alanda yüksek maliyet oluşmuştur. Bu durumu ortadan kaldırmak için, kümeleme işlemi yapılırken yönlendirici tabloları daha az kesişen düğümler hesaba katılabilir ya da kümeleme işlemi sonrasında yönlendirici tabloları üzerinde değişiklikler yapılabilir.

Toplam maliyet açısından HSRC kodlama şemasının kullanıldığı kümeleme tabanlı olmayan dağıtık depolama sisteminden sonra sırasıyla yalnızca HMBR kodlama şemasının ve yalnızca e-MBR kodlama şemasının kullanıldığı kümeleme tabanlı olmayan dağıtık depolama sistemleri gelmektedir. HMBR kodlama şemasının ikinci düğüm tamiri yönteminin zaman ve ağ trafiğinde sergilediği kötü performansına karşın sistemin farklı zaman aralıklarında farklı maliyet faktörü değerlerine sahip olması bu sistemi e-MBR şemasının kullanıldığı sistemden daha az maliyetli hale getirmiştir. Burada zamana göre değişen maliyet faktörlerinin sistemin toplam maliyetine büyük ölçüde etkisinin olduğu görülmektedir.

Elde edilen sonuçlar Çizelge 4.9'a göre oluşturulmuş sonuçlar olup, maliyet faktörleri farklı değerler aldığında bu sonuçlar değişebilmektedir. Sistem tasarımcıları oluşturulan maliyet fonksiyonlarını kullanarak kendi platformları için hangi kodlama şemasını ve hangi sistemi seçeceklerine karar verebilirler.

Kodlama şemalarının düğüm tamiri başarım olasılıkları da birer maliyet olarak düşünülürse yukarıda elde edilen maliyet fonksiyonları aşağıdaki şekilde genişletilebilir.

$$c_{sistem'_{NR}}(x) = \delta c_{\widehat{sistem}_{NR}}(x) + \gamma(1\text{–}p_{nr_{sistem}}) \tag{4.70}$$

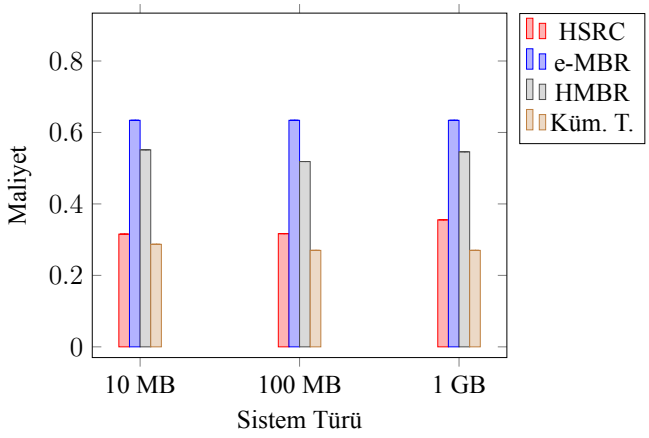
Eşitlik (4.70)'de $c_{\widehat{sistem}_{NR}}(x)$, ilgili dağıtık depolama sisteminin x boyutundaki bir dosyanın tamiri için oluşturduğu maliyetin normalize edilmiş hali, $1 - p_{nr_{sistem}}$ ise bu sistemdeki ortalama düğüm tamiri gerçekleştirilememe olasılığını vermektedir. γ ve δ değerleri ise sistem maliyetinde düğüm tamiri başarım olasılığının ve diğer kaynak kullanımlarının önemini belirleyen faktörlerdir. Örneğin yeni göreceli maliyet fonksiyonu sadece HSRC kodlama şemasının kullanıldığı ve 10 MB büyüklüğünde

dosyaların depolandığı bir sistem için aşağıdaki gibidir.

$$c_{HSRC'_{NR}}(10\text{ MB}) = \delta c_{\widehat{HSRC}_{NR}}(10\text{ MB}) + \gamma(1\text{–}p_{nr_{HSRC}}) \tag{4.71}$$

Düğümlerin çalışır durumda olma olasılığı; 0.5'den küçük eşit ve 0.5'den büyük olmak üzere iki gruba ayrılmıştır. Bu iki farklı duruma göre her bir kodlama şeması için alt bölüm 4.2.3'te verilen başarım olasılıkları sonuçlarının ortalaması kullanılmıştır. Bu iki farklı durum için iki farklı düğüm tamiri maliyet hesabı gerçekleştirilmiştir.

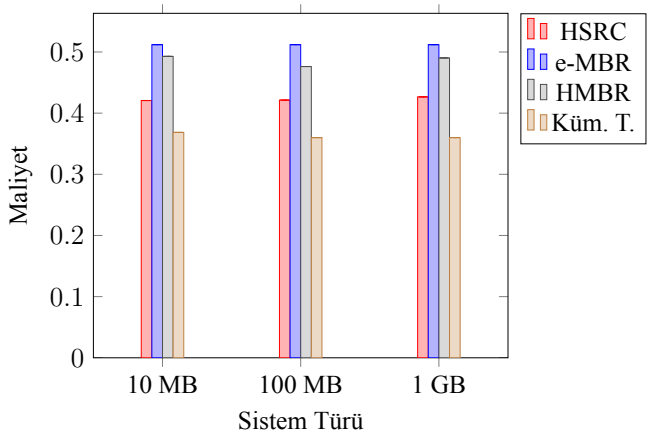
Sistemlerde bulunan düğümlerin çalışır durumda olma olasılıklarının 0.5'den küçük ve eşit olduğu durum için sistemlerin düğüm tamiri işlemindeki göreceli maliyetleri, $\delta = 0.4$ ve $\gamma = 0.6$ değerleri kullanıldığında Şekil 4.37'deki gibi çıkmıştır. Sistemlerde bulunan düğümlerin çalışır durumda olma olasılıklarının 0.5'den kü-



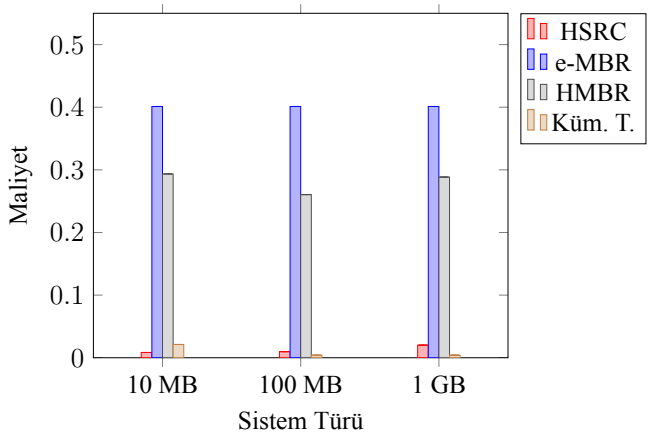
Şekil 4.37. 10 MB, 100 MB ve 1 GB dosya tamiri için sistemlerin göreceli maliyetleri ($\delta = 0.4$, $\gamma = 0.6$ ve $p_{saglam} \leq 0.5$ için).

çük ve eşit olduğu durum için sistemlerin düğüm tamiri işlemindeki yeni maliyetleri $\delta = 0.2$ ve $\gamma = 0.8$ değerleri kullanıldığında ise Şekil 4.38'deki gibi çıkmıştır. Bu ağırlıklandırmada düğüm tamiri başarımına daha çok önem vermiştir. Bu nedenle sadece e-MBR ve HMBR kodlama şemasının kullanıldığı kümeleme tabanlı olmayan dağıtık depolama sistemlerinin göreceli maliyeti düşmüştür. Bir sistem tasarımcısı bu değerleri değiştirerek sistemin düğüm tamiri başarımına ve kaynak kullanımına verdiği önemi ayarlayabilmektedir.

Sistemlerde bulunan düğümlerin çalışır durumda olması olasılığının 0.5'den büyük olduğu düşünüldüğünde ise üç kodlama şeması da 1'e yakın başarım olasılıklarına sahip olmaktadır. Bu durumda $\delta = 0.4$ ve $\gamma = 0.6$ değerleri için sistemlerin düğüm tamirindeki yeni göreceli maliyetleri Şekil 4.39'da verilmiştir. Düğümlerin çalışır durumda olma olasılığı 0.5'den büyük olduğunda tüm kodlama şemaları



Şekil 4.38. 10 MB, 100 MB ve 1 GB dosya tamiri için sistemlerin göreceli maliyetleri ($\delta = 0.2$, $\gamma = 0.8$ ve $p_{saglam} \leq 0.5$ için).



Şekil 4.39. 10 MB, 100 MB ve 1 GB dosya tamiri için sistemlerin göreceli maliyetleri ($\delta = 0.4$, $\gamma = 0.6$ ve $p_{saglam} > 0.5$ için).

için düğüm tamiri başarım olasılıkları 1'e yaklaşmaktadır. Bu yüzden, yeni göreceli maliyet hesabının sonucu daha çok kaynak kullanımı ile ilgili olmaktadır. Şimdiye kadar yapılan göreceli maliyet hesaplarında kümeleme tabanlı dağıtık depolama sistemi için tüm kümelerdeki düğüm tamiri başarım maliyeti aynı olarak kabul edilmiştir. Düğüm tamiri ile ilgili maliyet hesabı farklı kümelerin düğüm tamiri başarımına verdiği önem derecesi değiştirilerek yeniden yapılandırılabilir.

Aşağıda kümeleme tabanlı olan ve olmayan dağıtık depolama sistemlerinde verinin geri çatımı işlemleri için elde edilen maliyet fonksiyonları verilmiştir. Verinin geri çatımı ile ilgili oluşturulan maliyet fonksiyonunlarının maliyet faktörleri de Çizelge 4.9 ile aynıdır. Ancak, verinin geri çatılması işleminde HMBR kodlama şemasının sadece ilk yöntemi kullanıldığı için bu kodlama şemasının kullanıldığı küme ve gruplar için sırasıyla $_3$ ve $_{ort}$ alt indisli parametreler kullanılmıştır.

Verinin geri çatılması için oluşturulmuş maliyet fonksiyonlarının maliyet faktörü katsayılarını, düğüm tamiri işlemi için oluşturulan maliyet fonksiyonlarındaki gibi benzetim ile elde edilmiş maliyet faktörü sonuçlarının kendi aralarında normalize edilmiş değerleri oluşturmaktadır.

$$c_{HSRC_{DR}}(10\text{ MB}) = t_{ort}0.4466 + t_{ort}0.7414 + t_{ort}0.6293 + \beta_{ort}0.0839 + \beta_{ort}0.1364 + \beta_{ort}0.1119 + 3t_{cpu_{ort}}0 \tag{4.72}$$

$$c_{EMBR_{DR}}(10\text{ MB}) = t_{ort}0.9207 + t_{ort}0.7845 + t_{ort}0.7569 + \beta_{ort}0.7203 + \beta_{ort}0.7622 + \beta_{ort}0.8636 + 3t_{cpu_{ort}}1 \tag{4.73}$$

$$c_{HMBR_{DR}}(10\text{ MB}) = t_{ort}0.7862 + t_{ort}1 + t_{ort}0.6121 + \beta_{ort}0.9126 + \beta_{ort}1 + \beta_{ort}0.9441 + 3t_{cpu_{ort}}0.9737 \tag{4.74}$$

$$c_{Kume_{DR}}(10\text{ MB}) = t_1 0.2328 + t_2 0 + t_3 0.6379 + \beta_1 0.8427 + \beta_2 0.0105 + \beta_3 0 + t_{cpu1}0 + t_{cpu2}1 + t_{cpu3}0.9737 \tag{4.75}$$

$$c_{HSRC_{DR}}(100\text{ MB}) = t_{ort}0 + t_{ort}0.0727 + t_{ort}0.0929 + \beta_{ort}0.1437 + \beta_{ort}0.1562 + \beta_{ort}0.1313 + 3t_{cpu_{ort}}0 \tag{4.76}$$

$$c_{EMBR_{DR}}(100\text{ MB}) = t_{ort}0.8299 + t_{ort}1 + t_{ort}0.9048 + \beta_{ort}0.9 + \beta_{ort}1 + \beta_{ort}0.9125 + 3t_{cpu_{ort}}0.8776 \tag{4.77}$$

$$c_{HMBR_{DR}}(100\text{ MB}) = t_{ort}0.7334 + t_{ort}0.8749 + t_{ort}0.5840 + \beta_{ort}0.7906 + \beta_{ort}0.8031 + \beta_{ort}0.8375 + 3t_{cpu_{ort}}1 \tag{4.78}$$

$$c_{Kume_{DR}}(100\text{ MB}) = t_1 0.2067 + t_2 0.5441 + t_3 0.4947 + \beta_1 0.9313 + \beta_2 0.0375 + \beta_3 0 + t_{cpu1}0 + t_{cpu2}0.8776 + t_{cpu3}1 \tag{4.79}$$

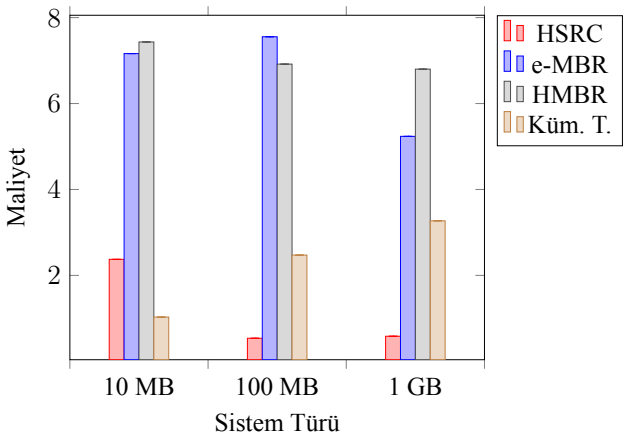
$$c_{HSRC_{DR}}(1\text{ GB}) = t_{ort}0 + t_{ort}0.1598 + t_{ort}0.0169 + \beta_{ort}0.0942 + \beta_{ort}0.2133 + \beta_{ort}0.1662 + 3t_{cpu_{ort}}0 \tag{4.80}$$

$$c_{EMBR_{DR}}(1\text{ GB}) = t_{ort}0.4495 + t_{ort}0.7839 + t_{ort}0.2761 + \beta_{ort}0.5651 + \beta_{ort}1 + \beta_{ort}0.7424 + 3t_{cpu_{ort}}0.6746 \tag{4.81}$$

$$c_{HMBR_{DR}}(1\text{ GB}) = t_{ort}1 + t_{ort}0.4511 + t_{ort}0.8176 + \beta_{ort}0.9418 + \beta_{ort}0.5346 + \beta_{ort}0.6981 + 3t_{cpu_{ort}}1 \tag{4.82}$$

$$c_{Kume_{DR}}(1\text{ GB}) = t_1 0.3655 + t_2 0.3649 + t_3 0.5264 + \beta_1 0.8338 + \beta_2 0.0499 + \beta_3 0 + t_{cpu1} 0 + t_{cpu2} 0.6746 + t_{cpu3} 1 \quad (4.83)$$

Şekil 4.40'ta Çizelge 4.9'daki maliyet faktörleri değerleri kullanılarak sırasıyla 10 MB, 100 MB ve 1 GB büyüklüğündeki dosyaların geri çatılması işlemi için sistemlerin oluşturduğu maliyetler görülmektedir. Şekil 4.40'a göre verinin geri çatılması



Şekil 4.40. 10 MB, 100 MB ve 1 GB dosyanın geri çatılması için sistemlerin maliyetleri.

için en düşük maliyeti, 10 MB dosyanın dışında, sadece HSRC kodlama şemasının kullanıldığı kümeleme tabanlı olmayan dağıtık depolama sistemi oluşturmaktadır. Bu sistemin oluşturduğu maliyet 100 MB ve 1 GB boyutlarındaki dosyaların geri çatılması için çok düşük seviyede olup 10 MB büyüklüğündeki bir dosya için daha yüksek seviyededir. Bu sistem 10 MB büyüklüğündeki bir dosyanın geri çatılması için kullanılan toplam süre alanında diğer sistemlere göre çok düşük sonuçlar göstermemiştir. Ancak sadece HSRC kodlama şemasının kullanıldığı kümeleme tabanlı olmayan dağıtık depolama sistemi 100 GB ve 1 GB büyüklüğünde bir dosyanın geri çatılması işlemi için hem kullanılan toplam sürede hem de CPU süresinde çok düşük sonuçlar oluşturduğu için diğer sistemlerin toplam maliyetinden çok daha düşük bir maliyete sahiptir.

Sadece e-MBR ve sadece HMBR kodlama şemasının kullanıldığı kümeleme tabanlı olmayan dağıtık depolama sistemleri ise birbirine yakın maliyetlere sahip olup bu maliyetler kümeleme tabanlı dağıtık depolama sisteminin oluşturduğu maliyetlerden daha yüksektir.

Dağıtık depolama sistemi tasarımcıları, çeşitli sistemler için verinin geri çatılması işlemleri hakkında yukarıda oluşturulan maliyet fonksiyonlarını kendi sistemlerine özel maliyet faktörleri ile kullanarak kendi ihtiyaçlarına en uygun sisteme

karar verebilirler.

Verinin geri çatılması işleminin başarılı bir gerçekleştirilmesi HSRC ve HMBR kodlama şemalarında her zaman mümkün olmamaktadır. Bu yüzden verinin geri çatılması işleminin gerçekleşememesi durumları da bir maliyet faktörü olarak görülebilir. Şekil 4.9'da $n = 15,\ k = 3,\ d = 4$ için HSRC, e-MBR ve HMBR kodlama şemalarında verinin geri çatılması başarım olasılığı değerleri verilmiştir. Bu $n,\ k,\ d$ değerleri için tüm kodlama şemalarının başarım olasılıkları hemen hemen aynı olduğu için burada ayrıca bir maliyet analizi gerçekleştirilmeyecektir. Ancak sistem tasarımcıları, bu kodlama şemalarının farklı verinin geri çatılması olasılığına sahip olduğu $n,\ k,\ d$ değerleri için elde edilen sonuçlardan yararlanarak verinin geri çatımı başarım olasılığının kendi sistemlerinin maliyeti üzerinde etkisini inceleyebilirler.

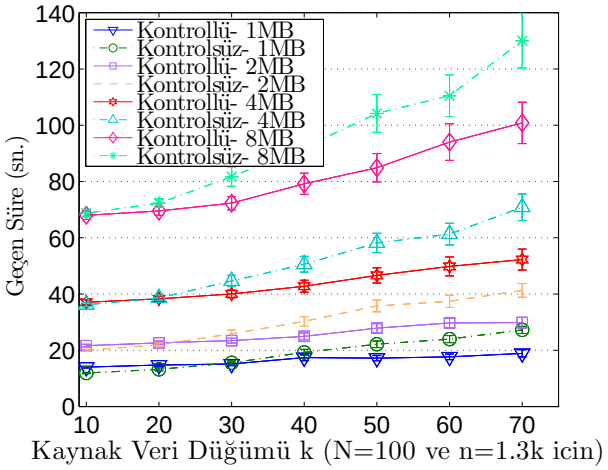
4.5 Silinti Kodlarında Dosya Geri Çatma İşlemi İçin TCP Bağlantılarının Yönetimi İle İlgili Elde Edilen Bulgular

Önerilen kontrol çatısını kullanan ve kullanmayan geri çatma işlemleri ns-3 (ns-3, 2015) benzetim ortamında toplamda $N = 100$ düğümlük bir ağda gerçekleştirilmiştir. Ağ modeli olarak Waxman topolojisi kullanılmış, bant genişliği ise 10Mbps ve 100 Mbps arasından düzgün dağılımla seçilmektedir. Ağdaki hatların gecikme süresi 50 ms olarak atanmıştır. Her iki geri çatma işleminde, tıkanıklıktan kaçınma algoritması olarak ns-3'ün varsayılan olarak kullandığı TCP New-Reno (Floyd and Henderson, 1999) kullanılmıştır. Reed-Solomon kodlama işleminde kodlama ve kod çözme işleminde sonlu cisim işlemleri için Jerasure (Plank et al., 2008) kütüphanesi kullanılmışır. Benzetim sonuçlarındaki her nokta toplamda 50 farklı benzetimin ortalaması olarak verilmiştir. Kontrol çatısının kullanıldığı geri çatma işlemi için geçen sürenin içerisinde hiyerarşi belirleme algoritmasının gerektirdiği süre de bulunmaktadır.

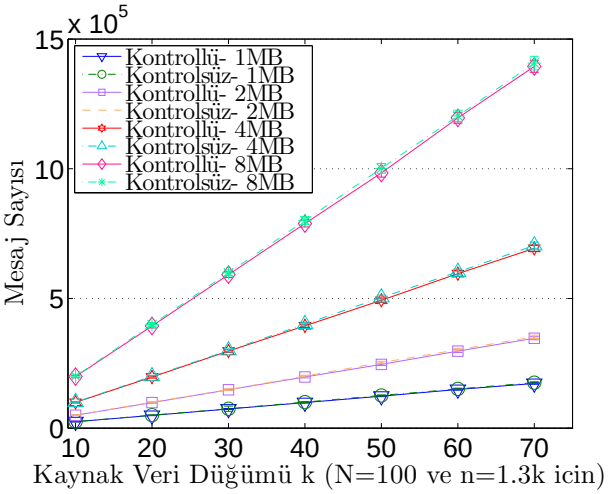
Şekil 4.41'de, Reed-Solomon kodlama kullanılan bir dağıtık depolama sisteminde dosya parça boyu 1, 2, 4 ve 8 MB olan dosyaların geri çatılması işlemi için farklı sayılarda kaynak veri düğümü (k) kullanıldığında her iki yöntemde harcanan toplam süreler gösterilmektedir. Kontrollü yöntem, geri çatma işlem sürelerini kaynak veri düğümü sayısı 30'u geçtiğinde parça boyutuna bağlı olarak %16-19 arasında azaltmaktadır.

Dosya geri çatma işleminde kullanılan mesaj sayıları Şekil 4.42'de görülmek-

tedir. Buna göre, önerilen yöntemi kullanan dosya geri çatma işlemi, daha az sayıda mesaj kullanımı gerektirmektedir. Bu fark, toplamda kullanılan mesaj sayıları çok yüksek olduğu için Şekil 4.42'de belirgin olmamakla birlikte yüksek k değerlerinde binlerle ifade edilmektedir (Örn. $k = 40$ ve 4 MB dosya parçası için bu fark 6643'tür). Kontrollü yöntemde, lime gönderimlerinin lime numaralarına göre kısıtlanması ağdaki tıkanıklığı hafifletmekte ve paket yeniden gönderimlerinin sayısını azaltmaktadır. Kontrollü yöntem, kontrolsüz yönteme kıyasla fazladan bir hiyerarşi belirleme algoritması ve ek kontrol mesajları kullanmasına rağmen çalıştırdığı algoritmaların zamansal maliyeti ve kullandığı kontrol mesajlarının boyutunun düşük olması nedeniyle geri çatma işleminde kullanılan toplam ağ trafiğini ve süreyi azaltmaktadır.



Şekil 4.41. Geri çatma için kullanılan toplam süre.



Şekil 4.42. Geri çatma için kullanılan toplam mesaj sayısı.

Bu alt bölümde, Reed-Solomon silinti kodlama kullanan dağıtık depolama sis-

temlerinde, dosya geri çatma işlemi için kullanılan toplam sürenin ve mesaj sayısının; ağ tıkanıklığı durumunda düşürülmesini amaçlayan yöntemin performansı incelenmiştir. Sonuç olarak, bu yöntemin, dağıtık depolama sistemlerinin geri çatma işlemi için kullandığı toplam süreyi azatabileceği ve ağ trafiğinin maliyetini düşürebileceği benzetim sonuçları ile gösterilmiştir.

5. TARTIŞMA, SONUÇ VE ÖNERİLER

Bu doktora tezinde temel olarak dağıtık depolama sistemlerinde verilerin yedekli bir şekilde hata toleranslı olarak depolanmaları konusu üzerinde çalışılmıştır. Bu sistemlerin verileri hata toleranslı bir şekilde yedeklemesi sağlanırken gerçekleştirdikleri işlemlerin çeşitli maliyetlerinin azaltılması üzerinde beş farklı çalışma gerçekleştirilmiştir.

İlk çalışmada, MDS kodlama kullanan dağıtık depolama sistemleri için topoloji farkındalıklı bir çalışma gerçekleştirilmiştir. Bu çalışmada düğüm tamiri ve verinin geri çatılması işlemlerinin başlatımı için STA ve MPA yaklaşımları denenmiştir. Ayrıca MDS silinti kodları ile kodlanmış bir verinin güncelleme işlemi için de STA ve MPA yaklaşımları test edilmiştir. Bu yaklaşımlarla oluşan mesaj karmaşıklığı ve iletişim süresi için kuramsal sonuçlar elde edilmiş, bu kuramsal sonuçlar benzetim sonuçları ile desteklenmiştir. Bu sonuçlara göre, düğüm tamirinin ve verinin geri çatılmasının başlatılması işlemlerinde MPA yaklaşımı iletişim süresi konusunda avantaj sahibi iken STA yaklaşımı kullanılan mesaj sayısı açısından avantajlı durumdadır. Güncelleme işleminde ise STA yaklaşımı hem mesaj kullanımı hem de iletişim süresi maliyetleri açısından MPA yaklaşımına üstünlük sağlamaktadır. Bu sonuçlar dağıtık depolama sistemi tasarlayıcılarına kendi sistemlerinin önceliklerini ve ihtiyaçlarını göz önünde bulundurarak iki yaklaşımdan birini mantıklı bir şekilde seçmelerinde yardımcı olacaktır. Örnek olarak, zaman kısıtı olmayan ama bant genişliği açısından kısıtlı olan bir sistem için hem düğüm tamirinin ve verinin geri çatılmasının başlatımı işlemleri hem de güncelleme işlemi için STA yaklaşımı daha mantıklı bir seçim olacaktır. Ancak bant genişliği maliyetleri düşük olan ama süre kısıtı olan sistemler için düğüm tamirinin ve verinin geri çatılmasının başlatımı işlemlerinde MPA yaklaşımının seçilmesi daha makul görünmektedir. Son olarak, STA yaklaşımının tüm işlemler için avantajlı hale getirilmesi için ağacın uzunluğunu belirli bir seviyede tutabilen yeni bir Steiner ağaç algoritmasının geliştirilmesi açık bir alan olarak görülmektedir.

Bu çalışmaların ikincisinde iki farklı düğüm tamiri ve iki farklı verinin geri çatımı yöntemi bulunduran bir melez kodlama şeması geliştirilmiştir. Bu melez kodlama şeması e-MBR kodlama şemasının düğüm tamirinde daha az bant genişliği kullanımı, HSRC kodlama şemasının ise düğüm tamirinde düşük hesapsal karmaşıklık oluşturması ve daha az düğüme bağlanması gibi avantajları bünyesinde bulundurmaktadır. Bu yeni kodlama şeması Homomorfik Minimum Bant Genişliği Tamir Kodları (HMBR) olarak adlandırılmıştır. HMBR kodlama şeması tamir edilen bir düğümün içeriğinin bütünlüğünün kontrol edilmesi için de bir mekanizma

bulundurmaktadır. Bu kodlama şeması, içerisinde farklı kaynak kapasitelerine sahip düğümlerin bulunduğu depolama sistemleri için uygun olabilir. Böylece düşük bant genişliği ve yüksek hesaplama kaynaklarına sahip olan düğümler $AMBR_{NR}$ tamir etme yöntemini, yüksek bant genişliği ve düşük hesaplama kapasiteli kaynaklara sahip olan düğümler ise $ASRC_{NR}$ tamir etme yöntemini kullanabilir. Sistemde bir çok düğüm arızalanması var ise ancak daha henüz sistemde tüm veriyi kaybetme olasılığı düşük ise arızalanmış düğümleri düşük bant genişliği kullanarak $AMBR_{NR}$ yöntemi ile tamir etmek iyi bir seçenek olabilir. Çünkü bu şekilde ağda oluşan tıkanmaların önüne geçilebilir. Ayrıca eğer sistemdeki arızalanmaların sayısı veriyi kaybedecek derecede çoksa bu durumda bant genişliği maliyeti ikinci plandadır ve düğümlerin bir an önce daha çok arızalanma olmadan tamir edilmesi gerekmektedir. Bu durumda daha az hesapsal karmaşıklıkla düğümleri tamir edebilen $ASRC_{NR}$ yöntemi kullanılabilir. Bunun yanında, dağıtık depolama sisteminin içerisindeki düğümler farklı farklı işlemler ile meşgül olabilir (Örn: verinin geri çatılması, düğüm tamirleri). Bu yüzden hesapsal karmaşıklık yükü düşük olan ve yüksek bant genişliğine sahip hatlar için daha hızlı yanıt süresi sağlayabilen $ASRC_{NR}$ tamir etme yöntemi kullanılabilir.

Üçüncü çalışmada da ikinci çalışmaya benzer şekilde yeni bir melez kodlama şeması geliştirilmiştir. Bu yeni kodlama şeması ise Homomorfik Minimum Depolama Tamir Kodları (HMSR) olarak adlandırılmıştır. Bu kodlar minimum depolama noktasını sağlamaktadır. Bu kodlama şeması da iki farklı düğüm tamiri işlemi bulundurmaktadır. Bunlardan birincisi ikincisine göre daha çok hesapsal karmaşıklık oluşturmakta ve daha az bant genişliği kullanımı gerektirmekte, diğer tamir etme yöntemi ise daha çok bant genişliği harcaması ve daha az hesapsal karmaşıklık gerektirmektedir. Bu kodlarda $AMSR_{NR}$ düğüm tamiri yönteminin başarım olasılığının kuramsal analizi açık bir problem olarak kalmıştır.

Dördüncü çalışmada ise kümeleme tabanlı dağıtık depolama sistemi çalışması gerçekleştirilmiştir. Bu çalışmanın temelini bir dağıtık depolama sistemini oluşturan düğümlerin farklı kısıtlı kaynaklara sahip olabilmesi oluşturmuştur. Bir dağıtık depolama sistemini oluşturan düğümlerin kaynakları ve/veya iş yükleri eş yapılı bir şekilde dağılmamışsa bir kodlama şeması bir düğüm için iyi sonuçlar verirken diğeri için iyi sonuçlar vermeyebilir. Bu yüzden depolama sistemini oluşturan düğümlerin farklı performans parametrelerini iyileştirmeyi hedefleyen kodlama şemalarını kullanması sistemin toplam performansını iyileştirebilecektir. Bu motivasyon ile dördüncü çalışmada bir sistemi oluşturan düğümlerin kümelendirilerek her bir kümenin kendine uygun olan kodlama şemasını kullandığı bir dağıtık depolama sistemi ns-3 ortamında gerçekleştirilmiştir. Bu sistem kümeleme tabanlı olmayan sistemlerle

karşılaştırılmıştır. Kümeleme tabanlı dağıtık depolama sisteminde düğüm tamiri işleminde harcanan süre, süre kısıtı olan düğümler için iyileştirilirken, CPU kısıtı olan düğümler ise daha az hesapsal karmaşıklık gerektiren kodlama şemalarını kullanabilmiştir. Ayrıca, bu çalışmada çeşitli sistemlerin düğüm tamiri ve verinin geri çatılması işlemlerinde oluşturduğu maliyetlerin analizi gerçekleştirilmiştir. Bu sayede, sistemlerin avantaj ve dezavantajları daha somut bir şekilde görülmüştür. Bu maliyet analizi, sistem tasarımcılarına kendi sistemlerine uygun çözümü bulmalarında faydalanacakları bir kaynak niteliğindedir. Bu çalışma daha fazla maliyet parametresinin benzetim ölçümü yapılarak genişletilebilir.

Son çalışmada ise ağ tıkanması durumunda TCP soketlerinin yönetilmesi yoluyla silinti kodlama kullanan dağıtık depolama sistemlerinin verinin geri çatılması işleminin toplam süresinin azaltılması amaçlanmıştır. Bu çalışmada iki farklı aşamadan oluşan bir algoritma önerilmiştir. İlk aşamada verinin geri çatılması işleminde görevli olan düğümler arasında bir hiyerarşi oluşturulmakta, ikinci aşamada ise aralarında hiyerarşi oluşmuş düğümlerin birbirleri arasındaki TCP bağlantıları yönetilmektedir. Yapılan benzetimlerde önerilen algoritma ile gerçekleştirilen verinin geri çatılması işleminin daha hızlı gerçekleştiği görülmüştür.

KAYNAKLAR DİZİNİ

Aguilera, M.K., Janakiraman, R. and Xu, L., 2005, On the erasure recoverability of MDS codes under concurrent updates, In Information Theory Proceedings (ISIT), IEEE International Symposium on, Adelaide, SA, Australia, 1358-1362pp.

Alizadeh, M., Atikoglu, B., Kabbani, A., Lakshmikantha, A., Pan, R., Prabhakar, B. and Seaman, M., 2008, Data center transport mechanisms: Congestion control theory and IEEE standardization, In Communication, Control, and Computing, 46th Annual Allerton Conference on, Urbana-Champaign, IL, USA, 1270-1277pp.

Alizadeh, M., Greenberg, A., Maltz, D.A., Padhye, J., Patel, P., Prabhakar, B., Sengupta, S. and Sridharan, M., 2010, Data Center TCP (DCTCP), In Proceedings of the ACM SIGCOMM Conference, New Delhi, India, 63–74pp.

Anthapadmanabhan, N., Soljanin, E. and Vishwanath, S., 2010, Update-efficient codes for erasure correction, In Communication, Control, and Computing (Allerton), 48th Annual Allerton Conference on, Allerton, IL, USA, 376-382pp.

Araujo, J., Giroire, F. and Monteiro, J., 2011, Hybrid Approaches for Distributed Storage Systems, In Proceedings of the 4th International Conference on Data Management in Grid and Peer-to-peer Systems, Springer-Verlag, Berlin, Heidelberg, Globe'11, 1–12pp.

Berman, P. and Ramaiyer, V., 1994, Improved approximations for the Steiner tree problem, *Journal of Algorithms*, 17(3):381–408pp.

Bernstein, D.S., 2005, Matrix Mathematics: Theory, Facts, and Formulas with Application to Linear Systems Theory, Princeton University Press, 682p.

Bhagwan, R., Tati, K., Cheng, Y.C., Savage, S. and Voelker, G.M., 2004, Total recall: system support for automated availability management, In Proceedings of the 1st conference on Symposium on Networked Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, 25–25pp.

Brookes M., 2011, "The Matrix Reference Manual", http://www.ee.imperial.ac.uk/hp/staff/dmb/matrix/intro.html (Erişim tarihi: 28 Eylül 2015).

KAYNAKLAR DİZİNİ (devam)

Calder, B., Wang, J., Ogus, A., Nilakantan, N., Skjolsvold, A., McKelvie, S., Xu, Y., Srivastav, S., Wu, J., Simitci, H., Haridas, J., Uddaraju, C., Khatri, H., Edwards, A., Bedekar, V., Mainali, S., Abbasi, R., Agarwal, A., Haq, M.F., Haq, M.I., Bhardwaj, D., Dayanand, S., Adusumilli, A., McNett, M., Sankaran, S., Manivannan, K. and Rigas, L., 2011, Windows Azure Storage: A Highly Available Cloud Storage Service with Strong Consistency, In Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, Cascais, Portugal, 143–157pp.

Charlap, L.S., Rees, H.D. and Robbins, D.P., 1990, The asymptotic probability that a random biased matrix is invertible, *Discrete Mathematics*, 82(2):153 – 163pp.

Chekuri, C., Fragouli, C. and Soljanin, E., 2006, On Average Throughput and Alphabet Size in Network Coding, *Information Theory, IEEE Transactions on*, 52(6):2410–2424pp.

Chen, J. and Shum, K.W., 2013, Repairing multiple failures in the Suh-Ramchandran regenerating codes, In Information Theory Proceedings (ISIT), IEEE International Symposium on, Istanbul, Turkey, 1441-1445pp.

Cho, B. and Aguilera, M.K., 2012, Surviving Congestion in Geo-distributed Storage Systems, In Proceedings of the 2012 USENIX Conference on Annual Technical Conference, Berkeley, CA, USA, 40-40pp.

Chuang, J.C. and Sirbu, M.A., 2001, Pricing multicast communication: A cost-based approach, *Telecommunication Systems*, 17(3):281–297pp.

Chung, F. and Lu, L., 2003, The Average Distance in a Random Graph with Given Expected Degrees, *Internet Math., Vol*, 1(1):91–114pp.

Cleversafe, "Why RAID is Dead for Big Data Storage", https://www.youtube.com/watch?v=x54s9cjMjPU (Erişim tarihi: 28 Eylül 2015).

Datta, A. and Oggier, F., 2013, An Overview of Codes Tailor-made for Better Repairability in Networked Distributed Storage Systems, *SIGACT News*, 44(1):89–105pp.

Dickson, L.E., 1901, Linear groups, with an exposition of the Galois field theory, Leipzig B.G. tuebner, 334p.

Dijkstra, E.W., 1959, A Note on Two Problems in Connexion with Graphs, *Numerische Mathematik*, 1(1):269–271pp.

KAYNAKLAR DİZİNİ (devam)

Dimakis, A.G., Godfrey, P.B., Wainwright, M.J. and Ramchandran, K., 2007, Network Coding for Distributed Storage Systems, In INFOCOM, Proceedings IEEE, Anchorage, AK, USA, 2000-2008pp.

Dolagh, S.V. and Moazzami, D., 2011, New approximation algorithm for minimum steiner tree problem, *International Mathematical Forum*, 6(53):2625–2636pp.

Dropbox Inc, "Dropbox File Hosting Service", https://www.dropbox.com (Erişim tarihi: 28 Eylül 2015).

Duminuco, A. and Biersack, E., 2008, Hierarchical Codes: How to Make Erasure Codes Attractive for Peer-to-Peer Storage Systems, In Peer-to-Peer Computing P2P '08. Eighth International Conference on, Aachen, Germany, 89-98pp.

Floyd, S. and Henderson, T., 1999, The NewReno Modification to TCP's Fast Recovery Algorithm, RFC Editor,16p.

Gabidulin, E.M., 1985, Theory of Codes with Maximum Rank Distance, *Problems of Information Transmission (English translation of Problemy Peredachi Informatsii)*, 21(1):1–12pp.

Gantz, J.F. and Reinsel, D., 2012, The Digital Universe in 2020: big data, bigger digital shadows, and biggest growth in the far east, In IDC iView, 16p.

Gastón, B., Pujol, J. and Villanueva, M., 2011, Quasi-cyclic Minimum Storage Regenerating Codes for Distributed Data Compression, In Data Compression Conference (DCC), Snowbird, UT, USA, 33-42pp.

Gerami, M., Xiao, M. and Skoglund, M., 2011, Optimal-cost repair in multi-hop distributed storage systems, In Information Theory Proceedings (ISIT), IEEE International Symposium on, St. Petersburg, Russia, 1437-1441pp.

Ghemawat, S., Gobioff, H. and Leung, S.T., 2003, The Google File System, *SIGOPS Operating Systems Review*, 37(5):29–43pp.

Google, "Google Drive ", https://drive.google.com/drive (Erişim tarihi: 28 Eylül 2015).

Gopalan, P., Huang, C., Simitci, H. and Yekhanin, S., 2012, On the Locality of Codeword Symbols, *Information Theory, IEEE Transactions on*, 58(11):6925–6934pp.

Goss, D., 1998, Basic Structures of Function Field Arithmetic, Springer-Verlag Berlin Heidelberg, Berlin, 424p.

KAYNAKLAR DİZİNİ (devam)

Gupta, A. and Awasthi, L.K., 2011, Peer-to-Peer Networks and Computation: Current Trends and Future Perspectives, *Computing and Informatics*, 30(3):559–594pp.

Haytaoglu, E. and Dalkilic, M.E., 2013a, Homomorphic Minimum Bandwidth Repairing Codes, In Information Sciences and Systems (Lecture Notes in Electrical Engineering), Springer International Publishing, Paris, France, 339-348pp.

Haytaoglu, E. and Dalkilic, M.E., 2013b, On the Communication Cost of Distributed Storage Systems Using MDS Erasure Codes, In Proceedings of the 3rd International Symposium on Computing in Science & Engineering, Kuşadası, Turkey, 102–107pp.

Haytaoglu, E. and Dalkilic, M.E., 2015, Managing TCP connections of file reconstruction process in erasure codes, In Signal Processing and Communications Applications Conference (SIU), 2015 23th, Malatya, Turkey, 1905-1908pp.

Hougardy, S. and Prömel, H.J., 1999, A 1.598 Approximation Algorithm for the Steiner Problem in Graphs, In Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, Maryland, USA, 448–453pp.

Hu, Y., Lee, P.P.C. and Shum, K.W., 2013, Analysis and construction of functional regenerating codes with uncoded repair for distributed storage systems, In INFOCOM, Proceedings, IEEE, Turin, Italy, 2355-2363pp.

Huang, C., Chen, M. and Li, J., 2007, Pyramid Codes: Flexible Schemes to Trade Space for Access Efficiency in Reliable Data Storage Systems, In Network Computing and Applications, Sixth IEEE International Symposium on, 79-86pp.

Huang, C., Simitci, H., Xu, Y., Ogus, A., Calder, B., Gopalan, P., Li, J. and Yekhanin, S., 2012, Erasure coding in windows azure storage, In Proceedings of the USENIX conference on Annual Technical Conference, USENIX Association, Berkeley, CA, USA, 2–2pp.

Huang, J., Qin, X., Zhang, F., Ku, W. and Xie, C., 2014, MFTS: A Multilevel Fault-tolerant Archiving Storage with Optimized Maintenance Bandwidth, *Dependable and Secure Computing, IEEE Transactions on*, 11(6):524–537pp.

KAYNAKLAR DİZİNİ (devam)

Kamath, G.M., Silberstein, N., Prakash, N., Rawat, A.S., Lalitha, V., Koyluoglu, O.O., Kumar, P.V. and Vishwanath, S., 2013, Explicit MBR all-symbol locality codes, In Information Theory Proceedings (ISIT), IEEE International Symposium on, Istanbul, Turkey, 504-508pp.

Karp, R.M., 1972, Reducibility among combinatorial problems, In Complex. of Computer Computat., Springer US, 85-103pp.

Karpinski, M. and Zelikovsky, A., 1995, New approximation algorithms for the steiner tree problems, *Journal of Combinatorial Optimization*, 1:47–65pp.

Karpovsky, M.G., Stankovic, R.S. and Astola, J.T., 2008, Spectral Logic and Its Applications for the Design of Digital Devices, John Wiley & Sons Inc, Hoboken, New Jersey, USA, 598p.

Krevat, E., Vasudevan, V., Phanishayee, A., Andersen, D.G., Ganger, G.R., Gibson, G.A. and Seshan, S., 2007, On Application-level Approaches to Avoiding TCP Throughput Collapse in Cluster-based Storage Systems, In Proceedings of the 2nd International Workshop on Petascale Data Storage: Held in Conjunction with Supercomputing '07, ACM, Reno, Nevada, USA, 1–4pp.

Kubiatowicz, J., Bindel, D., Chen, Y., Czerwinski, S., Eaton, P., Geels, D., Gummadi, R., Rhea, S., Weatherspoon, H., Weimer, W., Wells, C. and Zhao, B., 2000, OceanStore: an architecture for global-scale persistent storage, *SIGPLAN Not., ACM*, 35(11):190–201pp.

Kulkarni, S. and Agrawal, P., 2011, A Probabilistic Approach to Address TCP Incast in Data Center Networks, In Distributed Computing Systems Workshops, 31st International Conference on, Minneapolis, MN, USA, 26-33pp.

Kurose, J.F. and Ross, K.W., 2005, Computer Networking: A Top Down Approach Featuring the Internet (3rd edition), Addison Wesley, Reading, MA, 821p.

Lee, H.C., Park, J.E. and Lee, M.J., 2013, C3ware: A Middleware Supporting Collaborative Services over Cloud Storage, *The Computer Journal*.

Li, J., Yang, S., Wang, X. and Li, B., 2010, Tree-structured Data Regeneration in Distributed Storage Systems with Regenerating Codes, In INFOCOM, Proceedings IEEE, Charleston, SC, USA, 1-9pp.

Li, J., Yang, S., Wang, X., Xue, X. and Li, B., 2009, Tree-structured data regeneration with network coding in distributed storage systems, In Quality of Service, 2009. IWQoS.17th International Workshop, Charleston, SC, USA, 1-9pp.

KAYNAKLAR DİZİNİ (devam)

Medina, A., Matta, I. and Byers, J., 2000, On the Origin of Power Laws in Internet Topologies, *SIGCOMM Computer Communication Review*, 30(2):18–28pp.

Menezes, A.J., Oorschot, P.C.V. and Vanstone, S.A., 1996, Handbook of Applied Cryptography, CRC Press, Boca Raton, FL, USA, 810p.

ns-3, "Network simulator 3", http://www.nsnam.org/ (Erişim tarihi: 28 Eylül 2015).

Oggier, F. and Datta, A., 2011a, Self-Repairing Codes for distributed storage; A projective geometric construction, In Information Theory Workshop (ITW), IEEE, Paraty, Brazil, 30-34pp.

Oggier, F. and Datta, A., 2011b, Self-repairing homomorphic codes for distributed storage systems, In INFOCOM, Proceedings IEEE, Shanghai, China, 1215–1223pp.

Pamies-Juarez, L., Datta, A. and Oggier, F.E., 2013a, RapidRAID: Pipelined erasure codes for fast data archival in distributed storage systems, In INFOCOM, Proceedings IEEE, Turin, Italy, 1294-1302pp.

Pamies-Juarez, L., Hollmann, H.D.L. and Oggier, F.E., 2013b, Locally repairable codes with multiple repair alternatives, In Information Theory Proceedings (ISIT), IEEE International Symposium on, Istanbul, Turkey, 892-896pp.

Pamies-Juarez, L., Oggier, F.E. and Datta, A., 2012, An Empirical Study of the Repair Performance of Novel Coding Schemes for Networked Distributed Storage Systems, *CoRR*, abs/1206.2187.

Papailiopoulos, D.S. and Dimakis, A.G., 2012, Locally Repairable Codes, In Information Theory Proceedings (ISIT), IEEE International Symposium on, Cambridge, MA, USA, 2771 - 2775pp.

Phanishayee, A., Krevat, E., Vasudevan, V., Andersen, D.G., Ganger, G.R., Gibson, G.A. and Seshan, S., 2008, Measurement and Analysis of TCP Throughput Collapse in Cluster-based Storage Systems, In Proceedings of the 6th USENIX Conference on File and Storage Technologies, San Jose, California,USA, FAST'08, 1–14pp.

Plank, J.S., 1997, A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems, *Software Practice Experience*, 27(9):995–1012pp.

KAYNAKLAR DİZİNİ (devam)

Plank, J.S., Luo, J., Schuman, C.D., Xu, L. and Wilcox-O'Hearn, Z., 2009, A Performance Evaluation and Examination of Open-source Erasure Coding Libraries for Storage, In Proccedings of the 7th Conference on File and Storage Technologies, USENIX Association, San Francisco, California, USA, 253–265pp.

Plank, J.S., Simmerman, S. and Schuman, C.D., 2008, Jerasure: A Library in C/C++ Facilitating Erasure Coding for Storage Applications - Version 1.2, Technical Report, Department of Electrical Engineering and Computer Science University of Tennessee, Knoxville, USA, 39p.

Rabin, M.O., 1989, Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance, *J. ACM*, 36(2):335–348pp.

Rashmi, K.V., Shah, N.B., Gu, D., Kuang, H., Borthakur, D. and Ramchandran, K., 2013a, A Solution to the Network Challenges of Data Recovery in Erasure-coded Distributed Storage Systems: A Study on the Facebook Warehouse Cluster, In Proceedings of the 5th USENIX Conference on Hot Topics in Storage and File Systems, USENIX Association, Berkeley, CA, USA, 8–8pp.

Rashmi, K.V., Shah, N.B., Gu, D., Kuang, H., Borthakur, D. and Ramchandran, K., 2014, A "Hitchhiker's" Guide to Fast and Efficient Data Reconstruction in Erasure-coded Data Centers, *SIGCOMM Computer Communication Review*, 44(4):331–342pp.

Rashmi, K.V., Shah, N.B. and Kumar, P.V., 2011a, Enabling node repair in any erasure code for distributed storage, In Information Theory Proceedings (ISIT), IEEE International Symposium on, St. Petersburg, Russia, 1235-1239pp.

Rashmi, K.V., Shah, N.B. and Kumar, P.V., 2011b, Optimal Exact-Regenerating Codes for Distributed Storage at the MSR and MBR points via Product-Matrix Construction, *Information Theory, IEEE Transactions on*, 57(8):5227–5239pp.

Rashmi, K.V., Shah, N.B. and Ramchandran, K., 2013b, A piggybacking design framework for read-and download-efficient distributed storage codes, In Information Theory Proceedings (ISIT), IEEE International Symposium on, Istanbul, Turkey, 331-335pp.

KAYNAKLAR DİZİNİ (devam)

Rawat, A.S. and Vishwanath, S., 2012, On locality in distributed storage systems, In Information Theory Workshop (ITW), IEEE, Lausanne, Switzerland, 497-501pp.

Reed, I.S. and Solomon, G., 1960, Polynomial Codes Over Certain Finite Fields, *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304pp.

Shpiner, A., Keslassy, I., Bracha, G., Dagan, E., Iny, O. and Soha, E., 2012, A switch-based approach to throughput collapse and starvation in data centers, *Computer Networks*, 56(14):3333 – 3346pp.

Shum, K.W., 2011, Cooperative Regenerating Codes for Distributed Storage Systems, In Communications (ICC), IEEE International Conference on, Kyoto, Japan, 1-5pp.

Shvachko, K., Kuang, H., Radia, S. and Chansler, R., 2010, The Hadoop Distributed File System, In Proceedings of the IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Washington, DC, USA, 1–10pp.

Suh, C. and Ramchandran, K., 2011, Exact-Repair MDS Code Construction Using Interference Alignment, *Information Theory, IEEE Transactions on*, 57(3):1425–1442pp.

Takahashi, H. and Matsuyama, A., 1980, An Approximate Solution for the Steiner Problem in Graphs., *Math. Japonica*, 24:573 –577pp.

Tamo, I., Wang, Z. and Bruck, J., 2013, Zigzag Codes: MDS Array Codes With Optimal Rebuilding, *Information Theory, IEEE Transactions on*, 59(3):1597–1616pp.

Thomasian, A., 2005, Clustered RAID Arrays and Their Access Costs, *The Computer Journal*, 48(6):702–713pp.

Turner, L.R., 1966, Inverse of the Vandermonde matrix with applications, Nasa Technical Note, D-3547, 14p.

Turner, V., Gantz, J.F., Reinsel, D. and Minton, S., 2014, The Digital Universe of Opportunities: Rich Data and the Increasing Value of the Internet of Things, In IDC iView, 9p.

Vasudevan, V., Phanishayee, A., Shah, H., Krevat, E., Andersen, D.G., Ganger, G.R., Gibson, G.A. and Mueller, B., 2009, Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communication, *SIGCOMM Computer Communication Review*, 39(4):303–314pp.

KAYNAKLAR DİZİNİ (devam)

Vignesh, G. and Thangaraj, A., 2013, Quasi-cyclic regenerating codes for distributed storage: Existence and near-MSR examples, In Information Theory Proceedings (ISIT), IEEE International Symposium on, IEEE, Istanbul, Turkey.

Wadekar, M., 2007, Enhanced Ethernet for Data Center: Reliable, Channelized and Robust, In Local & Metropolitan Area Networks, 15th IEEE Workshop on, New York, NY, USA, 65-71pp.

Waring, E., 1779, Problems concerning interpolations, *Philos. Trans. Roy. Soc. London Ser. A*, 69:59–67pp.

Waxman, B.M., 1988, Routing of multipoint connections, *Selected Areas in Communications, IEEE Journal on*, 6(9):1617–1622pp.

Wikipedia, "Apache Hadoop", https://en.wikipedia.org/wiki /Apache_Hadoop (Erişim tarihi: 28 Eylül 2015).

Wu, H., Feng, Z., Guo, C. and Zhang, Y., 2013, ICTCP: Incast Congestion Control for TCP in Data-Center Networks, *Networking, IEEE/ACM Transactions on*, 21(2):345–358pp.

ÖZGEÇMİŞ

Elif Haytaoğlu

Adres: Uluslararası Bilgisayar Ensitüsü 35100 Bornova/İZMİR
Telefon: (+90) 5077672786
email:elifacar0@gmail.com

Kişisel Bilgiler
Milliyeti: Türkiye Cumhuriyeti
Doğum Yeri ve Tarihi: Denizli, 25.11.1986

Eğitim Durumu
Doktora : 2008 - , Ege Üniversitesi, Uluslarası Bilgisayar Enstitüsü (Not Ort: 3.63)
Lisans: 2004-2008, Pamukkale Üniversitesi, Bilgisayar Mühendisliği (Fakülte Birinciliği - Not Ort: 3.64)
Lise: 2000-2004, Denizli Anadolu Lisesi

Yabancı Dil
Türkçe : Anadil
İngilizce : İyi derecede (93.75, 2014 YDS)
Almanca: Başlangıç

Bilgisayar Dilleri
C/C++, Java, Matlab, C# , LaTeX

Çalışma Tecrübesi
Haziran 2007 - Ağustos 2008, Yazılım Geliştiricisi, Bilgi İşlem Dairesi, Pamukkale Üniversitesi, Denizli, Türkiye
Haziran 2009, Araştırma Görevlisi, Uluslararası Bilgisayar Enstitüsü, Ege Üniversitesi, İzmir, Turkey.

Projeler
2014, Dağıtık Depolama Sistemleri İçin Topoloji Farkındalıklı Çözümler Üzerine Bir Çalışma, 05-DPT-003/35.
2010, HP Innovations in Education Worldwide.

Yayınlar

Haytaoglu, E. and Dalkilic, M. E., 2015, On the Communication Cost of MDS Erasure Codes in Distributed Storage Systems, *Computing and Informatics*. (Accepted.)

Haytaoglu, E. and Dalkilic, M. E., 2015, Managing TCP connections of file reconstruction process in erasure codes, in 23th Signal Processing and Communications Applications Conference (SIU), Malatya, Turkey, 1905–1908 pp.

Karaoglan, B., Candemir, C., Haytaoglu, E., Algin, G. B., Demirci, S., 2014, Using Twitter as a Diagnostic Teaching and Learning Assessment Tool", in 25th European Association for Education in Electrical and Information Engineering Conference, IEEE, Cesme, Turkey, 73–76 pp.

Haytaoglu, E. and Dalkilic, M. E., 2013, Homomorphic Minimum Bandwidth Repairing Codes, in 28th International Symposium on Computer and Information Sciences, Paris, France, 339–348 pp.

Haytaoglu, E. and Dalkilic, M. E., 2013, On the Communication Cost of Distributed Storage Systems Using MDS Erasure Codes, Proc. of 3rd. International Symp. on Computing in Science and Engineering (ISCSE) , Kusadası, Turkey, 102-107 pp.

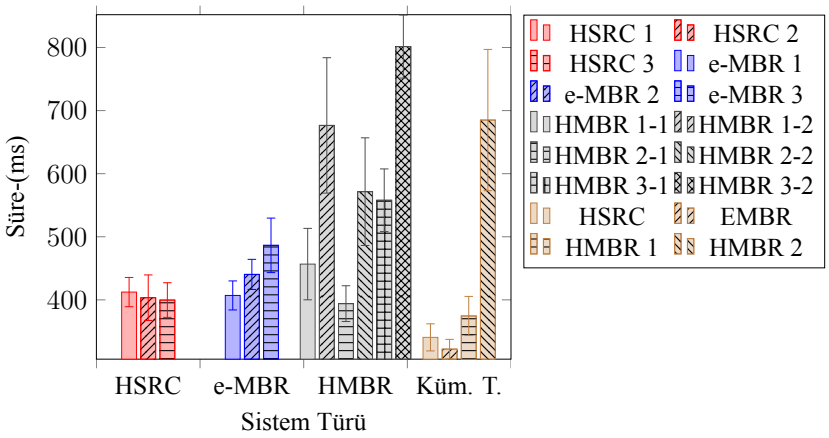
Challenger, M., Haytaoglu, E., Tokatli, G., Dagdeviren, O. and Erciyes, K., 2013, A Hybrid Distributed Mutual Exclusion Algorithm for Cluster-Based Systems, in *Speacial Issue on Distributed Control and Estimation of Networked Agent Systems, Mathematical Problems in Engineering,* , Volume 2013

Dalkilic, M. E., Acar, E. and Tokatlı, G., 2011, A simple shuffle-based stable inplace merge algorithm, in Procedia Computer Science, Volume 3, 2011, 1049-1054 pp.

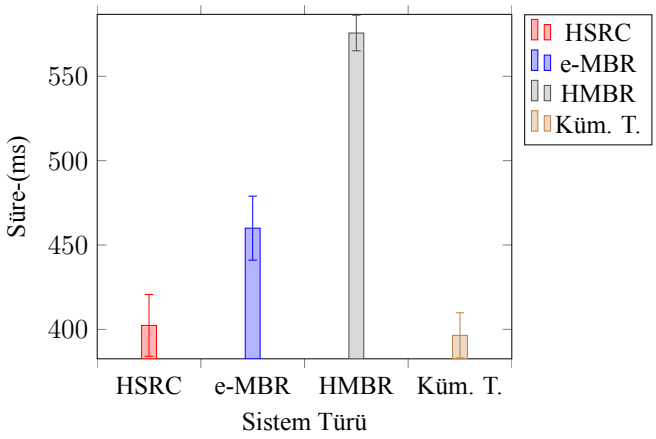
EKLER

Ek 1 Kümeleme Tabanlı Dağıtık Depolama Sistemi ile İlgili Şekiller

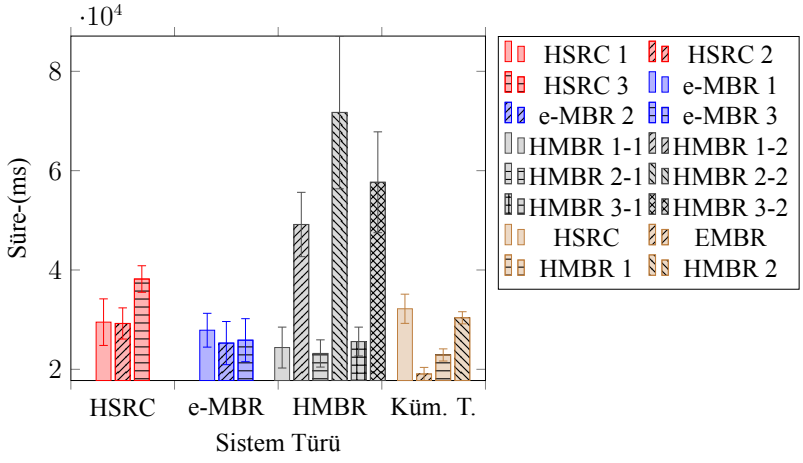
Ek 1.1 10 MB dosya tamiri için geçen süre



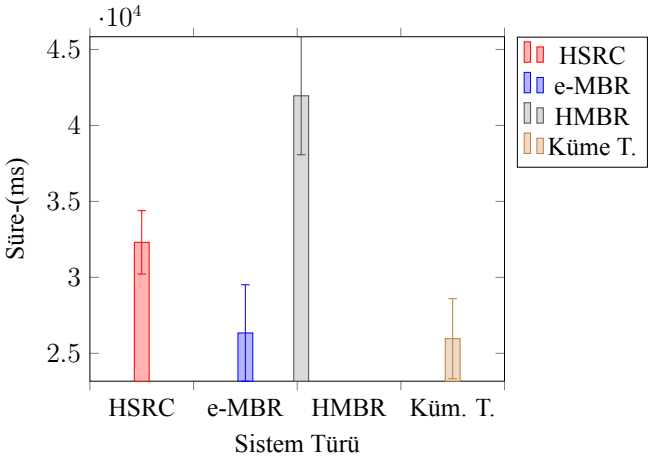
Ek 1.2 10 MB dosya tamiri için geçen sürelerin ortalaması



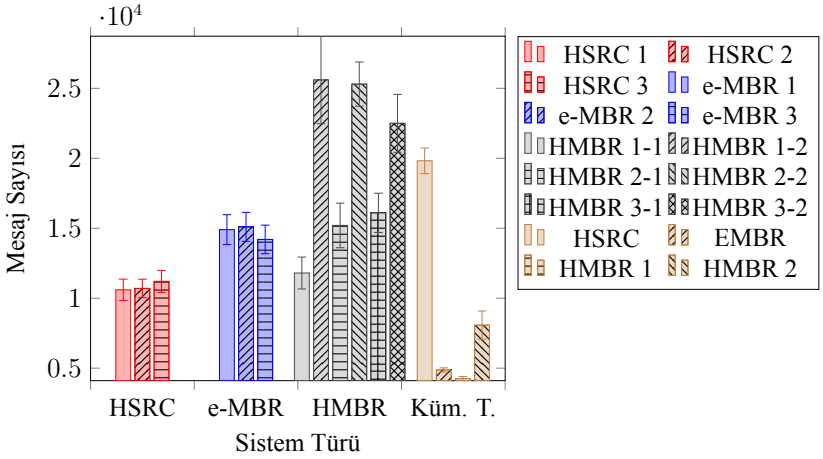
Ek 1.3 1 GB dosya tamiri için geçen süre



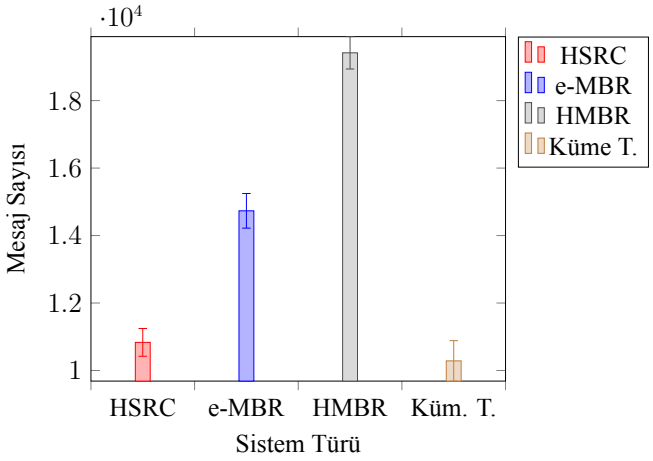
Ek 1.4 1 GB dosya tamiri için geçen sürelerin ortalaması



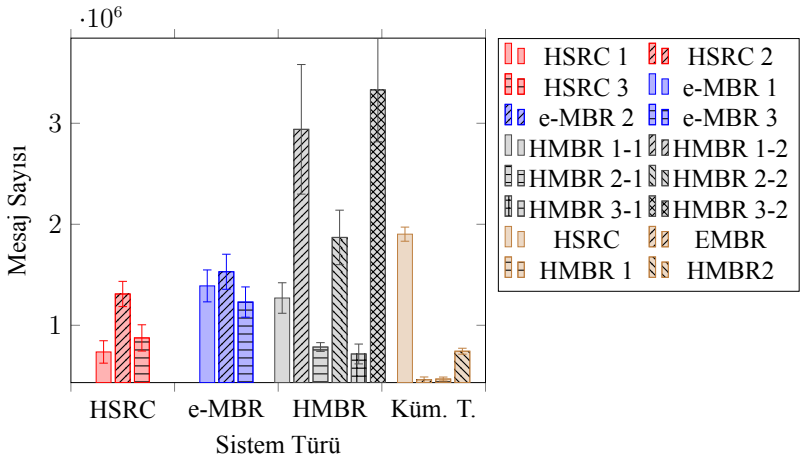
Ek 1.5 10 MB dosya tamiri için kullanılan mesaj sayısı



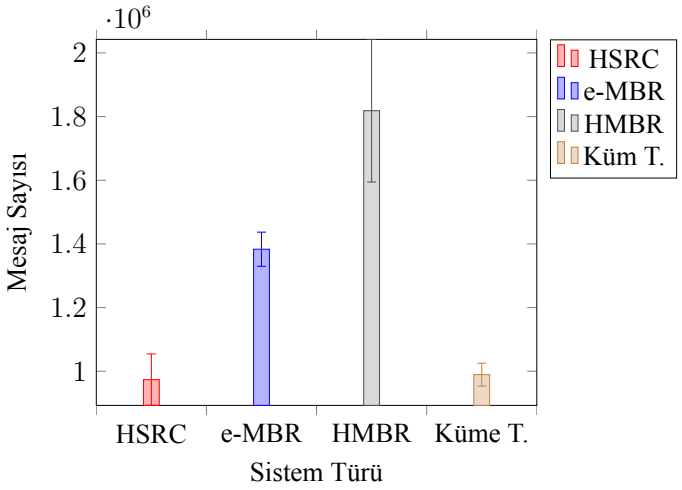
Ek 1.6 10 MB dosya tamiri için kullanılan mesajların ortalaması



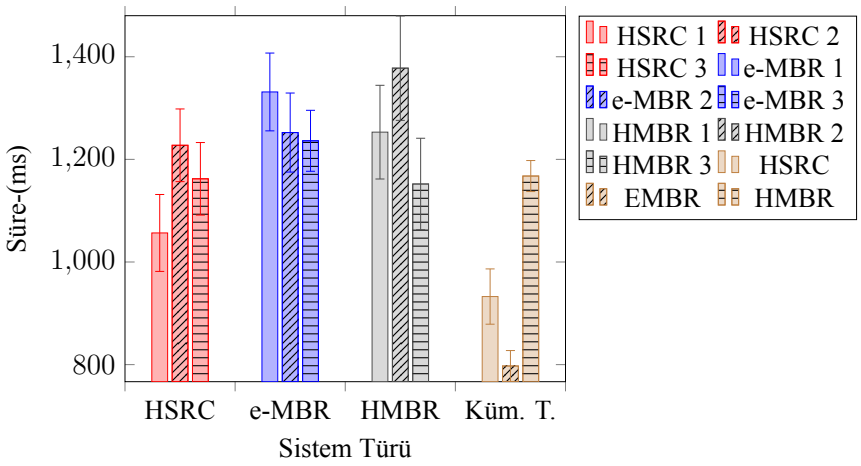
Ek 1.7 1 GB dosya tamiri için kullanılan mesaj sayısı



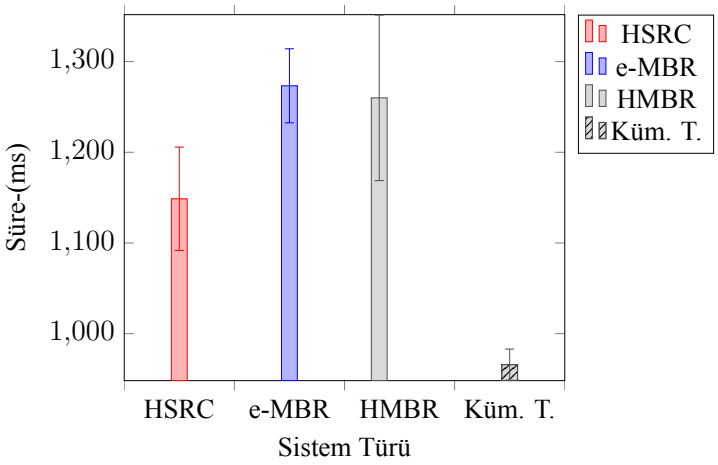
Ek 1.8 1 GB dosya tamiri için kullanılan mesajların ortalaması



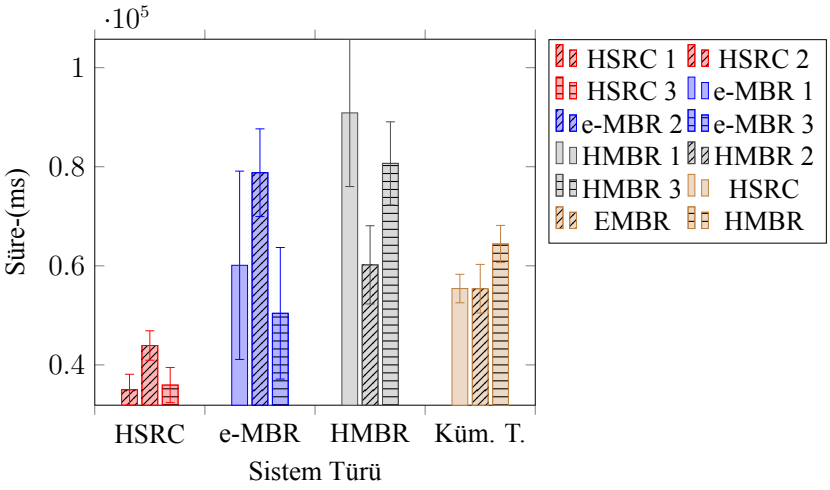
Ek 1.9 10 MB dosyanın geri çatımı için kullanılan süreler



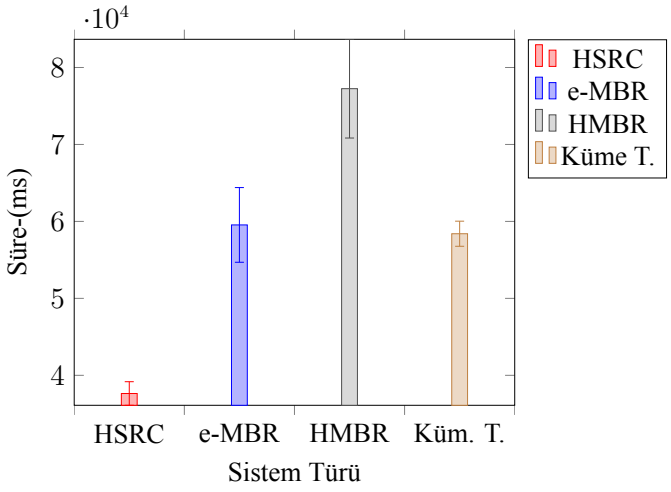
Ek 1.10 10 MB dosyanın geri çatımı için kullanılan sürelerin ortalaması



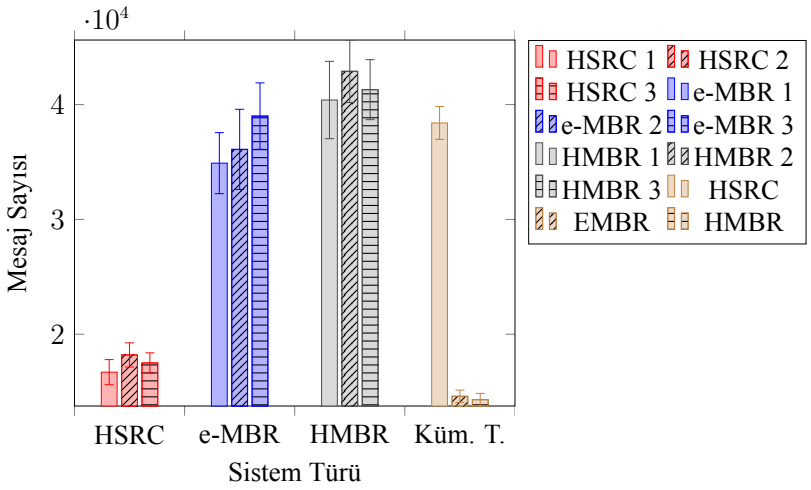
Ek 1.11 1 GB dosyanın geri çatımı için kullanılan süreler



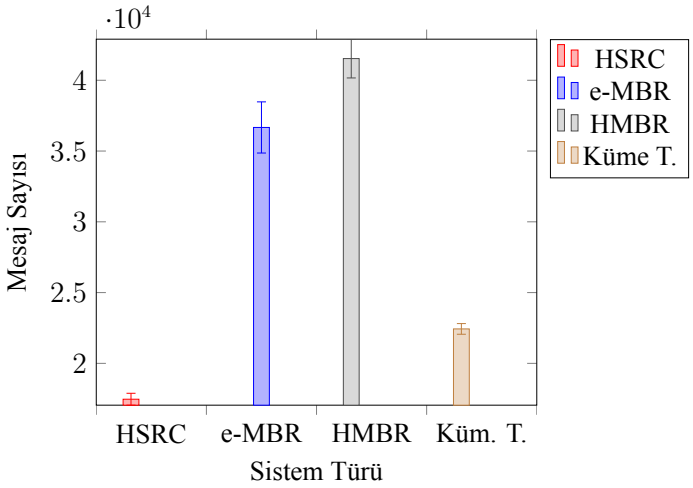
Ek 1.12 1 GB dosyanın geri çatımı için kullanılan sürelerin ortalaması



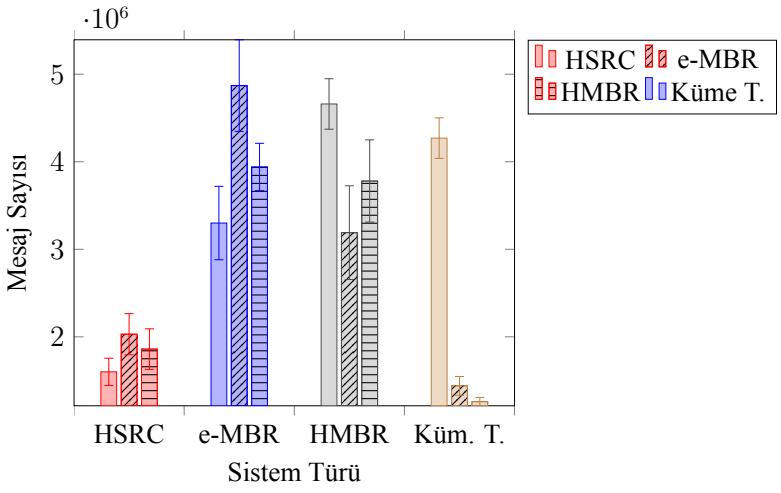
Ek 1.13 10 MB dosyanın geri çatımı için kullanılan mesajlar



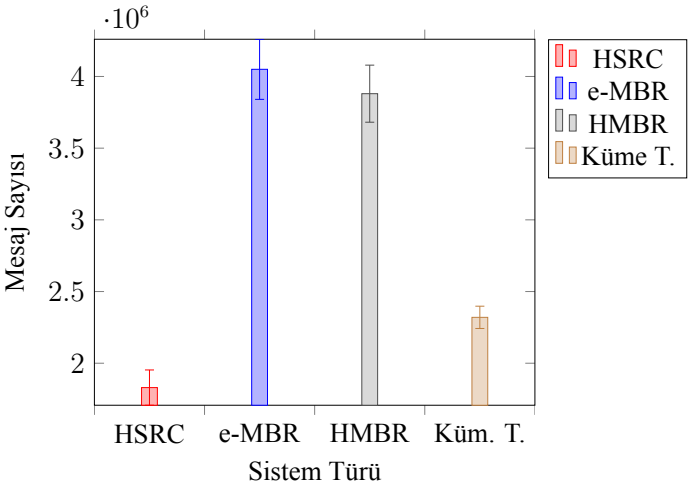
Ek 1.14 10 MB dosyanın geri çatımı için kullanılan mesajların ortalaması



Ek 1.15 1 GB dosyanın geri çatımı için kullanılan mesajlar



Ek 1.16 1 GB dosyanın geri çatımı için kullanılan mesajların ortalaması



Ek 2 Türkçe-İngilizce Terimler Sözlüğü

ağ anahtarı	switch
aradeğerleme	interpolation
belirteç	determinant
bütünlük kontrolü	integrity checking
cisim	field
çoğa gönderim	multicast
çoklu teke gönderim	multi unicast
dayanıklılık	reliability
devrik	transpose
düğüm tamiri	node repair
eş dağılımlı	uniform
eşlik	parity
hata dayanıklılığı	fault tolerant
homomorfik	homomorphic
homomorfizm	homomorphism
ilkel eleman	primitive element
izdüşümsel	projective
lime	strip
melez	hybrid
neredeyse çevrimsel	quasi-cyclic
özet	hash
şerit	stripe
teke gönderim	unicast
verinin geri çatılması	data reconstruction
veri-toplayıcı	data-collector
yeni-gelen	newcomer
yenileme kodları	regenerating codes

Ek 3 Algoritma 1

Algoritma 1 *Veri ve Veri Toplayıcı* düğümlerinin çalıştırdığı TCP bağlantı kontrolü algoritması.

Input: *AtaAdresi*, *Durum*, *ÇocukK*, *TorunK*, *TCPHaritası*,k,n *AkışK* $\leftarrow \emptyset$; *UyarılanK* $\leftarrow \emptyset$; *bitti* $\leftarrow$ yanlış

while ¬bitti **do**
  mesaj $\leftarrow$ gelen mesajı oku
  switch (*mesaj.tipi*)
  case LİME**:**
    AkışK $\leftarrow$ *AkışK* $\cup$ $\{$*mesaj.ilkAdres,mesaj.limeNo*$\}$
    if *Durum*= *Veri Toplayıcı* **then**
      f_1(*mesaj*,k,$n-k$)
    else
      soket $\leftarrow$ *TCPHaritası*'ından *mesaj.ilkAdres* için kullanılan soketi bul
      mesaj'ı *soket* üzerinden gönder
    end if
  case PAKET İPTALİ**:**
    ort$\leftarrow$ f_2(*AkışK*); *sapma* $\leftarrow$ f_3(*AkışK*)
    boyut$\leftarrow$ *AkışK*'nın eleman sayısı
    for each $z \in$ AkışK **do**
      $c \leftarrow$ z.ilkAdres'in paketini ileten çocuk
      if z.limeNo>(ort+$\frac{sapma}{\sqrt{boyut}}$) & $c \notin$ *UyarılanK* **then**
        c'ye *YAVAŞLA* mesajını gönder
        UyarılanK $\leftarrow$ *UyarılanK* $\cup$ c
      end if
      zamanlayıcıyı (*timer*) başlat
    end for
  case ZAMAN AŞIMI**:**
    UyarılanK $\leftarrow \emptyset$
  case YAVAŞLA**:**
    sSoket=f_4(*TCPHaritası, AkışK, ÇocukK,TorunK*)
    f_5(*sSoket*, *TCPHaritası*)
  case SON**:**
    bitti $\leftarrow$ doğru
  end switch
end while