ANKARA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

YÜKSEK LİSANS TEZİ

GERÇEK ZAMANLI BİLGİSAYAR KONTROL SİSTEMLERİNDE İŞLEM SÜRECİ ORGANİZASYONU

Orhan Fikret DUMAN
Danışman: Yrd.Doç.Dr.Refik SAMET

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

ANKARA
2008

ÖZET

Yüksek Lisans Tezi

GERÇEK ZAMANLI BİLGİSAYAR KONTROL SİSTEMLERİNDE İŞLEM SÜRECİ ORGANİZASYONU

Orhan Fikret DUMAN

Ankara Üniversitesi

Fen Bilimleri Enstitüsü

Bilgisayar Mühendisliği Anabilim Dalı

Danışman: Yrd. Doç. Dr. Refik SAMET

Aviyonik sistemler veya nükleer reaktörler gibi katı kısıtların olduğu gerçek zamanlı ve arıza kaldırılabilirlik özelliğine ihtiyaç duyan sistemler için girdiler gerçek dünyadan algılayıcılar vasıtasıyla elde edilmektedir. Böyle bir durumda, bir görevin algılayıcılardan gelen farklı durumlara yönelik ihtiyaçları karşılamak üzere her bir periyodunda birbirinden farklı kapasitelere ihtiyaç duyması olasıdır. Aynı zamanda bahsedilen kritik sistemler için, arıza kaldırılabilirliği sağlamak üzere, organize edilebilirliği bozmayacak şekilde dengeli bir kapasite artırımına ihtiyaç duyulmaktadır. Araştırmaların çoğunda görev kümelerine yönelik fizibilite testleri sadece belirli bir işlem süreci organizatörü veya geliş modeli ile sınırlandırılmakta ve sadece belirlenen durum test edilebilmekte iken, tez kapsamında yapılan incelemeler farklı önceliklendirme temelli işlem süreci organizatörleri ve değişik geliş modelleri ile gerçekleştirilmektedir. Bu tez çalışmasında, öncelikle kapasite güncelleme işlemini analiz edebilmek maksadıyla, kapasite güncellemesine ihtiyaç duyan bir görev kümesi, kapasite güncellemesinde kullanılacak kapasite matrisi ve değişik varış modellerine sahip bir sistem tasarlanmakta ve test edilmektedir. Değişken kapasite kullanımının, kapasite artırımını sağlayabilecek özellikte olmasına karşın, kaynak kullanımı ile ilgili herhangi bir sınırlayıcı etkiye sahip olmaması nedeniyle, belirli zamanlarda var olanın üzerinde oluşabilecek kaynak talepleri yüzünden sistemin organize edilemez hale gelmesine sebebiyet verebildiği sonucuna varılmaktadır. Yapılan test ve analiz sonuçları doğrultusunda, arıza kaldırılabilirlik özelliğini sağlayan ve organize edilebilirlik oranını düşürmeyen, tam kullanım seviyesinin altında sistem kullanım oranına sahip bir görev kümesini kapasite artırma-azaltma yöntemiyle azami seviyede bir sistem kullanım oranına ulaştıran "Sınırlandırılmış Kaynak Kullanımı ile Dengeli Sistem" adı verilen bir organizasyon şeması önerilmektedir.

Mart 2008, 72 sayfa

Anahtar Kelimeler: İşlem süreci organizasyonu, kapasite güncelleme, görev geliş modeli, gerçek zamanlı sistemler, arıza kaldırılabilirlik

ABSTRACT

Master Thesis

SCHEDULING FOR REAL TIME COMPUTER CONTROL SYSTEMS

Orhan Fikret DUMAN

Ankara University

Graduate School of Natural and Applied Sciences

Department of Computer Engineering

Supervisor: Asst. Prof. Dr. Refik SAMET

In hard real time environments like avionic systems or nuclear reactors which need the fault tolerance issue, the input to the system are obtained from real world by using sensors. And it is highly possible for a task to have different capacity needs for each period according to these changing conditions of real world. For these mentioned critical systems, it is needed to provide capacity incrementation to get fault tolerance feature while not violating the schedulability of the system. In this thesis, to be able to analyze capacity modification, a system which has task models needing random capacities in each period, a capacity matrix to be used for capacity modification and different arrival patterns is designed. Since feasibility tests for usual task models are just limited to some specific schedulers and arrival patterns, an analysis using different preemptive schedulers with different arrival patterns is made. The key results for the designed system are the ones that random capacity utilization has the ability to provide capacity incrementation, but it causes the system to degrade in schedulability because no mechanism exists to limit the source requests which can occur at the same time causing transient overloads. According to the results of these tests and analysis, a steady scheduling scheme named "Steady Scheme with Bounded Utilization", which provides assignment of more capacity values to the tasks than they originally have by an extend-shrink mechanism and providing a better utilization of the resources besides a beter schedulability rate, is suggested.

March 2008, 72 pages

Key Words: Scheduling, capacity modification, task arrival patterns, real time systems, fault tolerance

TEŞEKKÜR

Çalışmalarımı yönlendiren, araştırmalarımın her aşamasında bilgi, öneri ve yardımlarını esirgemeyerek akademik ortamda olduğu kadar beşeri ilişkilerde de engin fikirleriyle yetişme ve gelişmeme katkıda bulunan danışman hocam Sayın Yrd. Doç. Dr. Refik SAMET'e, çalışmalarım süresince manevi desteklerini esirgemeyen değerli bölüm başkanımız sayın Prof. Dr. Baki KOYUNCU'ya, çalışma arkadaşlarım Ersin ALAN ve Adnan GÜRBÜZ'e teşekkürü borç bilirim.

Ayrıca, hayatımın her döneminde sevgi ve desteklerini esirgemeyen ve bu günlere gelmemde büyük emeği bulunan anne ve babam ile ailemin tüm fertlerine saygı ve sonsuz şükranlarımı sunarım.

Orhan Fikret DUMAN
Ankara, Mart 2008

İÇİNDEKİLER

SİMGELER DİZİNİ

U	Utilization: Kaynak Kullanım Oranı
PR	Preemption: Önceliklendirme
CS	Context Switch: Kontekst Anahtarlama
WCET	Worst Case Execution Time: En Kötü Durumda Gerçekleştirilme Zamanı
C	Capacity: Kapasite
G	Task: Görev
T	Period: Periyot
D	Deadline: Son Tamamlanma Zamanı
RM	Rate Monotonic: Frekans Monotonik
EDF	Earliest Deadline First: En Erken Tamamlanma Zamanı İlk
LLF	Least Laxity First: En Düşük Ertelenebilirlik İlk
RC	Random Capacity: Rastgele Kapasite
SBU	Steady and Bounded Utilization: Dengeli ve Sınırlandırılmış Kaynak Kullanım Oranı
RoC	Rest of Capacity: Kalan Kapasite Miktarı
LCM	Least Common Multiple: Ortak Katların En Küçüğü
P	Hyper Period: Temel Periyot
W	Workload: İş Yükü
B	Available Time: İşlemci Tahsis Edilebilir Zamanı
CT	Current Time: Güncel Zaman
WRK	Resource Need: Kaynak İhtiyacı
SCH	Schedulability: Organize Edilebilirlik
Rand	Randomize: Rastgele Değer Atama
Abs	Absolute: Mutlak Değer
SIM	Simulation Range: Simülasyon Aralığı
ID	Idle Time: İşlemci Boş Zamanı

ŞEKİLLER DİZİNİ

ÇİZELGELER DİZİNİ

1. GİRİŞ

Gerçek zamanlı kontrol sistemlerinde bir işlemin belirlenen zaman dâhilinde gerçekleştirilmesi, gerçekleştirilen işlemin mantıksal doğruluğu kadar hayatiyet arz etmektedir. Dolayısıyla bu zaman kısıtlamasına uygun sistemler geliştirebilmek için sistem kaynaklarını kullanmak üzere tahsis edilen görevlerin en etkin ve adil şekilde planlanması gerekmektedir.

Katı zaman kısıtına sahip işlem süreci organizatörleri, aynı anda sistem kaynaklarını kullanmak üzere başvuran görevlere uygun bir şekilde kaynak tahsisi yapılabilmesini sağlayacak uygun algoritmaları kullanarak kaynakların belirlenen zaman aralığında ve herhangi bir gecikmeye sebebiyet vermeksizin ihtiyacı olan görevlere atanması işlemini gerçekleştirmeye çalışır. Bu organizatörler, kaynakların görevler tarafından paylaştırılması işlemini gerçekleştirmek üzere hangi görevin çalıştırılacağına ve yeni bir görevin sistem kaynaklarını ne zaman kullanmaya başlayacağına karar verirler.

Sistemin kritikliğine bağlı olarak, gerçek zamanlı olarak çalışan sistemlerin ölümcül sonuçlara yol açmaması maksadıyla arıza kaldırılabilirliğin bu sistemler için değerlendirilmesi gerekmektedir. Felaketlere sebebiyet verilmemesi amacıyla arıza kaldırılabilirlik özelliği kritik sistemler için muhakkak dikkate alınması gereken bir unsurdur. Katı gerçek zamanlı sistemlerin kritik yapısı sebebiyle hataların tolere edilmesi bir zorunluluktur. Bu nedenle gerçek zamanlı sistemler üzerindeki işlem süreci organizasyonu algoritmalarına arıza kaldırılabilirlik özelliğinin kazandırılabilmesi maksadıyla her bir görevin gerçekleştirilme zamanını ifade eden kapasite değerinin güncellenebilmesi ve ihtiyaç duyulan kapasite değerinin üzerinde sistem kaynağı tahsis edilerek fazladan atanan zaman diliminde görevin yedeklenebilmesi amaçlanmaktadır.

İşlem süreci organizasyonu için bir diğer önemli husus olarak sistem kaynaklarının mümkün olan maksimum kullanım oranıyla ve etkin bir şekilde paylaştırılmasının amaçlandığı bir sistemde, tasarlanan algoritmanın yüksek seviyede kullanım oranı sağlaması ve işlemci boş kalma zamanını minimize edebilmesi gerekmektedir. Ayrıca, görevlere sistem içerisinde kaynak tahsisi yapılıp yapılamayacağına karar veren “kabul

kontrol" işlemi, görevlerin geç tamamlanma ya da kritik görevlerin hiç gerçekleştirilememesi durumuna sebebiyet vermeyecek şekilde çok iyi planlanmalıdır.

Bahsedilen hedeflerin gerçekleştirilebilmesini sağlayabilecek bir sistemin tasarlanması için farklı bir takım yaklaşımlar geliştirilmiştir. Gerçek zamanlı işlem süreci organizasyonunu periyodik, sporadik veya poisson gibi farklı geliş modelleri doğrultusunda gerçekleştirmek üzere geliştirilmiş teknikler mevcuttur. Bütün bu teknikler belirli bir takım özel problemlere cevap olmak üzere tasarlanmış olmaları sebebiyle farklı değerlendirme kriterlerine ve durumlara göre diğer tekniklere üstünlük sağlar. Dolayısıyla, sistem tasarımı esnasında ihtiyaçlar detaylı olarak analiz edilerek bu ihtiyaçlara cevap verebilecek uygun işlem süreci organizasyonu algoritmalarının ve organizatörlerinin kullanılması gerekmektedir.

Etkin bir işlem süreci organizasyon şeması tasarımının Çizelge 1.1'de belirtilen hususlar doğrultusunda gerçekleştirilmesi gerekmektedir:

Çizelge 1.1 İşlem Süreci Organizasyonu tasarım çizelgesi

Sistem Durumu	Gerçek Zamanlı, Önceliklendirme Temelli, Arıza Kaldırılabilir
Sistem Parametreleri	Kontekst Anahtarlama Sayısı, Önceliklendirme Sayısı, Kaynak Kullanım Oranı, En Kötü Durumda Gerçekleştirilme Zamanı
Görev Modelleri	Görev Sayısı, Görev Parametreleri; $(G_{i,k} = (C_i, D_i, T_i))$ (Kapasite, Son Gerçekleştirme Zamanı, Periyot)
Görev Geliş Modelleri	Periyodik, Sporadik, Poisson
Önceliklendirme Algoritmaları	Statik: RM, Dinamik: EDF, LLF

Bu tez çalışmasında, işlem süreci organizasyonu algoritmaları, her periyotta farklı kapasite değerlerine sahip olacak şekilde kapasiteleri güncellenen görev kümeleri ve farklı geliş modelleri için sergileyeceği davranış biçimlerini tespit etmek üzere tasarlanan sistem doğrultusunda analiz edilmiş ve var olan algoritmaların üstünlük ve eksiklikleri tespit edilerek eksikliklerin giderilmesine yönelik olarak yeni bir işlem süreci organizasyonu şeması önerilmiştir. Önerilen şema tek işlemcili bir sisteme arıza kaldırılabilirlik özelliğinin ilave edilebilmesi açısından önem arz etmektedir.

Konuyla ilgili faydalanılan kaynaklar ve özetleri aşağıda belirtilmiştir:

Lauzac et al. (2003), makalelerinde işlem süreci organizasyon metotlarından birisi olan RM İşlem Süreci Organizasyonu metodunun safhalarından Kabul Edilebilirlik Kontrolü süreci üzerinde gerçekleştirilen iyileştirmelerden ve uygulamalarından bahsetmektedir. Çalışmada ”Rbound” olarak adlandırılan yeni bir kabul edilebilirlik kontrol algoritması önerilmektedir. Önerilen algoritmada, sistemde hata olması durumunda dahi yüksek işlemci kullanımı sağlanmakta ve sistemin belirtilen zaman kısıtları dâhilinde çalışmasını tamamlaması garanti edilmektedir.

Marti et al. (2001), makalelerinde gerçek zamanlı işlem süreci organizasyonu algoritmalarının özelliklerinden dolayı ortaya çıkan beklenmeyen kontrol sistem yanıtları ile mevcut algoritmaların kontrol döngüleri içerisinde ne tür zamanlama problemlerine sebebiyet verdiğinden bahsetmektedir. Problemin ortaya çıkış sebebi, gerçek zamanlı işlem süreci organizasyonu algoritmalarının neden olduğu, “jitter” olarak adlandırılan ve sistem performansında düşüşe hatta dengesizliğe sebebiyet verebilen gecikmelerin gerçek zamanlı sistemler üzerindeki zaman-kritik uygulamalar için ciddi bir sorun teşkil etmesidir. Bu sorunun önlenebilmesi için bu gecikmelerin telafi edilebilmesi ve denge analizleri önerilmiştir.

Lauzac et al. (1998), makalelerinde gerçek zamanlı işlem süreci organizasyonuna arıza kaldırılabilirlik özelliğinin eklenebilmesi için farklı sistemlerde uygulanabilir yöntemler önermişlerdir. Çok işlemcili mimari üzerinde geçici ve kalıcı hataların tolere edilebildiği gerçek zamanlı ve arıza kaldırılabilir bir algoritma önermişlerdir.

Ghosh et al. (1994), makalelerinde katı gerçek zamanlı çok işlemcili mimari üzerinde arıza kaldırılabilir işlem süreci organizasyonunun gerçekleştirilmesinden bahsetmektedir. Çok işlemcili bir mimaride arıza kaldırılabilirliğin sağlanabilmesi için işlemin birden fazla kopyasının farklı işlemciler üzerinde planlanması önerilmiştir. Böylece esas kopyanın herhangi bir hata veya arıza nedeniyle gerçekleştirilememesi durumunda yedek kopyanın çalıştırılarak işlemin belirlenen zaman diliminde bitirilebilmesi sağlanmaktadır. Algoritmaya göre farklı işlemlere ait birden fazla yedek kopyanın aynı işlemci üzerinde aynı zaman dilimi üzerine planlanması (backup overloading) ve orijinal kopyası gerçekleştirilen her yedek kopyanın sistem kaynaklarını geri bırakması (backup deallocation) sağlanarak işlemciler üzerindeki toplam kaynak kullanım oranının arttırılması mümkün olmaktadır. 3 farklı metot karşılaştırılmaktadır. Bunlar; Esas/Yedek (Primary/Backup), Yedek Donanım (Spare) ve Arıza Kaldırılabilir olmayan sistemlerdir.

Dong et al. (1998), makalelerinde gerçek zamanlı mesajlar için zaman yuvası tahsisi için bir şema önermektedirler. Esnek mesafe kısıtları ve katı frekans ihtiyaçlarının karşılanabilmesi amaçlanmaktadır. Öncelikle tek hat üzerinde mesaj kuyruğu oluşturularak işlem süreci organizasyonu gerçekleştirilmiştir. Daha sonra bu algoritmanın WDM optik çoklayıcıları ile çalıştırılabilirliği denenmiştir. Sonuçlar, gerçek zamanlı sistem üzerinde frekansın ve mesafenin artırılması ile daha yüksek planlama imkânı sağlanmasına rağmen işlem süreci organizasyonunun esnekliğinin azalması yönündedir.

Alvarez et al. (2000), makalelerinde işlem süreci organize edilebilirliği üzerinde analiz işlemi gerçekleştirerek her görevin gerçekleştirilmesi zorunlu ve seçime bağlı kısımlar olarak ayrılmasını önermişlerdir. Bu sayede gerçek zamanlı sistemler için analiz sonucunda zaman kısıtlamasına riayet edilemeyeceği değerlendirilen durumlarda her görev için zorunlu olan kısımların gerçekleştirilmesi ve seçime bağlı kısımların işlemci üzerinde planlanmayarak organize edilebilirliğin arttırılması sağlanmıştır. Önerilen metot Seçime Bağlı Hesaplama (Optional Computation) olarak adlandırılmıştır. Bu metot ile arıza kaldırılabilir sabit önceliklendirmeli görevler için kesin bir işlemci kaynakları üzerinde organize edilebilirlik işlemi gerçekleştirilmektedir. Bu analiz

sayesinde sistem tasarımcısının tüm görevlerin belirlenen zaman dahilinde gerçekleştirilip gerçekleştirilemeyeceğini bilmesi ve eğer geciken veya reddedilen görevler varsa seçime bağlı kısımların iptal edilerek organize edilebilirlik oranının yükseltilebilmesini sağlamaktadır.

Lauzac et al. (1998), makalelerinde çok işlemcili sistem üzerinde gerçekleştirilecek görevler için RM işlem süreci organizasyonunun kullanıldığı global bir organizasyon şeması önermektedirler (GRMS-Global Rate-Monotonic Scheduling). Zaman kısıtlamasının katı olduğu ve olmadığı sistemler için şema test edilmiştir. Sonuç olarak GRMS yönteminin zaman kısıtlamasının katı olmadığı sistemler için daha elverişli şekilde sonuçlar verdiği görülmüştür.

Zhang and Ferrari (1993), makalelerinde Frekans Kontrollü Statik Önceliklendirmeli (RCSP-Rate Controlled Static Priority) kuyruk disiplininden bahsetmektedir. Önerilen metot bağlantı temelli ve paket anahtarlamalı ağ üzerinde sistem çıktısı, gecikme ve kayıp durumlarıyla ilgili bir takım garantiler sağlamaktadır.

Naedele (1999), makalesinde periyodik ve periyodik olmayan görevler için hibrid yapıda gerçek zamanlı arıza kaldırılabilir işlem süreci organizasyonu algoritması önermektedir. Aktif ve Pasif Yedekleme yöntemleri algoritma ile birlikte tanımlanmıştır. Buna göre Aktif Yedekleme yönteminde aynı görev seti birden fazla işlemci üzerinde planlanmaktadır. Pasif Yedeklemede ise orijinal kopyanın yedek kopyasının yedek işlemci üzerinde bulundurulması ve esas kopyada problem olması durumunda gerçekleştirilmesidir.

Chen and Xiong (2002), makalelerinde gerçek zamanlı arıza kaldırılabilir sistem üzerinde herhangi bir hata veya arıza olmaması durumunda kaynak kullanım oranının arttırılmasını araştırmışlardır. Bu maksatla Kesin Olmayan Hesaplama(IC-Imprecise Computation) tekniği geliştirilmiştir.

Hong and Goo (2005), makalelerinde gerçekleştirme zamanı kısıtlaması bulunan durumlarda gerçek zamanlı arıza kaldırılabilir sistem tasarımını araştırmışlardır. Arıza

kaldırılabilir gerçek zamanlı işlem süreci organizasyonunda, planlaması yapılan görevler için orijinal ve yedek kopyaların, ihtiyaç kalmaması durumunda, sistem kaynaklarını yeniden iade etmesi oldukça önemlidir. Bu çalışmada garanti edilmiş birkaç sistem analiz edilmiştir. Ayrıca farklı işlemci seçim algoritmaları karşılaştırılmıştır.

Liu and Yann-Hang (2003), makalelerinde en erken teslim tarihli ilk önce (EDF-Earliest Deadline First) işlem süreci organizasyonu metodu kullanılarak paket anahtarlamalı ağlar için elverişli bir işlem süreci organizasyonu disiplini ortaya koymaya çalışmışlardır. Çerçeve temelli işlem süreci organizasyonu olan ve anahtarlama mimarisi üzerinde sabit uzunluklu hücre trafiğini temel alan EDF-RR modelini önermişlerdir.

Chen and Xiong (2002), makalelerinde kaynak geri dönüşümünün sağlandığı arıza kaldırılabilir EDF algoritmasını önermişlerdir. Bu algoritmada amaç kaynak kullanımının geliştirilmesi ve süreç çıktılarının arttırılması amaçlanmıştır. Esas/yedek yaklaşımı nedeniyle gereksiz yere kullanılan sistem kaynaklarının, ihtiyacın bitmesini müteakiben tekrar sisteme iadesi temel olarak ele alınmıştır.

Kim et al. (2000), makalelerinde gerçek zamanlı arıza kaldırılabilir işlem süreci organizasyonu için geliştirilmiş ve uygulanabilir bir yöntem önermektedirler. Gerçek zamanlı ve tek işlemcili mimaride iki adet kuyruk-temelli işlem süreci organizasyonu tekniği mevcuttur. Bunlar; En Kısa Yol (FSP - Feasable Shortest Path) ve Lineer Zaman Yakınsaması (LTH - Linear Time Heuristics) metotlarıdır. FSP algoritması üzerinde arıza kaldırılabilir işlem süreci organizasyonu gerçekleştirilirken daha fazla sayıda esas kopyanın planlanmasına olanak sağlayan bir algoritma önermişlerdir.

Tsuchiya et al. (1995), makalelerinde dağıtık gerçek zamanlı sistemler üzerinde az sayıda yedek düğümle arıza kaldırılabilir bir işlem süreci organizasyonu yapılabilmesine olanak veren bir algoritma önermişlerdir. Prosedür temel olarak eşzamanlı çalıştırılan görev kopyaları temeline dayanmaktır. Bu sayede daha az sayıda

yedek donanımla arıza kaldırılabilir bir sistem taarımının mümkün olabileceği değerlendirilmektedir.

Al-Omari et al. (2001), makalelerinde çok işlemcili gerçek zamanlı sistemler üzerinde işlem süreci organizasyonun geliştirildiği yeni bir arıza kaldırılabilir yöntem önermektedirler. Algoritmada temel olarak işlemlerin normal gerçekleştirilme oranının en kötü durum şartlarında gerçekleştirile olasılığından fazla olduğu değerlendirilerek kaynak geri dönüşümünün elverişli şekilde planlanması gerektiği değerlendirilmektedir.

Satyanarayana et al. (2001), makalelerinde dağıtık gerçek zamanlı sistemelr üzerinde arıza kaldırılabilir bir işlem süreci organizasyonu tasarımını değerlendirmişlerdir. Geliştirdikleri algoritmada görevlerin alt-görevlere bölünmesi ve al-görevlerin farklı istasyonlar üzerinde planlanması esas alınmıştır. Üç adet son tarih atama politikası tanıtılmıştır ve her düğümde bir yerel işlem süreci organizatörü kullanılması düşünülmüştür.

Peng et al. (2003), makalesinde asimetrik iletişim mimarisi üzerinde arıza kaldırılabilir işlem süreci organizasyonu için yeni bir sistem önermiştir. Temel olarak asimetrik mimaride veri alım hızının ve veri gönderim hızının aynı olmaması sebebiyle sistemde oluşabilecek tıkanıklığı önleyebilmek amacıyla veri gönderim tarafında bir kuyruklama politikası gerektiği ifade edilmiştir.

Stewart and Khosla (1992), makalelerinde algılayıcı temelli kontrol sistemleri üzerinde işlem süreci organizasyonu için önerilen temel algoritmaların bir değerlendirmesini yapmışlardır. Sabit önceliklendirmeli ve dinamik önceliklendirmeli işlem süreci organizasyon şemaları değerlendirilerek melez yapıda bir işlem süreci organizasyonu tekniği önermişlerdir.

Locke et al. (1997), makalesinde işlem süreci organizasyonu konseptinin ortaya çıkmasıyla birlikte temel organizaston şemasının zaman-çizgisi üzerine kurulduğunu ifade ederek kritik kontrol uygulamalarının offline olarak çalışan tablo temelli bir yaklaşımla ve zaman-çizgisi modeli kullanılarak gerçekleştirilmesinin analizini ortaya

koymuştur. Zaman çizgisi modelinde işlem süreci organizasyonu sabit uzunluklu (küçük döngü) zaman dilimlerine ayrılmıştır ve görevler gerçekleştirilme frekanslarına göre statik olarak organize edilmektedir.

Lehoczky, et al. (1989), makalelerinde periyodik görev kümelerinin frekans monoton işlem süreci organizasyonu algoritması kullanılarak en son tamamlanma zamanı dâhilinde organize edebilirliği ve karakterizasyonunu ortaya koymuşlardır. İlave olarak rastgele üretilen görev kümeleri için bir olasılık dağılımı şeması için stokastik bir analiz çalışması yapılmıştır. Görev kümesinin boyutları büyüdükçe hesaplanma zamanlarının öneminin azaldığı ve kaynak kullanım oranı parametresinin kırılma noktasının sabit bir değere yakınsadiği ortaya konmuştur. Uniform dağılımda görev kümelerinin organize edilebilirlik değerinin %88 olduğu tespit edilmiştir. Liu and Layland *et al*. tarafından ortaya konmuş olan en kötü durum senaryosuna karşılık olarak ortalama durum senaryosu ifade edilmiştir.

Leung and Merril (1980), makalelerinde temel periyot değerinin ve görev seti için elde edilebilecek bir büyük döngünün nasıl hesaplanacağını belirtmişlerdir. Periyodik geliş modeline sahip bir görev kümesi için benzer şekilde tekrar eden temel periyot değerinin tüm görevlerin periyot değerlerinin ortak katlarının en küçüğü olduğunu ortaya koymuşlardır.

Liu and Layland (1973), makalelerinde tek işlemcili bir sistem üzerinde çoklu programlama tekniklerinin değerlendirilerek farklı algoritmaların tasarlanması üzerinde durmuşlardır. Statik önceliklendirmeli algoritma ile garantili servis sağlayan düşük seviyedeki bir kaynak kullanım oranı elde edilebilirken dinamik önceliklendirme ile bu oranın tam kapasiteye yükseltilebileceği ele alınmış ve iki algoritmanın birleştirilmesi değerlendirilmiştir.

Baruah, et al. (1990), makalelerinde belirlenen bir zaman dahilinde bir görev kümesinde yer alan tüm görevlerin tüm aktivasyonları için ihtiyaç duyacağı toplam işlem zamanı hesaplanarak sistemin iş yükü değeri ortaya konmuştur. Bu değer her görev için temel

periyot içerisindeki aktivasyon sayısı ile kapasite değerinin çarpımlarından elde edilen değerlerin toplamıdır.

Goossens and Macq (2001), makalelerinde periyodik ve bağımsız özellikte görevlerden oluşan rastgele görev kümelerinin üretilmesi konusunda bir sistem ortaya koymaktadırlar. Görev kümesinde yer alan tüm görevlerin ortak katlarının en küçüğüyle ifade edilen temel periyot değerinin, dolayısıyla simülasyon aralığının küçültülmesi için bir takım metotlar önermektedirler. Bu metotlarda görev periyotlarını ifade eden sayılar içerisinde yer alan asal sayılardan parametre olarak yararlanılmaktadır.

Buttazzo et al. (2005), makalesinde statik bir işlem süreci organizasyonu algoritması olan RM algoritması ile dinamik işlem süreci algoritması olan EDF algoritmalarının kıyaslamasını ortaya koymaktadırlar. Bu kıyas ve değerlendirmeleri; algoritmaların kompleksliğine, çalışma zamanı giderlerine, organize edilebilirliğine, aşırı yüklenme durumundaki davranış şekillerine, gecikmelere, kaynak paylaşım durumuna, aperiyodik görevlerin ilavesi durumundaki davranış şekillerine göre gerçekleştirmektedirler. Her iki algoritmaya yönelik olarak yanlış bilinen hususları ortaya koymuşlardır.

George, et al. (1996), makalelerinde işlem süreci organizasyonu konseptinin ortaya çıkışından itibaren ortaya konmuş olan tüm çalışmaların bir derlemesini yapmışlar ve matematiksel modellemesini sunmuşlardır. Bunların yanında işlemci meşgul zamanı parametresini ortaya koyarak senkron olarak işlemcinin meşgul bulunduğu zaman dilimini ve işlemci boş zamanını hesaplayarak işlem süreci organizasyonu algoritmaları kullanılarak elde edilen işlemci meşgul ve işlemci boş zamanı parametrelerinin analizini gerçekleştirmişlerdir.

Yukarıda belirtilen kaynakların incelenmesi sonucunda, tek işlemcili bir sistemde arıza kaldırılabilirliğin sağlanabilmesi için işlemci üzerinde görevlerin pasif kopyalarının da organize edilebileceği miktarda zaman diliminin ayrılmasının gerektiği görülmüştür. Bu ise mevcut çalışmalarda sabit bir parametre olarak düşünülen görev kapasitelerinin değiştirilebilir olmasını gerektirmektedir. Görevler için sağlanacak kapasite artırımı ile her görevin pasif kopyasının görev kapasitesinin tamamlanmasını müteakip planlanması

ile arıza kaldırılabilirlik özelliğinin sağlanabileceği sonucuna varılmış ve kapasite parametresinin değiştirilebilirliği üzerine yoğunlaşılarak, yapılan test ve analiz sonuçları doğrultusunda dengeli bir organizasyon şeması önerilmiştir.

2. KURAMSAL TEMELLER VE ANALİZİ

Gerçek zamanlı sistemlerde sistem durumu, sistem parametreleri, arıza kaldırılabilirlik yöntemleri, görev geliş modelleri, önceliklendirme temelli işlem süreci organizasyonu algoritmaları hakkındaki tanım ve detaylı analizler aşağıda belirtilmiştir.

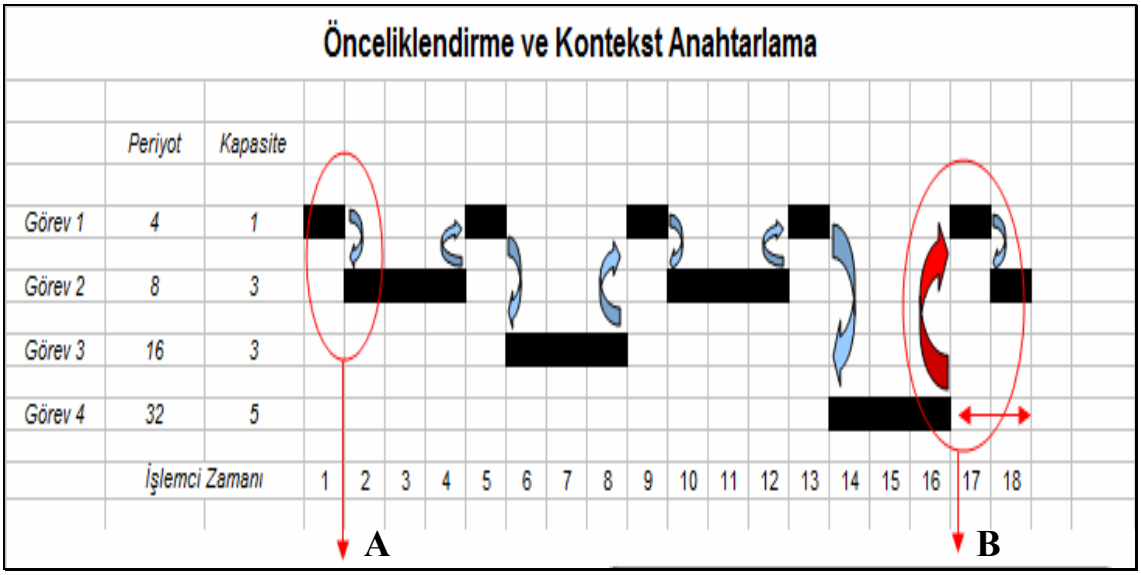
2.1 Sistem Durumu ve Sistem Parametreleri

Gerçek zamanlı sistemlerde görev kümelerine kaynak tahsisi işlemi, işlem süreci organizatörünün görevlerin kaynak tahsisi başvurularına neye göre cevap vereceğini belirleyen görev **önceliği** parametresi doğrultusunda gerçekleştirilir. Görev önceliği doğrultusunda bir görevin diğer görevden daha önce kaynak kullanımına izin verilmesi işlemine ise **önceliklendirme** adı verilmektedir. Önceliklendirme parametresi statik olarak sistem çalışmaya başlamadan belirlenebileceği gibi sistemin değişen durumlara cevap verebilmesini sağlayacak şekilde dinamik olarak da belirlenebilmektedir.

Önceliklendirme, dinamik işlem süreci organizasyonu işleminin temel parametresidir. Sistem içerisinde aynı anda kaynak başvurusunun yapıldığı her bir zaman diliminde önceliklendirme işlemi gerçekleştirilerek görev kümesi içerisindeki tüm görevlerin en son gerçekleştirilme zamanı öncesinde gerçekleştirilmelerini sağlamak amaçlanmakta ve sistemin değişen durumlara cevap verebilmesini sağlayacak organizatörlerin tasarlanmasına imkân vermektedir. Önceliklendirme hesaplamalarının sistem kaynakları kullanılarak gerçekleştirildiği hesaba katıldığında önceliklendirme işleminin, belirtilen avantajlarının yanında **çalışma zamanı giderlerini** de beraberinde getirdiği değerlendirilmelidir. Etkin bir sistemde, ihtiyaç bulunan durumlarda önceliklendirmenin gerçekleştirilmesi arzu edilmekle beraber, çalışma zamanı giderlerinin azaltılması maksadıyla önceliklendirme değerinin belirli bir sınırda tutulması gerekmektedir.

Çalışma zamanı giderlerinin artmasına yol açan bir diğer parametre ise **kontekst anahtarlama** işlemidir. Kontekst anahtarlama işlemi, paylaşılan kaynakların görevler arasındaki değişimidir. Sistem içerisinde kaynak değişimi işlemi, kaynak tahsisi (allocation) ve kaynak sonlandırma (deallocation) işlemleri dolayısıyla sisteme ilave çalışma zamanı giderleri getirmektedir.

Önceliklendirme ve kontekst anahtarlama işlemleri Şekil 2.1'de ifade edilmiştir.



Şekil 2.1 Önceliklendirme ve Kontekst Anahtarlama İşlemleri

(A) Konteks Anahtarlama:

Sistem kaynaklarının görevler arasındaki her bir değişimine **kontekst anahtarlama** adı verilir. Yukıda belirtilen organizasyon şemasında toplam 9 adet konteks anahtarlama işlemi görülmektedir.

(B) Önceliklendirme:

Sistem kaynakları herhangi bir görevin kullanımında iken daha yüksek önceliğe sahip bir görevin kaynakları devralması işlemine **önceliklendirme** denir. 17'nci zaman diliminde daha yüksek önceliğe sahip olan Görev 1, işlemci kaynaklarını, kapasitesinin tamamını aktive etmemiş olmasına rağmen Görev 4'ten devralır.

2.2 Arıza Kaldırılabilirlik

Arıza kaldırılabilirlik, katı gerçek zamanlı sistemler için destekledikleri görevlerin kritikliği nedeniyle oldukça önemli bir konudur. Dolayısıyla sistem üzerinde geçici veya kalıcı olarak meydana gelebilecek hata veya arıza durumlarında sistemin çalışabilirliğinin etkilenmemesi gerekir. Arıza kaldırılabilirliğin seviyesini etkileyen en önemli unsur sistem kaynaklarının durumudur. Kullanılan kaynakların fiziksel olarak yedeğinin tutulabileceği yedek bir kaynağın bulunması durumunda hem donanımsal hem de yazılımsal arıza kaldırılabilir sistemler tasarlanabilirken kaynağın tek olması durumunda sisteme yalnızca yazılımsal arıza kaldırılabilirlik özelliği kazandırılabilmektedir. Bu ise her bir görevin her bir aktivasyonu için genişletilmiş bir kapasite ayrılması ve görev yedeklerinin bu ek zaman dilimlerinde gerçekleştirilmesi esasına dayanmaktadır.

Arıza kaldırılabilir bir sistem üzerinde bu özelliğin sağlanabilmesi amacıyla orjinal görevin kopyaları yedek olarak planlanmakta ve kaynak tahsisi yapılmaktadır. Bu çalışmada tek işlemcili bir sistem üzerinde yazılımsal arıza kaldırılabilirliğin sağlanabilmesi için kapasite değerlerinin artırılabilmesini sağlayacak bir işlem süreci organizasyonu şemasının tasarlanması amaçlanmıştır.

Sistemler arıza kaldırılabilirlikleri açısından aşağıdaki şekilde sınıflandırılabilirler:

1) Esas/Yedek (Primary/Backup) Metodu ile Arıza Kaldırılabilir Sistemler
2) Yedek Donanım (Spare) Metodu ile Arıza Kaldırılabilir Sistemler

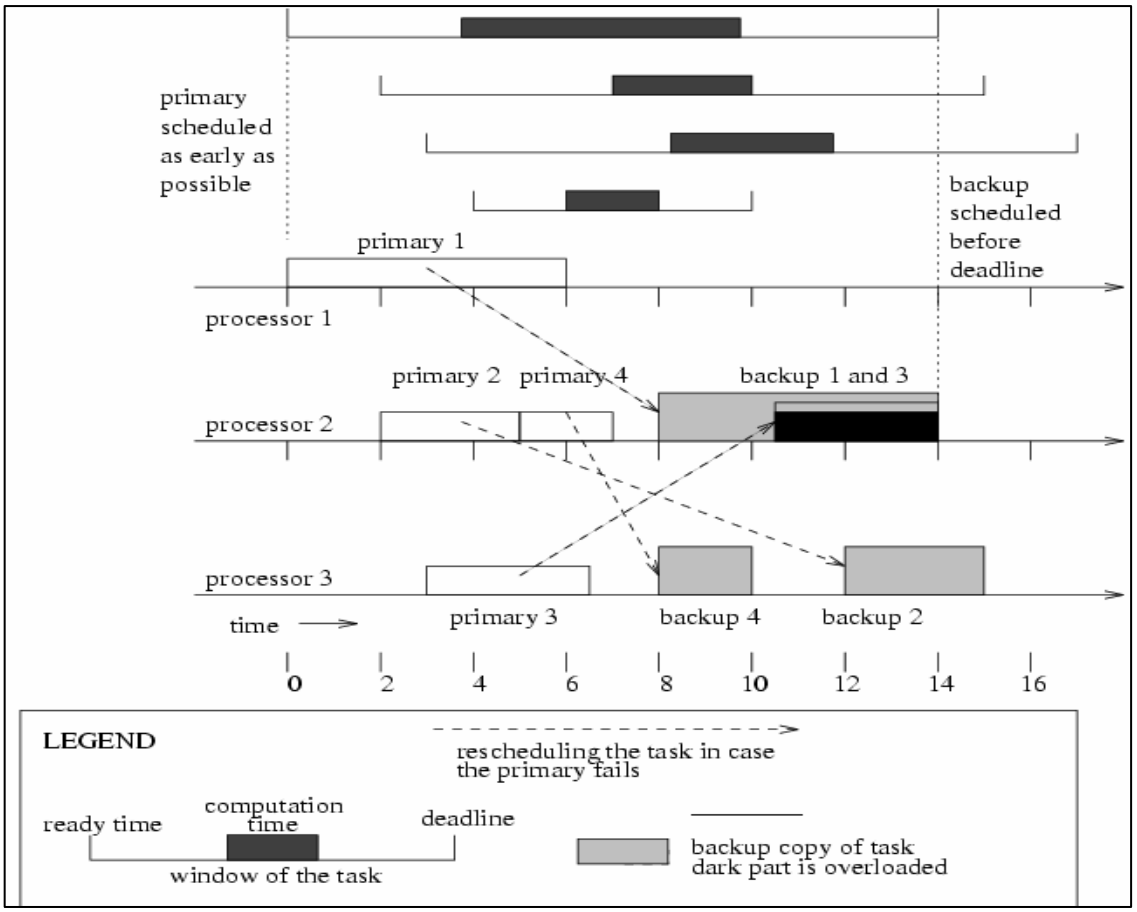
2.2.1 Esas Yedek Metodu

Çok işlemcili bir mimaride değerlendirilebilecek bir yöntemdir. Bu mimaride arıza kaldırılabilirlik bir görevin orjinal ve yedek kopyalarının farklı işlemciler üzerinde farklı zaman dilimlerine planlanması yoluyla sağlanır.

Esas Yedek yönteminde takip edilmesi gereken hususlar:

1) Bir göreve ait esas ve yedek kopyaların her ikisinin de tamamlanma zamanı öncesinde planlanmış olması gerekir.
2) Bir görevin yedek kopyası, esas kopyanın planlandığı işlemciden farklı bir işlemci üzerine planlanmalıdır.
3) İki veya daha fazla farklı işlemcinin yedek kopyası bir diğer işlemci üzerine planlanıyorsa bunların aynı zaman dilimi üzerine planlanması (yedek bindirmesi – backup overloading) kaynak verimliliği açısından mutlaka kullanılması gereken bir özelliktir.
4) Her görev için esas kopyanın herhangi bir arıza olmadan gerçekleştirilmesi durumunda yedek kopyalara atanan sistem kaynaklarının geri alınması (yedek geri tahsisi – backup deallocation) işlemi gerçekleştirilmelidir.

Esas Yedek metodunun gerçekleştirilmesine ait bir örnek Şekil 2.2'de (Ghosh *et al.* 1994) gösterilmiştir. 1 ve 3 numaralı görevlerin yedek kopyaları 2 numaralı işlemci üzerinde ve aynı zaman diliminde planlanmıştır. Şekilden de görüldüğü üzere her görevin yedeği farklı bir işlemci üzerinde planlanmıştır. Bu, sistemin aynı anda yalnızca tek bir işlemci arızasını tolere edebileceği anlamına gelmektedir.



Şekil 2.2 Esas Yedek yöntemiyle arıza kaldırılabilirlik (Ghosh et al. 1994)

Tek işlemcili bir sistem üzerinde donanımsal arızaların tolere edilebilmesi ise mümkün değildir. Dolayısıyla arıza kaldırılabilirliğin sağlanabilmesi ancak yazılımsal olarak sağlanabilir ve yalnızca görevin gerçekleştirilmesi esnasında sadece göreve yönelik olarak oluşabilecek hataları tolere eder. Bu ise tek işlemci üzerinde her bir görevin hem orijinal hem de yedek kopyalarının son gerçekleştirilme zamanları dâhilinde organize edilebilmesine bağlıdır. Bu işlem ancak her bir görev için kapasite değerlerinin üzerinde kaynak tahsisi yapılabilmesi durumunda gerçekleştirilebilir. Dolayısıyla yazılımsal arıza kaldırılabilirliğin sağlanabilmesi için kapasite değerinin sistemin çalışması esnasında güncellenebilirliği önem arz etmektedir.

Bu tez çalışmasında tek işlemcili bir sistem üzerinde yazılımsal arıza kaldırılabilirliğin sağlanabilmesi için ihtiyaç duyulan dengeli kapasite artırımı işlemi üzerinde durulmuştur.

2.2.2 Yedek Donanım Metodu

Bu yöntem görevlere ait yedek kopyaların çalışan sistem kaynakları haricinde tahsis edilen bir yedek donanım üzerinde ve yalnızca arıza durumunda kullanılmak üzere planlanması işlemidir.

Yedek olarak ayrılan donanım sistem içerisinde ve herhangi bir arıza olmaması durumunda üzerinde hiçbir görev çalıştırmayan şekilde bekletilmektedir. Bu yöntem sistem kaynaklarının atıl durumda bekletilmesini temel prensip olarak aldığından dolayı tercih edilmemektedir.

Ancak kalıcı donanım arızası olması durumunda eşdeğer seviyede bir donanımın bulunması sebebiyle sistemin bu durumları da tolere edebilmesine imkân verir.

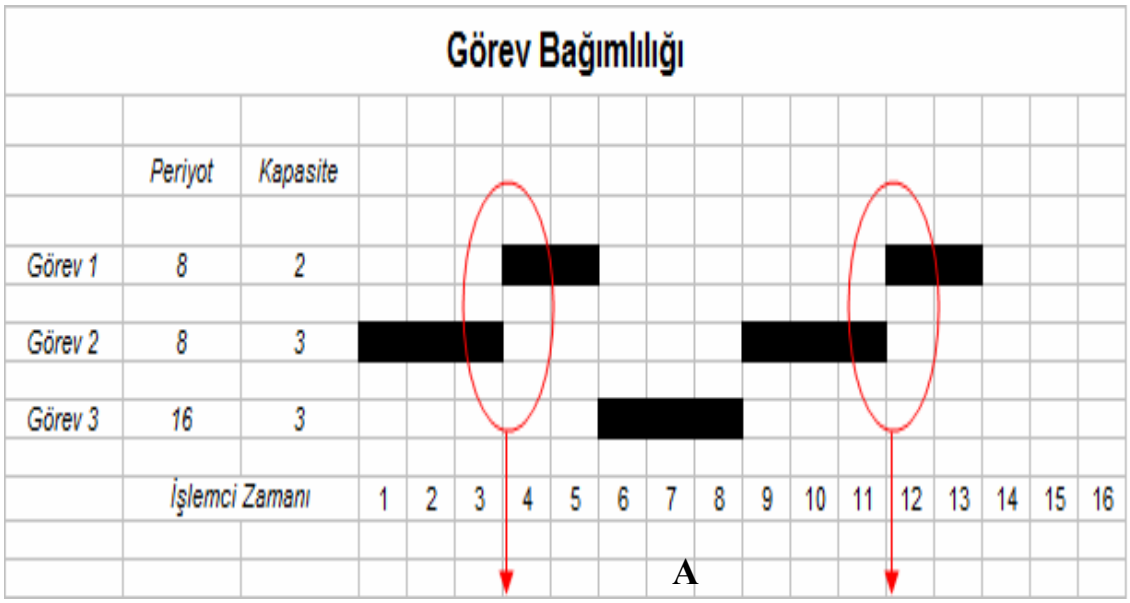
2.3 Görev Modelleri

Bir görev, belirlenen bir işlem süreci organizatörü tarafından ele alınan ve belirli bir frekansla sistem üzerinde aktive edilen sıralı işlemlerin toplamıdır. Görevler periyodik veya aperiyodik olarak sisteme dâhil olabilir. Aperiyodik bir görev sistem içerisinde rastgele veya belirli bir zaman diliminde aktive edilirken periyodik görevler sistem içerisinde belirtilen periyotlar dâhilinde tekrar tekrar gerçekleştirilir.

Her bir periyodik görev $G_{i,k}$, (k = 1, 2…) sonsuz sıralı işlemlerden oluşur ve sırasıyla kapasite, en son gerçekleştirme zamanı ve periyot değerlerinden oluşan $G_{i,k} = (C_i, D_i, T_i)$ ile ifade edilir. Görevlerin belirlenen zaman dilimleri içerisinde tamamlanamamasının sistem için ölümcül sonuçlar vereceği katı kısıtlara sahip bir sistemde k'ıncı görevin gerçekleştirilmesi işlemi $(k - 1) * T_i$ anında başlar ve herhangi bir gecikme (offset delay) tanımlanmaması durumunda $(k - 1) * T_i + D_i$ anından önce tamamlanmak zorundadır.

Bu çalışma kapsamında, belirli kriterlere göre belirlenmiş olan ve G= {G_1, G_2…, G_n} ile ifade edilen n adet periyodik görevden oluşan görev kümelerine ait genel özellikler, tek işlemcili, katı gerçek zamanlı kısıtlara sahip bir sistem üzerinde test edilmiştir.

1) Tüm görev modelleri, her hangi bir bağımlılık kısıtına sahip olmayan bağımsız görevlerden oluşmaktadır. Görev bağımlılığı Şekil 2.3'te ifade edilmiştir.



Şekil 2.3 Görev Bağımlılığı

(A) Görev Bağımlılığı:
Görev 1, Görev 2'ye bağımlı olarak tanımlanmıştır. Dolayısıyla Görev2 tamamlanmadan Görev 1'e kaynak tahsisi yapılamaz.

2) Her görev kümesi içerisindeki her bir görev $G_{i,k}$ = (C_i, D_i, T_i) parametrelerine sahiptir.

3) Detaylı analizler EDF ve LLF dinamik işlem süreci organizasyonu algoritmalarının aşağıda belirtilen durumlar karşısındaki davranışlarının tespiti ve eğer varsa dezavantajlı durumların iyileştirilmesine yönelik olarak gerçekleştirilmiştir:

a) Görev kümesi içerisindeki tüm görevlerin her bir aktivasyonda farklı kapasite değerlerine ihtiyaç duyduğu bir sistemde dinamik işlem süreci algoritmalarının davranışları test edilerek analiz edilmiştir.

b) Görev kümesi içerisindeki her bir görev periyodik, sporadik ve poisson geliş modelleri ile test edilerek analiz edilmiştir.

c) İşlem süreci organize edilebilirliği ve görevlerin en kötü durumda gerçekleştirilme zamanları parametre olarak değerlendirilmiş ve algoritmaların fizibilite durumları ele alınmıştır.

d) Her iki dinamik işlem süreci organizasyonu algoritması da önceliklendirme temelli olarak değerlendirilmiştir.

e) Her görevin her bir aktivasyonu için kapasite güncelleme ihtiyacını karşılayabilecek bir sistemi simüle edebilmek maksadıyla, kaynak kullanımını belli değerler arasında tutacak şekilde belirlenmiş alt ve üst kapasite değerleri doğrultusunda rastgele kapasite dizisi üretim şeması tasarlanmıştır.

2.4 Görev Geliş Modelleri

Görevler, görevlerin aktivasyonunu ifade eden geliş modellerine sahiptir. Aperiyodik bir görev sistem tamamlanıncaya kadar yalnızca bir defa aktive edilirken **periyodik görevler** defalarca gerçekleştirilir ve ard arda gelen iki aktivasyon arasındaki gecikme periyot değerine eşit sabit bir değerdir. Periyodik görevlerin gecikme zamanındaki değişimlere bağlı olarak poisson ve sporadik olarak adlandırılan türleri mevcuttur.

Bir **poisson süreç görevi** periyodik görevler gibi defalarca aktive edilir ve ard arda gelen iki aktivasyon arasındaki gecikme sabit bir değer değil rastgele bir değerdir. Bu gecikme değerini oluşturmak maksadıyla kullanılan rastgelelik kuralı, bu gecikme değerini üstel olarak (poisson süreci) hesaplar.

Sporadik görevler ise ard arda gelen iki görev arasındaki gecikme değerinin mimimum olacak şekilde hesaplandığı ve periyodik görevler gibi defalarca aktive edilen görevlerdir.

2.5 Önceliklendirme Temelli İşlem Süreci Organizasyonu Algoritmaları

Algılayıcı temelli ve dinamik olarak değişen durumlara cevap verebilen kontrol sistemleri üzerinde kullanılmak üzere bir dizi işlem süreci organizasyonu teknikleri tasarlanmıştır. Burada kaynak tahsisi için önceliklendirmenin neye göre yapıldığı ayırt edici bir özellik olarak karşımıza çıkmaktadır.

Bu aşamada mevcut yöntemler; önceliklendirmenin algılayıcı sistemin dinamik olarak değişen durumları değerlendirmeyerek başlangıçta sabit olarak belirlenen öncelik değerlerini kullanarak gerçekleştirmesi (statik işlem süreci organizasyonu) ve sistemin dinamik olarak değişen durumlarına cevap verebilmek maksadıyla her bir eş zamanlı kaynak talebi durumunda dinamik öncelik değerleri hesaplanarak önceliklendirmenin yapılması (dinamik işlem süreci organizasyonu) şeklinde statik veya dinamik olarak gerçekleştirilmektedir.

Önceliklendirme temelli işlem süreci algoritmaları; statik önceliklendirme temelli Rate Monotonic (RM) ile dinamik önceliklendirme temelli Earliest Deadline First (EDF) ve Least Laxity First (LLF) algoritmalarının tasarlanmalarıyla ortaya çıkmıştır. Genel bir kıyaslama ve algoritmaların önceliklendirme kuralları ile örnek organizasyon şemalarının simülasyonları (The Cheddar Project) aşağıda belirtildiği şekildedir:

2.5.1 Statik İşlem Süreci Organizasyon Şeması - RM

Sistem içerisinde frekansı en düşük olan göreve en düşük, en yüksek frekanslı göreve en yüksek önceliklendirmenin atandığı sabit önceliklendirmeli işlem süreci organizasyonu metodudur.

Temel olarak herhangi bir zamanda işlem süreci organizatörü en yüksek önceliğe sahip görevi önce gerçekleştirir. Bir görev kümesi için öncelikle frekanslarına bağlı olarak bir sıralama işlemi gerçekleştirilir. Belirtilen önceliklendirme sırasına göre işlemci kaynakları tahsis edilmeye başlanır. Aynı zaman diliminde sistem kaynaklarını

kullanmak üzere birden fazla görevin talepte bulunması durumunda görevlerin önceliklerine göre kaynak ataması gerçekleştirilir.

Bir görev kümesi için kaynak kullanım oranı aşağıdaki formülle hesaplanmaktadır: (Stewart and Khosla 1999)

$$W_n = n * (2^{1/n} - 1) \quad (1)$$

Örneğin;
1 görev için $W_1 = 1 * (2^{1/1} - 1) =$ %100 lük bir kaynak kullanım oranı,
2 görev için $W_2 = 2 * (2^{1/2} - 1) =$ %83 lük bir kaynak kullanım oranı gerçekleşmektedir.

Limit durumunda belirtilen kaynak kullanım oranı W_∞=69% dur. Dolayısıyla %69'un altındaki işlemci kullanım oranı için bütün görevlerin mutlaka gerçekleştirilebileceği garanti edilmektedir. Bu oranın üstündeki kaynak tahsisi ihtiyacı durumunda ise önceliklendirmesi düşük olan (düşük frekanslı) görevlerin reddedilmesi söz konusudur.

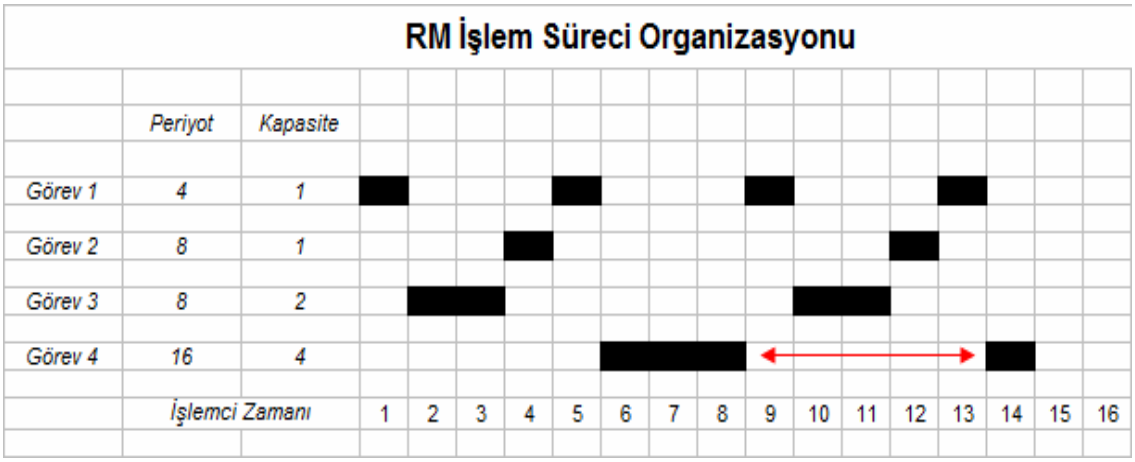
Frekans Monoton algoritmanın en önemli dezavantajı sistem üzerinde dinamik olarak değişen durumlara cevap verebilir özellikte olmamasıdır. Örneğin algılayıcı sistemler vasıtasıyla değerlendirilmek üzere gelen görev kümeleri için periyot değerlerinde oluşabilecek değişimler görev kümelerinin önceliklendirmelerinin sabit olarak belirlenmiş olması nedeniyle önceliklendirme hesaplamalarında değerlendirmeye alınmamaktadır.

Algoritmanın bir diğer önemli dezavantajı ise planlanabilecek maksimum görev için organize edilebilirlik yüzdesinin %100'den düşük olmasıdır. Bu nedenle sistem kaynakları üzerinde anlık aşırı yüklenmeler olması durumunda önemli sayıda kritik görevin reddedilmesi ve sistem kaynaklarını kullanmak üzere bu görevlere kaynak tahsisi yapılmaması söz konusu olacaktır. Bu ise sistemde sadece yüksek frekanslı ve kısa zaman dilimi içerisinde gerçekleştirilebilecek görevler haricinde görev çalıştırılamaması anlamına gelmektedir. Dolayısıyla düşük frekanslı ama kritik

seviyedeki görevlerin gerçekleştirilememesi gerçek zamanlı bir sistem için istenen bir durum değildir.

Algoritma görev periyot değerlerinin indekslenmesi yöntemiyle çalışmaktadır. Minimum periyot değerine (maksimum frekansa) sahip görev en yüksek öncelik değerine sahip olur. RM İşlem Süreci Organizasyon Şeması kullanılarak gerçekleştirilen kaynak tahsisi işlemi Şekil 2.4'te görüldüğü gibidir.

Statik Öncelik : Return (Minimum Görev Periyot Değeri) (2)



Statik Önceliklendirme

Her zaman diliminde periyot değeri en küçük yani aktivasyon frekansı en yüksek olan göreve kaynak tahsisi yapılır.
(*) 9 anında periyodu daha küçük olan Görev 1, sistem kaynaklarının kullanımını Görev 4'ten devralır (önceliklendirme).
(*) Sistemde toplam 9 adet konteks anahtarlama islemi gerçekleştirilmistir.

Şekil 2.4 RM İşlem Süreci Organizasyon Şeması Kullanılarak Gerçekleştirilen Kaynak Tahsis İşlemi

2.5.2 Dinamik İşlem Süreci Algoritmaları – EDF ve LLF

EDF ve LLF algoritmaları sistemin dinamik parametrelerine göre önceliklendirme işleminin gerçekleştirilebileceği bir işlem süreci organizasyonu sağlayabilmek amacıyla tasarlanmıştır.

2.5.2.1 EDF

EDF algoritması bir görevin tamamlanma zamanını o görevin önceliklendirmesi için parametre olarak kullanır. En yakın tamamlanma zamanına sahip göreve daha yüksek önceliklendirme belirlenir.

Sistem kaynaklarına eş zamanlı erişim talebi geldiğinde görevler içerisinden tamamlanma zamanı en erken olan görev daha yüksek önceliğe sahip olduğundan önce gerçekleştirilir. Bununla amaç görev kümelerinin belirtilen tamamlanma zamanı içerisinde gerçekleştirilebilirliğini arttırmaktır.

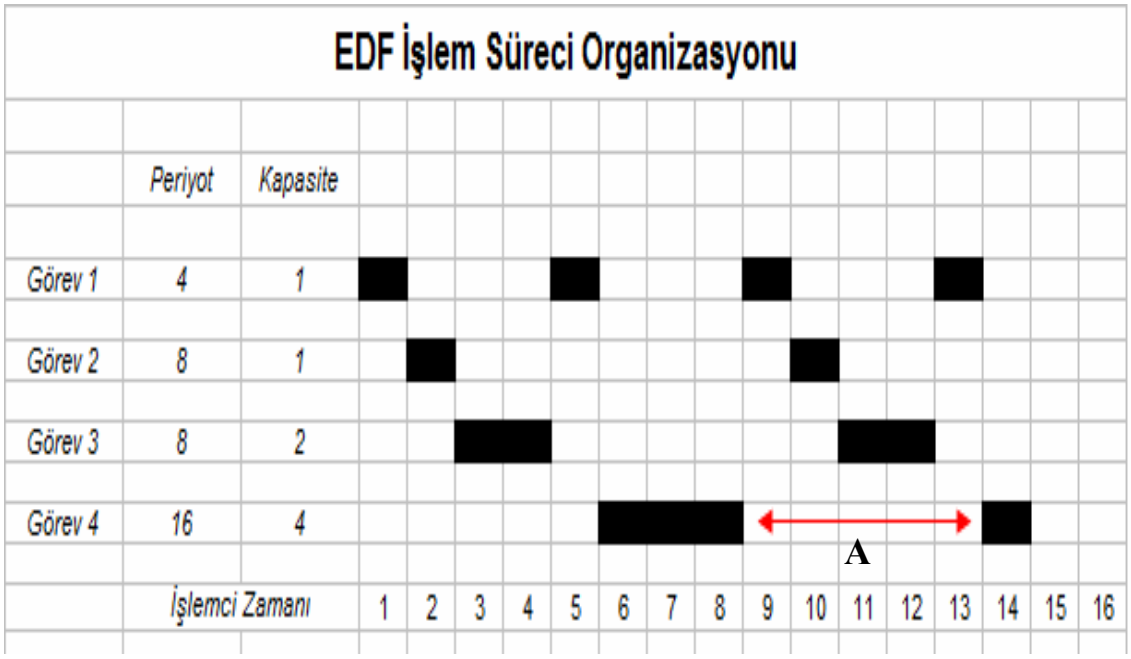
Algoritmanın Monoton Frekanslı algoritmaya göre avantajı sistemin değişen durumlara göre görev kümesi önceliklendirmelerini dinamik olarak belirleyebilmesidir. Ayrıca organize edilebilirlik oranının tüm görev kümeleri için %100 seviyesinde olması da önemli bir avantajdır. Bunun anlamı, toplam kaynak kullanım oranı %100 seviyesinin üzerinde olmadığı müddetçe EDF algoritmasının kaynak tahsisi işlemini herhangi bir gecikmeye sebebiyet vermeksizin gerçekleştirebileceğidir.

EDF algoritması ile ilgili en temel problem; statik bir önceliklendirme tahsisi yapılmamasından dolayı sistem üzerindeki anlık aşırı yüklenme durumunda hangi görevlerin reddedileceğinin önceden belirlenebilmesi imkânının olmamasıdır.

EDF - Dinamik Öncelik: Görev Başlangıç Zamanı (3)
+ [(Görev Aktivasyon Sayısı – 1)
* Görev Periyodu]
+ Görev Son Gerçekleştirme Zamanı

Return (Minimum Dinamik Öncelik Değeri)

EDF İşlem Organizasyon Şeması kullanılarak gerçekleştirilen kaynak tahsisi işlemi Şekil 2.5'te görüldüğü gibidir.



Şekil 2.5 EDF İşlem Süreci Organizasyon Şeması Kullanılarak Gerçekleştirilen Kaynak Tahsis İşlemi

(A) Dinamik Önceliklendirme (EDF) :

Her zaman diliminde son tamamlanma zamanı (deadline) doğrultusunda hesaplanan dinamik öncelik değerine göre kaynak tahsisi yapılır. 9 anında dinamik önceliği daha yüksek olan Görev 1, sistem kaynaklarının kullanımını Görev 4'ten devralır (önceliklendirme). Sistemde toplam 9 adet konteks anahtarlama işlemi gerçekleştirilmiştir.

2.5.2.2 LLF

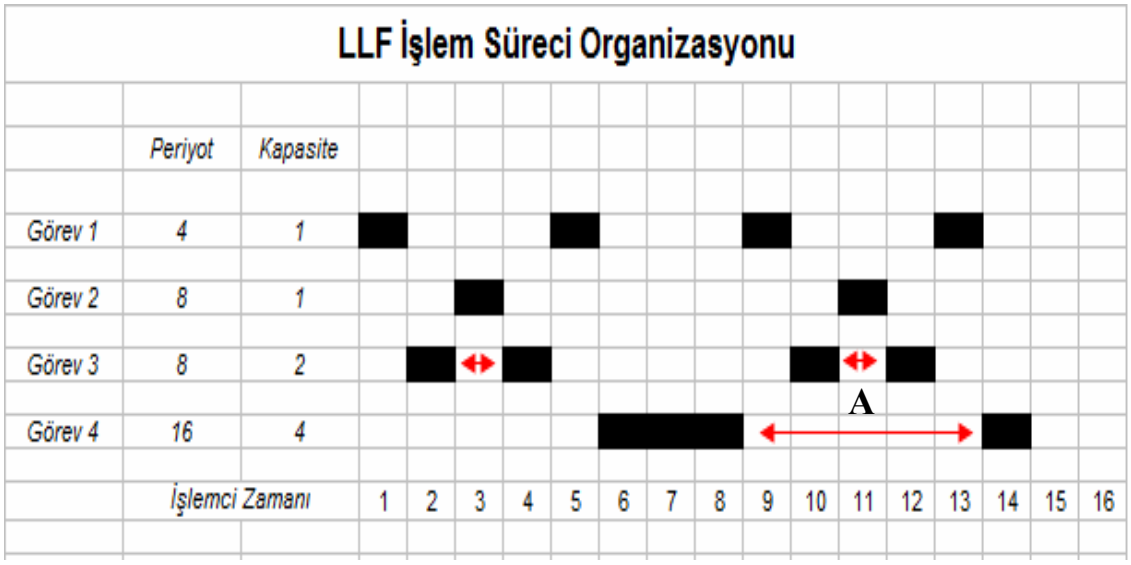
LLF algoritması önceliklendirmelerin belirlenmesi aşamasında göreve ait ertelenebilirlik parametresini değerlendirmektedir. Ertelenebilirliği en az olan görev için en yüksek önceliklendirme değeri belirlenir.

LLF - Dinamik Deadline: Görev Başlangıç Zamanı (4)
+ [(Görev Aktivasyon Sayısı – 1)
* Görev Periyodu]
+ Görev Son Gerçekleştirme Zamanı

Ertelenebilirlik: Dinamik Deadline – Görev İçin Kalan Kapasite

Return (Minimum Ertelenebilirlik Değeri)

LLF İşlem Organizasyon Şeması kullanılarak gerçekleştirilen kaynak tahsisi işlemi Şekil 2.6'da görüldüğü gibidir.



Şekil 2.6 LLF İşlem Süreci Organizasyon Şeması Kullanılarak Gerçekleştirilen Kaynak Tahsis İşlemi

(A) Dinamik Önceliklendirme (LLF):

Her zaman diliminde ertelenebilirlik (laxity) değeri doğrultusunda hesaplanan dinamik öncelik değerine göre kaynak tahsisi yapılır. Sistemde 3, 9 ve 11 anlarında dinamik öncelik değeri doğrultusunda önceliklendirme işlemi gerçekleştirilmiştir. Sistemde toplam 11 adet konteks anahtarlama işlemi gerçekleştirilmiştir.

LLF Algoritması da EDF algoritmasına benzer şekilde değişen durumlara göre görev seti önceliklendirmelerini dinamik olarak belirleyebilmektedir ve organize edilebilirlik oranı tüm görev setleri için %100 seviyesindedir.

EDF algoritması ile aynı şekilde statik bir önceliklendirme tahsisi yapılmamasından dolayı sistem üzerindeki anlık aşırı yüklenme durumunda hangi görevlerin reddedileceğinin önceden belirlenebilmesi imkânı bulunmamaktadır.

Ayrıca ortalama Kontekst Anahtarlama ve Önceliklendirme sayıları sistemdeki yükle doğru orantılı olarak EDF algoritmasından daha fazladır. Bu ise çalışma zamanı giderlerinin daha yüksek olması anlamına gelmektedir.

2.5.3 Parametreler ve Tanımlar

Sistemde kullanılan parametreler ve tanımları aşağıda belirtilmiştir:

*** Temel Periyot (P)** parametresi periyodik görev kümesinin geliş modelinin benzer şekilde tekrarlandığı döngüdür (Leung and Merril 1980).

$$P = \mathrm{LCM}\{ T_i \} \text{ where} (1 \le i \le n) \tag{5}$$

*** İşlemci Kullanım Oranı (U)**, görev kümesindeki tüm görevlerin gerçekleştirilmesi için kullanılan işlemci kaynağının oranıdır (Liu and Layland 1973).

$$U = \sum_{1}^{n} (C_i / T_i) \tag{6}$$

*** İş Yükü (W)**, verilen bir kritik anda (s_i, i Є [1,n], s_i =0) başlangıç zamanları [0,t] aralığında olan tüm görev parçacıklarının talep ettiği toplam işlemci kaynağı miktarıdır (Baruah, Mok and Rozier 1990).

$$W(t) = \sum_{1}^{n} \left(P / T_i \right) * C_i \tag{7}$$

*** İşlemci Boş Zamanı (B),** görev kümesi içerisindeki belirli bir görev için, temel periyot içerisinde, diğer görevlerin ihtiyaç duyduğu iş yüklerini de dikkate alarak hesaplanan kullanılabilir işlemci zamanını ifade eder.

$$B(t) = P - CT(\text{CurrentTime}) - W(t) \tag{8}$$

*** Kaynak İhtiyacı (Wrk)**, görev kümesi içerisindeki belirli bir görev için, görev aktivasyon sayısını da dikkate alarak (içerisinde bulunduğu aktivasyonu tamamladığı varsayılır) hesaplanan işlemci kaynağı ihtiyacıdır.

$$\text{Wrk}(t) = \left(P / T_i \right) * C_i - C_i * \text{Task\_Act\_Number}(i) \tag{9}$$

Kapasite güncelleme işleminin uygulanacağı gerçek zamanlı arıza kaldırılabilir bir sistemin tasarımında kullanılmak üzere yukarıda belirtilen temel parametre, formül, algoritma ve organizasyon şemaları dikkate alınarak öncelikle görevler için değişken kapasiteyi sağlayacak bir kapasite matrisi oluşturulmuş, daha sonra oluşturulan sistemin farklı dinamik işlem süreci organizasyonu algoritmaları ve farklı geliş modelleri ile test ve analizi gerçekleştirilmiştir. Elde edilen sonuçlar doğrultusunda daha dengeli ve elverişli bir sistemin oluşturulmasına yönelik olarak bir algoritma ve sistem şeması tasarlanarak uygulanmıştır.

3. İŞLEM SÜRECİ ORGANİZASYONU TASARIMI İÇİN GEREKLİ MODELLERİN GELİŞTİRİLMESİ

Gerçek zamanlı bir işlem süreci organizasyon şemasına arıza kaldırılabilirlik özelliğinin kazandırılabilmesi için ihtiyaç duyulan kapasite güncellemesini sağlayabilmek üzere bir görev kümesi ve kapasite üreteci tasarımı ile sistemde kullanılmak kapasite matrisi oluşturulması işlemleri gerçekleştirilmiştir.

Bu çalışmada önceliklendirme temelli dinamik işlem süreci organizasyonu algoritmaları kullanılarak algoritmaların aşağıda belirtilen durumlardaki davranış şekilleri derinlemesine incelenmiştir;

(1) G= {G1, G2…} ile ifade edilen ve n adet görevden oluşan bir görev kümesinin tek işlemcili katı gerçek zamanlı bir sistemdeki işlem süreci organizasyon şemasının oluşturulması problemi ele alınmıştır. Bütün görevler bağımsız ve $G_{i,k} = (C_i, D_i, T_i)$ parametrelerine sahiptir.
(2) Belirli bir görev kümesi için, görev kümesindeki tüm görevlerin her aktivasyonlarında değişen kapasitelere ihtiyaç duyduğu bir durumda algoritmaların davranışlarının;

a) Organize Edilebilirlik (SCH),
b) En Kötü Durumda Gerçekleştirilme Zamanı (WCET),
c) Önceliklendirme (PR),
d) Kontekst Anahtarlama (CS)

parametreleri kullanılarak analizi ve kıyaslanması gerçekleştirilmiştir.
(3) Aynı sonuçların statik kapasite değerlerine sahip görevlerden oluşan görev kümesi ile elde edilen sonuçlarla karşılaştırılması sağlanmıştır.
(4) Görev kümesindeki görevlerin periyodik, sporadik ve poisson gibi farklı geliş modelleri için nasıl davranış gösterdiğinin analizi yapılmıştır.

Söz konusu analizlerde kullanılacak görev kümelerinin oluşturulması işlemi, sistemin toplam işlemci kullanım oranı dikkate alınarak geliştirilen ve her bir göreve belirlenen alt ve üst limit değerler arasından farklı kapasite değerleri atayan Rastgele Dizi Üreteci ile gerçekleştirilmektedir.

3.1 Kapasite Güncelleme

Dinamik işlem süreci organizasyonu teorisinin çok önemli özelliklerinden birisi de sistemin değişen koşullarına ve ihtiyaçlarına cevap verebilmektir. Kapasite değerinin görev kümesindeki tüm görevler için sabit olarak tanımlanması nedeniyle kapasite güncellemesine ihtiyaç duyan görevler için daha az veya daha fazla kapasite atanmasına ihtimal bulunmamaktadır. Bu durum sistemin değişen durumlara cevap verebilirliğini ortadan kaldırdığı gibi arıza kaldırılabilirliğin sağlanabilmesi için de görev kümesindeki her bir görev için birer yedek görevin tanımlanmasını zorunlu kılar. Bu sorunu ortadan kaldırabilmek maksadıyla; tanımlanmış olan başlangıç kapasite değerlerini, görev kümesindeki her bir görev için ve her bir periyotta, tasarlanan kapasite dizisi üreteci kullanılarak elde edilen kapasite değerleriyle güncelleyen bir sistem geliştirilmiştir. Kapasite üreteci sistemin iş yükü dikkate alınarak tasarlanmıştır. Böyle bir sistemin geliştirilmesi ile değişen durumlara cevap verebilecek şekilde kapasite güncellemesinin mümkün olduğu dinamik bir işlem süreci organizasyon şeması elde edilmiş ve bu sayede algoritmaların değişen durumlara verdiği cevaplar analiz edilebilmiştir.

3.2 Görev Kümesi Oluşturulması

Algoritmaların test edilebilmesi maksadıyla kullanılmak üzere uygun görev kümelerinin oluşturulması önem arz eden bir konudur. Bu testleri anlamlı hale getirebilmek için görev kümesi seçimini konu alan bazı araştırmalar dikkate alınmış ve ilk olarak temel periyot olarak adlandırılan test aralığının belirlenmesi üzerinde durulmuştur. Bazı özel durumlar haricinde bir sistemin fizibilitesinin araştırılması işlemi, sistemin sonlu bir fizibilite aralığında simüle edilerek görev kümesindeki tüm görevlerin en son tamamlanma zamanından önce gerçekleştirilebilmesi demektir (Goossens 2001). Daha elverişli bir işlem süreci organizasyonunun elde edilebilmesi için en önemli konu çalışma alanının mümkün olduğunca genişletilmemesi ve çok yüksek olmayan temel periyot içerisinde gerçekleştirilmesidir. Simülasyon aralığının azaltılabilmesi ve periyotların en küçük ortak katı ile ifade edilen temel periyot değerinin düşük seviyede tutulabilmesi için görev periyot değerlerinin mantıklı şekilde seçilmesi gerekmektedir. Yapılan çalışmalar neticesinde görev periyot değerlerinin aralarında harmonik ilişki

bulunan değerlerden tespit edilmesinin daha düşük temel periyot değeri sağlayacağı ve statik işlem süreci organizatörleri için bile daha yüksek kullanım oranı sağlayabildiği ortaya konmuştur (Buttazzo, 2005).

G= {G_1, G_2, G_3, G_4} ile ifade edilen görev kümesi için kapasite, en son gerçekleştirilme zamanı (WCET) ve periyot değerleri Çizelge 3.1'de belirtildiği gibidir.

Çizelge 3.1 Periyotları arasında harmonik ilişki bulunan görev kümesi

Adı	**Kapasite**	**WCET**	**Periyot**
G_1	1	4	4
G_2	3	8	8
G_3	3	16	16
G_4	5	32	32

Çizelge 3.1'de görüldüğü gibi en son gerçekleştirilme zamanları görecelidir (George 1996) ve aralarında harmonik ilişki bulunan 4, 8, 16, 32 periyot değerlerine eşittir. Bu periyot değerleri temel periyodu P = LCM (4, 8, 16, 32) = 32 olarak belirler ve analizi kolaylaştıracak seviyede bir simülasyon aralığı ortaya koyar. Bu 32 zaman dilimi içerisinde görev kümesi içerisinde yer alan tüm görevlerin geliş modelleri doğrultusunda en az 1 kez aktivasyonu sağlanmaktadır. 32'nci zaman diliminde birinci görev $P/T_1 = 32/4 = 8$ defa gerçekleştirilirken ikinci görev $P/T_2 = 32/8 = 4$ kez, üçüncü görev $P/T_3 = 32/16 = 2$ kez ve dördüncü görev ise $P/T_4 = 32/32 = 1$ kez gerçekleştirilmektedir.

Simülasyon aralığının tespit edilmesi dışında bir diğer önemli konu ise başlangıçtaki kaynak kullanım oranının (U) tespitidir. Görev kümesini oluşturan görevlerin başlangıç kaynak kullanım oranı tam kullanım oranına yakın bir değer olan "U = ¼ + 3/8 + 3/16 + 5/32 = 0,96875" olarak tespit edilmiştir.

3.3 Rastgele Kapasite Üreteci

Her bir görev için ilk aktivasyonun gerçekleştirilmesini müteakip atanacak değişken kapasite değerlerinin tespit edilmesi işlemi aşağıda belirtilen kurallar doğrultusunda yapılmış ve Çizelge 3.2'de belirtilen aralıklarda kapasite ve kaynak kullanım değerlerine ulaşılmıştır:

1) Görev kümesi içerisinde yer alan tüm görevlerin kapasite değerlerinin 1'e eşit olduğu "minimum kaynak kullanım oranı" ile görevlerin her birinin %25 kaynak kullanım oranını sağladığı "maksimum kaynak kullanım oranları" hesaplanır.
2) İlk görev için rastgele bir değer elde edilir (1 ile görev periyodunun %25'i arasında).
3) İkinci görev için rastgele bir değer elde edilir (1 ile görev periyodunun %25'i ve ilk göreve atanan rastgele değere göre kullanılmayan kapasite değerinin toplamı)
4) Üçüncü kural devam edegelen görevlerin her birine uygulanır.

Görevlere atanan rastgele azalmayan kapasite değerlerinin elde edilmesinde kullanılan ve bu kuralları ifade eden formüller aşağıda belirtildiği şekildedir:

$$RC1 = \text{Rand}(Min1, Max1) \tag{10}$$

$$RC2 = \text{Rand}(Min2, Max2 + \text{abs}(Max1 - RC1)) \tag{11}$$

$$RC3 = \text{Rand}(Min3, Max3 + \text{abs}(Max1 - RC1) + \text{abs}(Max2 - RC2)) \tag{12}$$

$$RC4 = \text{Rand}(Min4, Max4 + \text{abs}(Max1 - RC1) + \text{abs}(Max2 - RC2) + \text{abs}(Max3 - RC3)) \tag{13}$$

[RC = Random Capacity: Rastgele Kapasite, Rand = Randomize: Rastgele Değer Atama Abs = Absolute: Mutlak Değer]

Çizelge 3.2 Görev kapasitelerine atanabilecek minimum ve maksimum değerler

Görev	Perioyot	Min	U(Min)	Max	U(Max)
G_1	4	1	0,25	1	0,25
G_2	8	1	0,125	2	0,25
G_3	16	1	0,0625	4	0,25
G_4	32	1	0,03125	8	0,25
Toplam Kullanım Oranı			**0.46875**		**1,00**

Belirtilen esaslar dâhilinde oluşturulan görev kümesi minimum %46,875 ve maksimum %100 kaynak kullanım oranı sağlamaktadır. Bu çalışma kapsamında Rastgele Kapasite Üreteci kullanılarak elde edilen ortalama kaynak kullanım oranı %85'dir. Rastgele Kapasite Üreteci kullanılarak 4x4 ebatlarında Çizelge 3.3'te belirtilen kapasite matrisi oluşturulmuştur. Her iki dinamik işlem süreci organizasyonu algoritmasının da aynı rastgele kapasite değerleriyle test edilebilmesini sağlamak maksadıyla, her bir görevin tüm aktivasyonları için işaretçiler kullanılarak kapasite matrisindeki değerlerin kullanımı sağlanmıştır. Örneğin, EDF dinamik işlem süreci organizasyonu algoritması ve ikinci görev için kapasite matrisindeki birinci dizinin kullanımını sağlayan kod parçacığı aşağıda belirtilmiştir (EDF-RC LLF-RC kodları **EK 1**'de sunulmuştur):

Start_Section:

```
i : integer; dynamic_priority : array (tasks_range) of integer;
T2_ptr : integer;
T2_randomcapacity_array: array (tasks_range) of integer;
T2_randomcapacity_array(0):=2; T2_randomcapacity_array(1):=1;
T2_randomcapacity_array(2):=1; T2_randomcapacity_array(3):=2;
```

Priority_Section:

```
  for i in tasks_range loop
    if (tasks.capacity(i)>1) then
      if (tasks.rest_of_capacity(i) = 1) then
      if (i=1) then
        tasks.capacity(i):=T2_randomcapacity_array(T2_ptr);
```

```
        T2_ptr:=(T2_ptr+1) mod 4;
      end if;
    end if;
  end loop;
      dynamic_priority := tasks.start_time
       + ((tasks.activation_number-1)*tasks.period) + tasks.deadline;
election_section:
    return min_to_index(dynamic_priority);
```

Çizelge 3.3 Rastgele kapasite üreteci kullanılarak elde edilen kapasite matrisi

	Dizi 1				**Dizi 2**			
C1	1	1	1	1	1	1	1	1
C2	2	1	1	2	1	1	2	1
C3	3	5	4	4	5	4	1	1
C4	2	6	4	4	3	5	8	10
U	*0,75*	*0,875*	*0,75*	*0,875*	*0,78125*	*0,78125*	*0,8125*	*0,75*
	Dizi 3				**Dizi 4**			
C1	1	1	1	1	1	1	1	1
C2	2	1	1	2	2	2	2	2
C3	4	5	2	3	2	2	4	4
C4	7	7	10	7	10	5	5	8
U	*0,96875*	*0,90625*	*0,8125*	*0,90625*	*0,9375*	*0,78125*	*0,90625*	*1*
	Dizi 5				**Dizi 6**			
C1	1	1	1	1	1	1	1	1
C2	2	2	2	2	2	2	1	2
C3	2	4	3	2	4	2	3	3
C4	6	7	9	5	8	5	3	9
U	*0,8125*	*0,96875*	*0,96875*	*0,78125*	*1*	*0,78125*	*0,65625*	*0,96875*
	Dizi 7				**Dizi 8**			
C1	1	1	1	1	1	1	1	1
C2	2	2	1	2	2	2	2	1
C3	3	4	4	3	4	4	4	3
C4	5	6	9	8	1	2	3	7
U	*0,84375*	*0,9375*	*0,90625*	*0,9375*	*0,78125*	*0,8125*	*0,84375*	*0,78125*
	Dizi 9				**Dizi 10**			
C1	1	1	1	1	1	1	1	1
C2	2	1	2	1	1	2	1	1
C3	2	5	1	1	5	2	1	2
C4	6	7	11	6	8	9	4	11
U	*0,8125*	*0,90625*	*0,90625*	*0,625*	*0,9375*	*0,90625*	*0,5625*	*0,84375*

Bu bölümde görev kümesinin tasarımına yönelik olarak ;

1) Başlangıç kapasite ve periyot değerleri kaynak kullanım oranları doğrultusunda belirlenmiştir.
2) Görev kümesinin simülasyon aralığının, aralarında harmonik ilişki bulunan periyot değerlerinin kullanılması ile düşük seviyede tutulması sağlanmıştır.
3) EDF ve LLF dinamik işlem süreci organizasyonu algoritmalarının objektif bir kıyaslamaya tabi tutulabilmesi için kapasite üreteci kullanılarak belirli kısıtlar kullanılarak geliştirilen formüller doğrultusunda kapasite matrisi oluşturulmuştur.

Oluşturulan görev kümesi kullanılarak yapılan testler ve bu testlerin detaylı analizleri ile değişken kapasite kullanımının sabit kapasite durumuna göre üstün ve dezavantajlı durumları tespit edilerek karşılaşılan sorunların çözümüne yönelik sistem tasarımı aşamaları bir sonraki bölümde belirtilmiştir.

4. UYGULAMA VE SİMÜLASYON SONUÇLARI

Bu bölümde, öncelikle, yukarıda belirtilen esaslar dâhilinde oluşturulan görev kümesi, EDF ve LLF dinamik işlem süreci organizasyonu algoritmalarının farklı geliş modelleri ve değişken kapasite durumunda nasıl bir davranış sergileyeceğini analiz etmek üzere, üretilen kapasite matrisindeki kapasite değerleri kullanılmak suretiyle test edilmiştir.

Sonrasında ise elde edilen simülasyon sonuçları ve çalışmanın analizi neticesinde ortaya çıkan problemleri çözmek ve arıza kaldırılabilirliği sağlayabilmek maksadıyla geliştirilen teknik açıklanarak test ve simülasyon sonuçları belirtilmiştir.

4.1 Değişken Kapasiteli Dinamik İşlem Süreci Organizasyon Şemaları

Kapasite atama işlemi her bir görev için içinde bulunduğu aktivasyonda kalan kapasite değerinin 1'e eşitlenmesi durumunda gerçekleştirilmektedir. Bir görev için geriye kalan kapasite değerinin 1'e eşitlenmesi durumunda, işlem süreci organizatörü, kapasite matrisi üzerinde işaretçinin işaret ettiği kapasite değerini söz konusu göreve atar.

Başlangıç kapasitesi, son tamamlanma zamanı ve periyot değerleri belirlenen görev kümesi ve Rastgele Kapasite Üreteci ile oluşturulan kapasite matrisi doğrultusunda dinamik işlem süreci organizasyonu algoritmaları test edilmiştir. Test işlemleri aşağıda belirtilen şekilde yürütülmüştür:

- Değişken Kapasiteli EDF (EDF with RC) – Periyodik, Sporadik, Poisson Görev Geliş Modelleri ile
- Sabit Kapasiteli EDF – Periyodik, Sporadik, Poisson Görev Geliş Modelleri ile
- Değişken Kapasiteli LLF (LLF with RC) – Periyodik, Sporadik, Poisson Görev Geliş Modelleri ile
- Sabit Kapasiteli LLF – Periyodik, Sporadik, Poisson Görev Geliş Modelleri ile

Yukarıda belirtilen simülasyon şemalarıyla yapılan test sonuçları Çizelge 4.1 - 4.4'de belirtilmiştir:

Çizelge 4.1 Değişken Kapasiteli EDF uygulama sonuçları

ALG			TÜR: DEĞİŞKEN KAPASİTE (RC)								
EARLIEST DEADLINE FIRST	Geliş Modeli		PERIODIC			SPORADIC			POISSON		
	Parametre		SCH	CS	PR	SCH	CS	PR	SCH	CS	PR
	Sonuç		20%	23	3,2	40%	21,5	3,7	30%	22,1	4,2
	ORTALAMA WCET	G_1	1			1,5			2,2		
		G_2	8,9			9			10,7		
		G_3	9,6			9			12,1		
		G_4	23,4			17,1			26,4		

Çizelge 4.2 Sabit Kapasiteli EDF uygulama sonuçları

ALG			TÜR: SABİT KAPASİTE								
EARLIEST DEADLINE FIRST	Geliş Modeli		PERIODIC			SPORADIC			POISSON		
	Parametre		SCH	CS	PR	SCH	CS	PR	SCH	CS	PR
	Sonuç		100%	31	2	100%	28	6,6	30%	24,7	7,2
	ORTALAMA WCET	G_1	1			1			1		
		G_2	4			4			8		
		G_3	8			7,7			19,2		
		G_4	31			19,4			36,7		

Çizelge 4.3 Değişken Kapasiteli LLF uygulama sonuçları

ALG	TÜR: DEĞİŞKEN KAPASİTE (RC)									
LEAST LAXITY FIRST	**Geliş Modeli**		**PERIODIC**			**SPORADIC**			**POISSON**	
	Parametre		**SCH**	**CS**	**PR**	**SCH**	**CS**	**PR**	**SCH**	**CS**	**PR**
	Sonuç		20%	25	3,8	20%	21,9	3,2	20%	22,7	7,7
	ORTALAMA WCET	G1	1			1,3			1,7		
		G2	9,9			8,9			10,4		
		G3	9,6			9,6			14,3		
		G4	23,3			17,4			24,2		

Çizelge 4.4 Sabit Kapasiteli LLF uygulama sonuçları

ALG	TÜR: SABİT KAPASİTE									
LEAST LAXITY FIRST	**Geliş Modeli**		**PERIODIC**			**SPORADIC**			**POISSON**	
	Parametre		**SCH**	**CS**	**PR**	**SCH**	**CS**	**PR**	**SCH**	**CS**	**PR**
	Sonuç		100%	35	6	100%	27,6	5,7	0%	24,7	6,6
	ORTALAMA WCET	G1	1			1			1		
		G2	4			4			8,5		
		G3	14			8			21,8		
		G4	31			22,1			33,5		

Elde edilen sonuçların analizi aşağıda belirtilmiştir:

1) Her iki algoritma da değişen kapasite durumunda dengesiz sonuçlara sebebiyet vermektedir. Tüm geliş modelleri için elde edilen organize edilebilirlik (SCH) değerleri sabit kapasite durumunda elde edilenlerden daha düşüktür. Çizelge 4.1'de görüldüğü üzere, değişen kapasite durumunda EDF için periyodik görevlerde %20, sporadik görevlerde %40 ve poisson görevlerde %30 organize edilebilirlik değeri elde edilirken, Çizelge 4.2'deki sabit kapasiteli görevlerde bu oranlar periyodik görevler için %100, sporadik görevler için %100 ve poisson görevler için %30'dur. LLF algoritması için de periyodik görevlerde değişen kapasite durumunda periyodik görevlerde %100'e karşılık %20, sporadik görevlerde %100'e karşılık %20 ve poisson görevlerde %0'a karşılık %20'lik organize edilebilirlik oranları elde edilmiştir. Bu sonuç, en kötü durumda gerçekleştirilme zamanlarının (WCET) sabit kapasite durumunda elde edilenlere göre daha yüksek olması nedeniyle gerçekleşmektedir. Bunun yanında her görev için yeni kapasite değerleri atanması işleminin de işlemci kaynaklarına getireceği çalışma zamanı giderlerini de olumsuz bir katkı olarak ilave edebiliriz. Çalışma zamanı giderlerinin hesaplanmasında önemli parametreler olan kontekst anahtarlama ve önceliklendirme sayıları açısından ise kayda değer bir değişim gözlenmemektedir.

2) Periyodik ve sporadik görevler için EDF ve LLF algoritmaları benzer şekilde daha düşük performansla çalışmakla birlikte poisson geliş modelindeki görevler için EDF algoritmasının LLF algoritmasına oranla daha iyi sonuçlar verdiği görülmektedir. Bunun yanında LLF algoritması da değişen kapasite durumunda daha iyi sonuç vermiştir (%0'a karşılık %20).

3) Gecikmelerin daha düzenli olduğu geliş modelleri olan periyodik ve sporadik görev kümeleri için organize edilebilirlik oranının değişen kapasite durumunda neden daha düşük olduğu sorusuna cevap ararken, bunun görev kümesi içerisinde yer alan 1 veya 2 görevin belirlenen en son tamamlanma zamanı öncesinde tamamlanma gereksinimi doğrultusunda, kaynak tahsisi için işlemci kuyruğunda beklerken kendilerinden daha yüksek öncelik değerine sahip olan görevlerin de kaynak tahsisi başvurusunda bulunarak kaynakları kullanmaları sebebiyle kapasitelerinin bir kısmını gecikmeli

olarak tamamlayabildikleri görülmüştür. Aslında görevlerin son tamamlanma zamanından geç tamamlanması problemi statik kapasiteli görev kümesi kullanan sistemler için de geçerlidir. Ancak statik kapasite kullanımı durumunda tasarımcı sistemin toplam kaynak kullanımı oranlarına bakarak görev kümesinin organize edilebilip edilemeyeceğini önceden bilebilirken dinamik kapasite kullanımı esnasında bunun tahmin edilebilmesi mümkün değildir.

4) Organize edilebilir bir görev kümesine arıza kaldırılabilirlik özelliğinin kazandırılabilmesi için kapasite artırımına ihtiyaç duyulmaktadır. Değişen kapasite kullanımı, kapasite artırımını sağlayabilecek özellikte olmasına karşın, kaynak kullanımı ile ilgili herhangi bir sınırlayıcı etkiye sahip olmadığından, belirli zamanlarda var olanın üzerinde oluşabilecek kaynak talepleri yüzünden sistemin organize edilemez hale gelmesine sebebiyet verebilmektedir. Sabit kapasiteli sistemlerde ise kaynak kullanım oranı zamanla değişim göstermemekte, bu ise görev setinin düşük kaynak kullanım oranına sahip olması durumunda işlemci zamanının boşa sarf edilmesine sebebiyet vermektedir.

Bu sonuçlar doğrultusunda bahsedilen problemlerin üstesinden gelebilmek ve kapasite güncelleme işlemini daha tahmin edilebilir hale getirmek maksadıyla kapasite artırma ve azaltma işlemini bir takım kurallar doğrultusunda gerçekleştiren bir kapasite değişim sistemi üzerinde çalışılarak daha dengeli işlem süreci organizasyonları sağlayan ve kaynak kullanım oranını işlemci kaynaklarının daha etkin kullanımı ile artıran bir sistem tasarlanmıştır.

Bu sistem ile görevlerin en son tamamlanma zamanını kaçırma riski azaltılarak organize edilebilirlik artırılmış, aynı zamanda daha yüksek kaynak kullanım oranlarına çıkılması sağlanmış ve esnek bir kapasite güncelleme sistemi oluşturulmuştur.

4.2 Kapasite Değişimi ile Dengeli ve Sınırlandırılmış Kaynak Kullanımı Sağlayan İşlem Süreci Organizatörü (SBU)

Değişken kapasite kullanımında karşılaşılan sorunların çözümü ve görev kümesine arıza kaldırılabilirlik özelliğinin ilave edilmesi gerekliliği doğrultusunda geliştirilen algoritma ve tasarlanan sistem ile

1) Belirli zamanlarda anlık aşırı yüklenmelere sebebiyet verebilecek kapasite artırımlarının sınırlandırılması,
2) Görevlerin en son tamamlanma zamanını kaçırma riskinin azaltılarak organize edilebilirlik değerinin artırılması,
3) %100'ün altında başlangıç kaynak kullanımına sahip sistemler için organize edilebilirliği olumsuz yönde etkilemeden kaynak kullanımının dereceli olarak artırılması hedeflenmiştir.

Kapasite değişimi ile dengeli ve sınırlandırılmış kaynak kullanımını sağlamak maksadıyla geliştirilen SBU algoritması önceliklendirme işleminde EDF ve LLF dinamik işlem süreci organizasyonu algoritmaları ile birlikte kullanılacak şekilde geliştirilmiştir (Uygulamanın kaynak kodları **EK 2**'de sunulmuştur).

Algoritma, yukarıda belirtilen temel periyot (P), işlemci kullanım oranı (U), çalışma yükü (W), işlemci boş zamanı (B), kaynak ihtiyacı (WRK) parametreleri ile kalan kapasite (RoC) parametrelerinden yararlanmaktadır. Algoritmanın çalışma prensibi aşağıda belirtilmiştir:

Parametreler: *i (i Є [1, n]), P, U, W(t), B(t), Wrk(t), RoC(Rest of Capacity)*

Start:

For *all tasks in the task set*

If *the capacity of the task is greater than 1*

If *Rest of Capacity (RoC) for the task equals to 1*

If *the Resource Need (Wrk) for the task is greater than 0*

Backoff:

If *the Available Time(B) is greater than or equals to Resource Need(Wrk)*

If *[Available Time(B) – Resource Need(Wrk)] is greater than half of the Resource Need(Wrk)*

Increment Ci;

Else

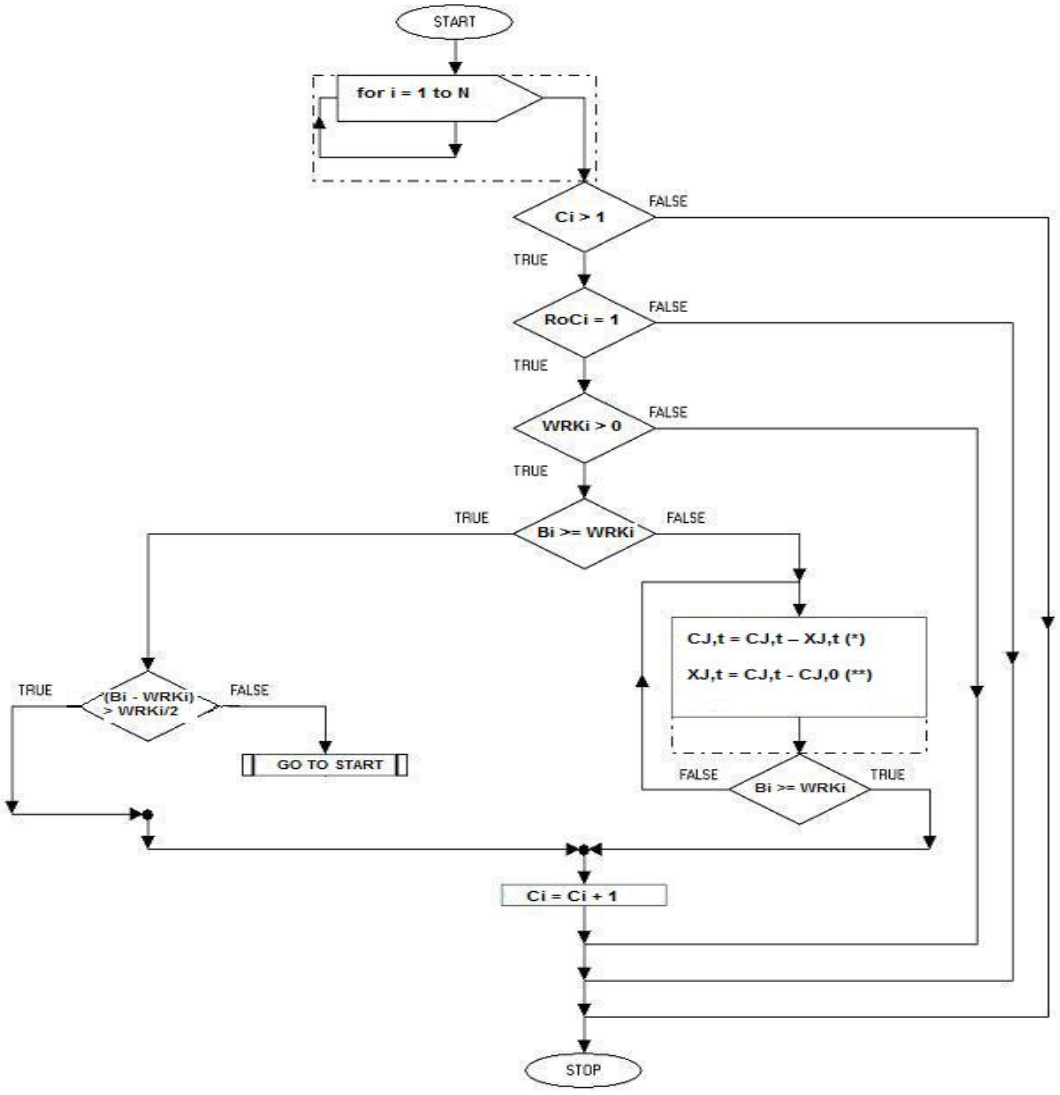
Go to Start;

Else

Decrement Cn (n is the other tasks in the task set)
until they reach back to their initial values;
Go to Backoff;

End for

Algoritmaya ait akış diyagramı Şekil 4.1 ‘de belirtilmiştir:



Şekil 4.1 Güncellenen Kapasite Durumunda SBU Algoritması Blok Diyagramı

(*) : {J : 1,2,…i-1,i+1….N } N : Toplam Görev Sayısı
(**) : (Başlangıç anına göre J’nci görev için artırılmış kapasite miktarı)

Algoritmada işlemci boş zamanı (B) parametresi simülasyon anı ile simülasyon bitiş zamanı arasında hesaplanır. Eğer boş zaman görevin kapasitesini artırmak için yeterliyse (kaynak ihtiyacının %50 fazlası) artırma işlemi gerçekleştirilirken boş zamanın yeterli olmaması durumunda ise diğer görevler için o ana kadar yapılmış olan kapasite artırım işlemleri geriye alınır ve başlangıç kapasite değerlerine döndürülene kadar azaltılarak gerçekleştirilmeye çalışılan görev için daha fazla kapasite ayrılması sağlanır. Bu işlem, söz konusu görevin katı zaman kısıtı altında dahi ilerideki aktivasyonlarını sorunsuz olarak tamamlamasına yetecek kadar kaynak ayrılması ve bir darboğaza girilmesinin önlenmesini sağlar.

Geliştirilen algoritmanın başlangıç kaynak kullanım oranı %100'ü geçmeyen görev kümelerine uygulanması sonucunda, algoritma doğrultusunda kapasite değerlerinin her simülasyon zamanı için artış ve azalışlar sergilediği gözlemlenmiştir. Görev kapasiteleri için sağlanan bu artış azalış mekanizması, bir yandan dinamik ve tek işlemcili arıza kaldırılabilir sistemler için kapasite güncelleme işleminin gerçekleştirilmesini sağlarken, diğer taraftan daha dengeli ve %100'le sınırlandırılmış bir kapasite kullanım oranına imkân vermekte ve organize edilebilirlik değerini artırmaktadır.

Algoritmanın sağladığı önemli bir diğer husus ise görev kapasitelerinin asla başlangıç kapasite değerlerinin altına düşmemesi ve kaynak yetersizliği durumunda yalnızca daha evvel artırılmış kapasite miktarlarının geri alınarak başlangıç değerlerine dönülmesidir. Geliştirilen değişen kapasite durumunda SBU algoritmasının (başlangıç değerleri olarak Çizelge 3.1'de belirtilen görev kümesi değerleri kullanılarak) EDF ve LLF dinamik işlem süreci algoritmaları ile birlikte tatbiki sonucunda elde edilen değerler Çizelge 4.5 ve 4.6'da belirtilmiştir:

Çizelge 4.5 Dengeli ve Sınırlandırılmış (SBU) EDF Algoritması test sonuçları

ALG			STEADY AND BOUNDED UTILIZATION (SBU)								
EARLIEST DEADLINE FIRST	Geliş Modeli		PERIODIC			SPORADIC			POISSON		
	Parametre		SCH	CS	PR	SCH	CS	PR	SCH	CS	PR
	Sonuç		100%	63	4	90%	52,4	10,6	10%	50,2	9,4
	ORTALAMA WCET	G1	1			2,4			3		
		G2	4			5,1			14,4		
		G3	8			8			25,3		
		G4	31			24,2			41,7		

Çizelge 4.6 Dengeli ve Sınırlandırılmış (SBU) LLF Algoritması test sonuçları

ALG			STEADY AND BOUNDED UTILIZATION (SBU)								
LEAST LAXITY FIRST	Geliş Modeli		PERIODIC			SPORADIC			POISSON		
	Parametre		SCH	CS	PR	SCH	CS	PR	SCH	CS	PR
	Sonuç		100%	74	14	80%	52,7	12,7	0%	74	28,9
	ORTALAMA WCET	G1	1			3,6			3,1		
		G2	6			5			15,1		
		G3	15			9			24,2		
		G4	31			20,4			37,7		

Tasarlanan algoritma kullanılarak geliştirilen işlem süreci organizatörleri kullanılarak elde edilen bu sonuçların analizi aşağıdaki maddelerde belirtilmiştir:

1) SBU işlem süreci organizasyon şeması kullanılarak elde edilen organize edilebilirlik değerleri, periyodik ve sporadik geliş modelleri için, değişken kapasiteli (RC) dinamik işlem süreci organizatörleriyle elde edilen organize edilebilirlik değerlerinden daha yüksektir. Çizelge 4.5'te görüldüğü gibi SBU-EDF organizatörü ile elde edilen organize edilebilirlik değerleri periyodik görevler için %100, sporadik görevler için ise %90'dır. Bu sonuçlar, aynı görev kümesi kullanılarak RC-EDF organizatörüyle elde edilen (Çizelge 4.1) periyodik görevler için %20 ve sporadik görevler için %40 organize edilebilirlik değerlerinin çok üzerindedir. Poisson görev geliş modelleri için ise RC-EDF organizatörüyle %30'luk bir organize edilebilirlik değerine ulaşılmışken SBU-EDF organizatörüyle %10'luk bir başarı sağlanmıştır.

LLF algoritması için sonuçları irdelersek, periyodik ve sporadik geliş modelleri için RC-LLF organizatörüyle sırasıyla %20 ve %20'lik organize edilebilirlik değerleri sağlanırken aynı geliş modellerinde SBU-LLF organizatörü kullanılarak sırasıyla %100 ve %80'lik bir organize edilebilirlik değerine ulaşılmıştır. Çizelge 4.3 ve 4.6'da poisson geliş modelleriyle elde edilen organize edilebilirlik değerleri incelendiğinde ise RC-LLF organizatörüyle %20'lik bir başarı sağlanırken SBU-LLF organizatörü %0'lık bir organize edilebilirlik değeri sağlamıştır.

2) SBU-EDF ve SBU-LLF işlem süreci organizasyon şemaları dinamik olarak gerçekleştirdiği kapasite artış ve azalışlarıyla periyodik ve sporadik görevler için yüksek oranda organize edilebilirlik sağlamaktadır.

3) Dinamik işlem süreci organizatörleri (EDF ve LLF) ile birlikte kullanılan SBU işlem süreci organizasyon şeması poisson görev geliş modelleri için bir çözüm sağlamamaktadır.

4) Yapılan testlerin doğruluk seviyesini artırmak maksadıyla söz konusu işlem tekrar sayıları her bir geliş modeli için 100'e çıkarılarak tekrarlanmış ve tekrar sayılarının artırılması sonucunda aşağıda belirtilen organize edilebilirlik değerlerine ulaşılmıştır:

Değişken Kapasiteli EDF: Periyodik: %27 , Sporadik: %39 , Poisson: 28

Değişken Kapasiteli LLF: Periyodik: %22 , Sporadik: %24 , Poisson: 18

Dengeli ve Sınırlandırılmış (SBU) EDF: %98 , Sporadik: %91, Poisson: %16

Dengeli ve Sınırlandırılmış (SBU) LLF: % 99 , Sporadik: %87, Poisson: %9

5) Başlangıç kaynak kullanım oranı sabit tutulmak suretiyle sistemdeki görev sayısı kademeli olarak (Görev sayısı: 5,6,...10) artırılarak söz konusu testler yinelenmiş kontekst anahtarlama ve önceliklendirme değerlerinde ciddi bir artış gözlenmekle beraber organize edilebilirlik değerlerinde kayda değer bir değişim olmadığı ve yüksek organize edilebilirlik değerlerine ulaşıldığı teyit edilmiştir. Elde edilen sonuçlar aşağıda belirtilmiştir:

Organize edilebilirlik: 4 görev için elde edilen organize edilebilirlik değeri başlangıç kaynak kullanım oranı sabit tutulmak kaydıyla 5,6,…10 görev için tekrarlanan simülasyonlar sonucunda kayda değer bir değişim göstermemiştir (Periyodik görevler için ortalama %96,7, sporadik görevler için ortalama %89,6, poisson görevler için %11,2).

Kontekst anahtarlama ve önceliklendirme: Görev kümesi içerisindeki görev sayılarının artırılması sonucunda sistem kaynaklarını paylaşan görev sayısının artması nedeniyle kontekst anahtarlama ve preemption sayılarında beklenen ve belirgin bir artış gözlemlenmiştir:

4 görev için Dengeli ve Sınırlandırılmış (SBU) EDF [Context Switch (Preemption)]: Periyodik: 63 (4), Sporadik: 52,4 (10,6), Poisson: 50,2 (9,4)

4 görev için Dengeli ve Sınırlandırılmış (SBU) LLF: Periyodik: 74 (14) , Sporadik: 52,7 (12,7), Poisson: 74 (28,9)

5 görev için Dengeli ve Sınırlandırılmış (SBU) EDF: Periyodik: 75,2 (5,7) , Sporadik: 57,9 (13,9), Poisson: 51,5 (14,8)

5 görev için Dengeli ve Sınırlandırılmış (SBU) LLF: 91,1 (19,2), Sporadik: 67,4 (16,0), Poisson: 92,2 (36,5)

6 görev için Dengeli ve Sınırlandırılmış (SBU) EDF: Periyodik: 82,3 (7,7) , Sporadik: 64,0 (17,1), Poisson: 58,1 (17,2)

6 görev için Dengeli ve Sınırlandırılmış (SBU) LLF: 104,6 (23,6), Sporadik: 73,1 (19,8), Poisson: 103,1 (41,8)

7 görev için Dengeli ve Sınırlandırılmış (SBU) EDF: Periyodik: 89,0 (10,1) , Sporadik: 72,7 (20,3), Poisson: 63,3 (19,9)

7 görev için Dengeli ve Sınırlandırılmış (SBU) LLF: 111,6 (26,1), Sporadik: 79,2 (24,0), Poisson: 108,8 (44,4)

8 görev için Dengeli ve Sınırlandırılmış (SBU) EDF: Periyodik: 101.3 (13,9), Sporadik: 84,2 (22,1), Poisson: 78,1 (23,0)

8 görev için Dengeli ve Sınırlandırılmış (SBU) LLF: 111,6 (30,5), Sporadik: 90,8 (27,1), Poisson: 117,1 (48,6)

9 görev için Dengeli ve Sınırlandırılmış (SBU) EDF: Periyodik: 110,1 (17,2), Sporadik: 93,6 (25,6), Poisson: 90,5 (27,8)

9 görev için Dengeli ve Sınırlandırılmış (SBU) LLF: 122,1 (34,8), Sporadik: 103,9 (32,3), Poisson: 130,2 (50,2)

10 görev için Dengeli ve Sınırlandırılmış (SBU) EDF: Periyodik: 119,5 (20,7), Sporadik: 101,1 (27,8), Poisson: 97,8 (29,9)

10 görev için Dengeli ve Sınırlandırılmış (SBU) LLF: 131,2 (38,6), Sporadik: 114,2 (20,5), Poisson: 136,4 (54,6)

6) Sistemde meydana gelebilecek hataların gözlemlenmesi maksadıyla sisteme aperiyodik görevler ilave edilmesi durumunda sistemin davranışları analiz edilmiştir. Rastgele kapasiteli ve SBU algoritmasının uygulandığı görev kümesinin bulunduğu sisteme, önceliği diğer görevlerden daha yüksek olan aperiyodik görevlerin ilave edilmesi durumunda sistemde organize edilebilirlik değerinde bir düşüş gözlemlenmiştir. Düşüşün nedeninin daha yüksek öncelikli aperiyodik görevin diğer görevlerden daha önce organize edilmesi ve görev kümesi içerisindeki görevlerin belirlenen son tamamlanma zamanı dahilinde organize edilememesi olduğu sonucuna varılmıştır. Her iki algoritma ile benzer sonuçlar elde edilmiştir.

Yukarıda belirtilen testlerle elde edilen sonuçlar haricinde geliştirilen algoritma kullanılarak elde edilen bir diğer önemli sonuç da, statik sistemlerde kullanılmayarak atıl durumda bırakılan kaynakların daha etkin kullanılması ve daha yüksek kaynak kullanım oranlarının sağlanmasıdır. Gerçek zamanlı ve dinamik bir sisteme arıza kaldırılabilirlik özelliğinin kazandırılması açısından hayati bir öneme haiz olan bu sonucu daha iyi açıklayabilmek için başlangıçta %75 kaynak kullanımı sağlayan ve Çizelge 4.7'de özellikleri belirtilen görev kümesi kullanılmıştır.

4.3 TEST 1 : %75 Kaynak Kullanım Oranı

Çizelge 4.7 SBU Algoritmasının sağladığı kaynak kullanım oranının artırılması özelliğinin testi için üretilen görev kümesi (%75 Kaynak Kullanım Oranı)

Görev	**Periyot**	**Kapasite**	**Kullanım Oranı (U)**
G1	4	1	0,25
G2	8	2	0,25
G3	16	2	0,125
G4	32	4	0,125
Toplam Kaynak Kullanım Oranı			0.75

Çizelge 4.7'de belirtilen görev kümesi için temel periyot değeri (LCM (4, 8, 16, 32)) 32'dir ve simülasyon aralığı olarak temel periyot değerinin katları olan 32, 64, 128, 256 ve 512 değerleri kullanılmıştır.

Belirtilen görev kümesinin EDF ve LLF algoritmalarının kullanıldığı işlem süreci organizatörleri ile elde edilen kaynak kullanım oranı %75'dir ve %25'lik kullanılmayan (ID) işlemci zamanı kalmıştır. Organize edilebilirlik değeri %100'dür.

Organize edilebilirlik değerinin aynı kalacağı ve daha yüksek kaynak kullanım oranlarının sağlanabileceği bir organizasyon şemasının elde edilmesi amaçlanmıştır. Çizelge 4.8, Şekil 4.2 ve Şekil 4.3'de SBU algoritması kullanılarak elde edilen sonuçlar görülmektedir.

Çizelge 4.8 SBU-EDF ve SBU-LLF İşlem Süreci Organizatörleri kullanılarak elde edilen kapasite ve kaynak kullanım oranı artışı (Başlangıç U: 0,75)

	SIM	ID	U	CS	PR
SBU EDF	32	2	93,7	18	4
	64	3	95,3	34	5
	128	5	96,1	66	7
	256	10	96,1	127	9
	512	18	96,5	255	17

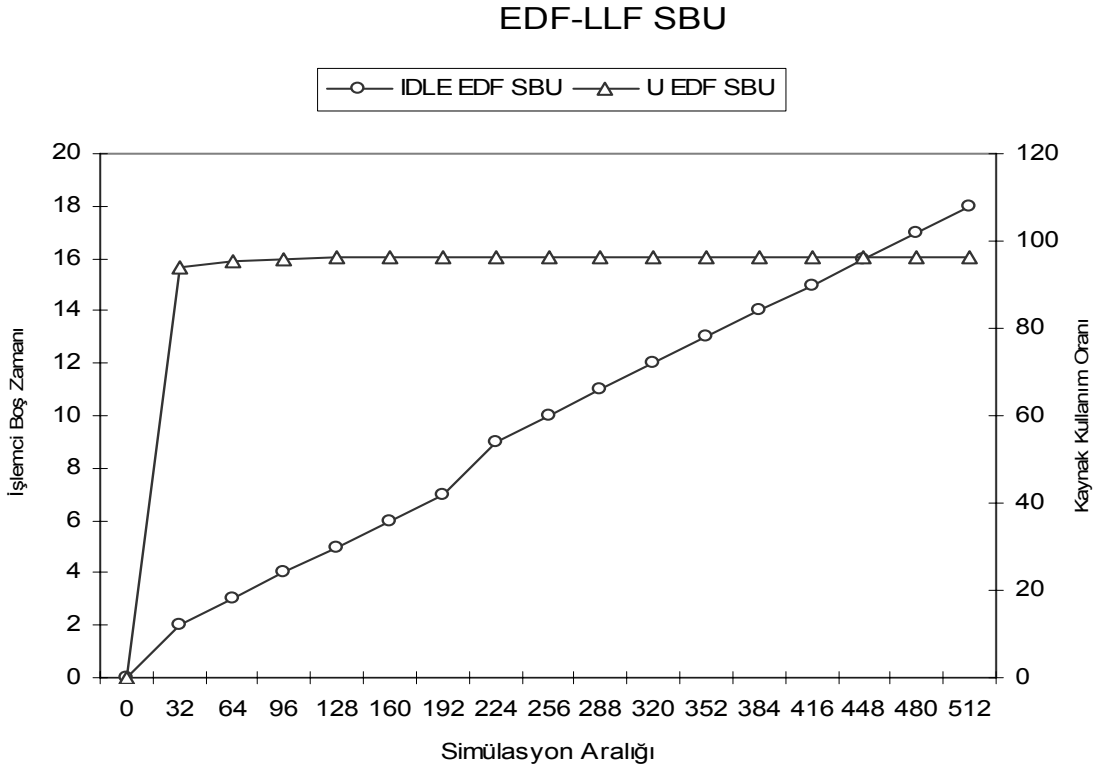
	SIM	ID	U	CS	PR
SBU LLF	32	2	93,7	18	4
	64	3	95,3	39	7
	128	5	96,1	75	10
	256	10	96,1	144	11
	512	18	96,5	288	19

	SIM	ID	U	CS	PR
EDF	32	8	75,0	18	4
	64	16	75,0	36	8
	128	32	75,0	72	16
	256	64	75,0	144	32
	512	128	75,0	288	64

	SIM	ID	U	CS	PR
LLF	32	8	75,0	18	4
	64	16	75,0	36	8
	128	32	75,0	72	16
	256	64	75,0	144	32
	512	128	75,0	288	64

Çizelge 4.8'de görüldüğü üzere SBU algoritmasının EDF ve LLF algoritmaları ile birlikte kullanılması durumunda amaçlanan sonuçlara ulaşılmış ve %100 organize edilebilirlik değeri yanında %75'e karşılık %90'ın üzerinde kaynak kullanım oranına ulaşılmıştır. Simülasyon aralığı arttıkça sistemdeki kaynak kullanım oranının (U) artış hızı azalmakta bu ise organize edilebilirlik değerinin %100'ün altına düşmesini önlemektedir. Tasarlanan sistem kullanılarak kaynak kullanım oranı artırılırken önceliklendirme sayılarında belirgin bir azalma gözlenmektedir (EDF ve LLF organizatörleri için PR: 64 iken SBU-EDF ve SBU-LLF organizatörleri için PR: 17-19'dur).

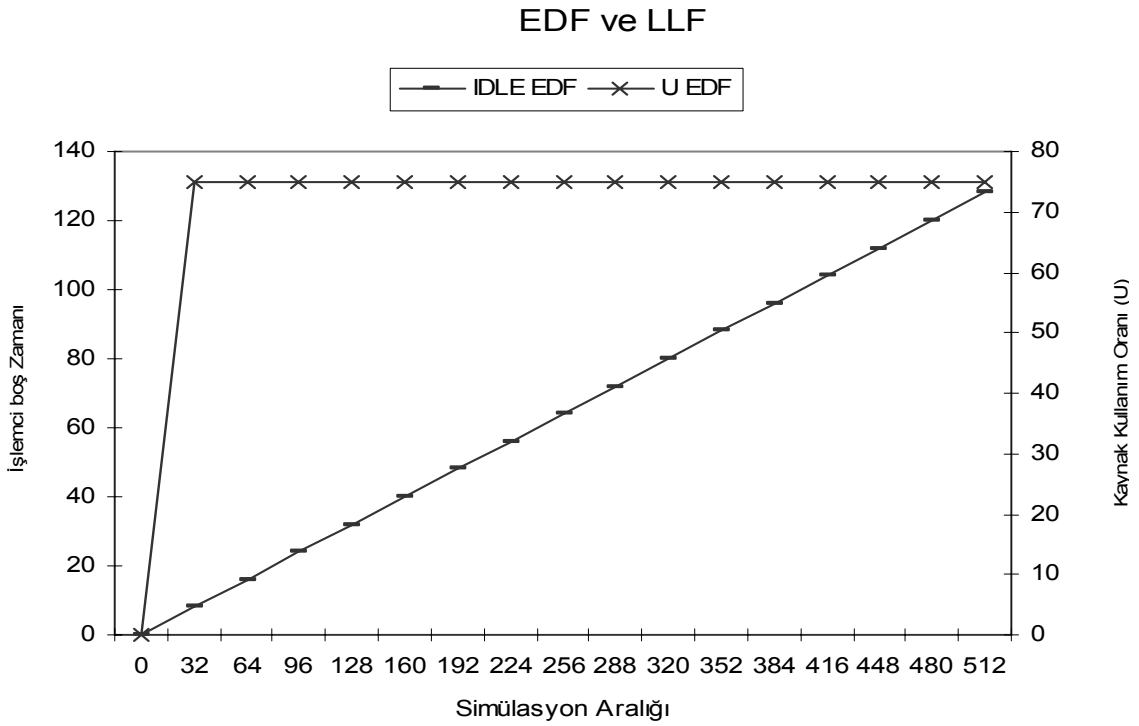
Şekil 4.2'de başlangıçta %75 kaynak kullanım oranına sahip olan bir görev kümesi için SBU algoritması kullanılarak kaynak kullanım oranında elde edilen artış ve işlemci boş zamanı için elde edilen değerler görülmektedir. İşlemci boş zamanındaki artış simülasyon aralığındaki artıştan daha azdır, dolayısıyla sistemde kaynak kullanım oranında bir artış elde edilebilmektedir.



Şekil 4.2 EDF-SBU ve LLF-SBU İşlem Süreci Organizatörleri ile Elde Edilen İşlemci Boş Zamanı ve Kaynak Kullanım Oranı (Başlangıç: U=0,75)

(Not: Her iki organizatör (EDF-SBU, LLF-SBU) de aynı sonuçları vermiştir)

Şekil 4.3'de ise aynı görev kümesi için kaynak kullanım oranında bir artıştan söz edilememektedir. İşlemci boş zamanındaki artış simülasyon aralığındaki artışla aynıdır, dolayısıyla sistemde kaynak kullanım oranında bir artış söz konusu değildir.



Şekil 4.3 EDF ve LLF İşlem Süreci Organizatörleri ile Elde Edilen İşlemci Boş Zamanı ve Kaynak Kullanım Oranı (Başlangıç: U=0,75)

(Not: Her iki organizatör (EDF, LLF) de aynı sonuçları vermiştir)

Önerilen işlem süreci organizasyon şemasının başlangıçta %25 ve %50 kaynak kullanım oranlarına sahip olan görev kümeleri için test edilerek daha düşük kaynak kullanım oranları için artışın nasıl bir gelişim gösterdiğinin gözlenebilmesi amaçlanmıştır. Bu kapsamda aşağıda belirtilen işlemler gerçekleştirilmiştir:

4.4 TEST 2: %25 Kaynak Kullanım Oranı

Çizelge 4.9 SBU Algoritmasının sağladığı kaynak kullanım oranının artırılması özelliğinin testi için üretilen görev kümesi (%25 Kaynak Kullanım Oranı)

Görev	**Periyot**	**Kapasite**	**Kullanım Oranı (U)**
G1	16	1	0,0625
G2	32	2	0,0625
G3	32	2	0,0625
G4	64	4	0,0625
Toplam Kaynak Kullanım Oranı			0.25

Çizelge 4.9'de belirtilen görev kümesi için temel periyot değeri (LCM (16, 32, 32, 64)) 64'tür ve simülasyon aralığı olarak temel periyot değerinin katları olan 64, 128, 256 ve 512 değerleri kullanılmıştır.

Belirtilen görev kümesinin EDF ve LLF algoritmalarının kullanıldığı işlem süreci organizatörleri ile elde edilen kaynak kullanım oranı %25'tir ve %75'lik kullanılmayan (ID) işlemci zamanı kalmıştır. Organize edilebilirlik değeri %100'dür.

Organize edilebilirlik değerinin aynı kalacağı ve daha yüksek kaynak kullanım oranlarının sağlanabileceği bir organizasyon şemasının elde edilmesi amaçlanmıştır. Çizelge 4.10, Şekil 4.4 ve Şekil 4.5'de SBU algoritması kullanılarak elde edilen sonuçlar görülmektedir.

Çizelge 4.10 SBU-EDF ve SBU-LLF İşlem Süreci Organizatörleri Kullanılarak Elde Edilen Kapasite ve Kaynak Kullanım Oranı Artışı (Başlangıç U: 0,25)

	SIM	ID	U	CS	PR
SBU EDF	64	46	28,1	7	0
	128	83	35,2	14	0
	256	129	49,6	33	3
	512	172	66,4	78	12

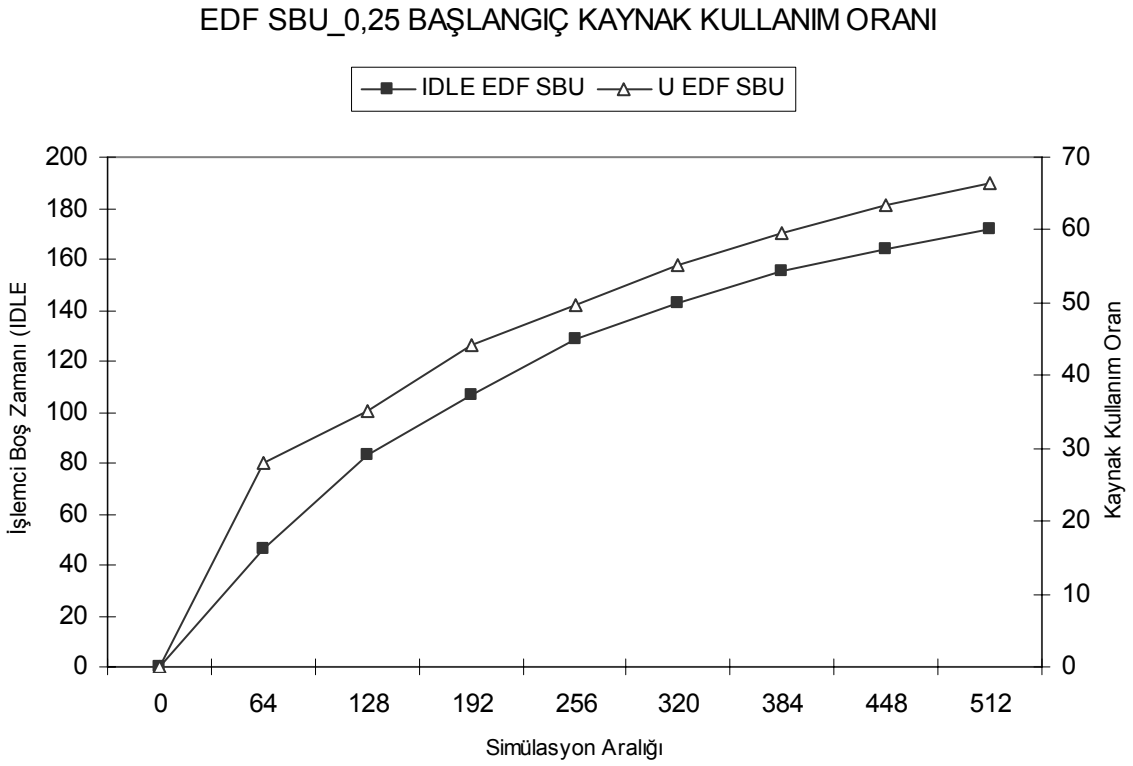
	SIM	ID	U	CS	PR
SBU LLF	64	44	31,3	15	6
	128	71	44,5	48	27
	256	91	64,5	138	92
	512	129	74,8	311	215

	SIM	ID	U	CS	PR
EDF	64	48	25	7	0
	128	96	25	14	0
	256	192	25	28	0
	512	384	25	56	0

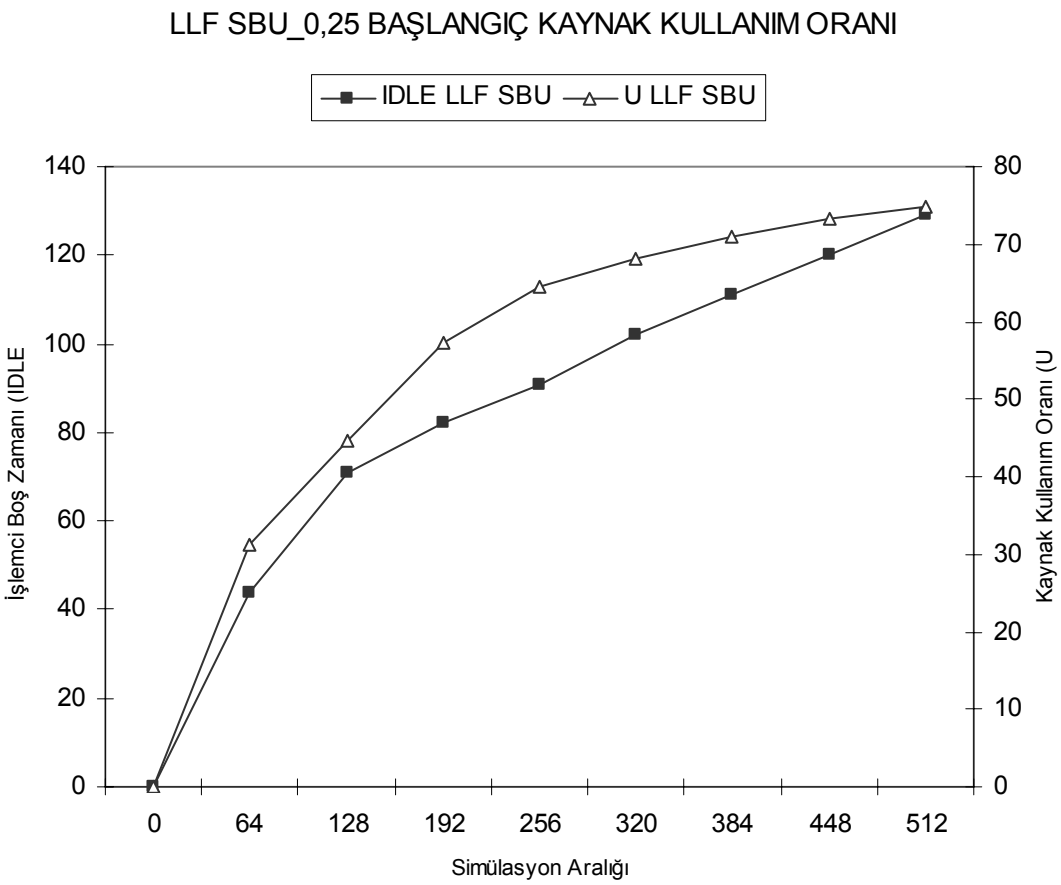
	SIM	ID	U	CS	PR
LLF	64	48	25	11	4
	128	96	25	22	8
	256	192	25	44	16
	512	384	25	88	32

Çizelge 4.10'da görüldüğü üzere SBU algoritmasının EDF ve LLF algoritmaları ile birlikte kullanılması durumunda amaçlanan sonuçlara ulaşılmış ve %100 organize edilebilirlik değeri yanında %25'e karşılık %70 civarında kaynak kullanım oranına ulaşılmıştır. Simülasyon aralığı arttıkça sistemdeki kaynak kullanım oranının (U) artış hızı azalmakta bu ise organize edilebilirlik değerinin %100'ün altına düşmesini önlemektedir. Tasarlanan sistem kullanılarak kaynak kullanım oranı artırılırken önceliklendirme ve kontekst anahtarlama parametrelerinde EDF algoritması için belirgin bir değişiklik söz konusu olmamakla birlikte LLF algoritması için ciddi bir artış söz konusudur. Bu ise *ertelenebilirlik eşitliği (laxity tie)* adı verilen durum sebebiyle oluşmakta ve sistem kaynakları aynı öncelik değerine sahip iki görev arasında görevler tamamlanana kadar değiş tokuş edilmektedir. (SBU-LLF için 0-512 simulasyon aralığında gerçekleştirilen CS : 311 , PR : 215 iken LLF için aynı aralıkta CS : 88, PR : 32'dir).

Şekil 4.4 ve Şekil 4.5'te başlangıçta %25 kaynak kullanım oranına sahip olan bir görev kümesi için EDF-SBU ve LLF-SBU algoritmaları kullanılarak kaynak kullanım oranında elde edilen artış ve işlemci boş zamanı için elde edilen değerler görülmektedir. İşlemci boş zamanındaki artış simülasyon aralığındaki artıştan daha azdır, dolayısıyla sistemde kaynak kullanım oranında bir artış elde edilebilmektedir.



Şekil 4.4 EDF-SBU İşlem Süreci Organizatörü ile Elde Edilen İşlemci Boş Zamanı ve Kaynak Kullanım Oranı (Başlangıç: U=0,25)



Şekil 4.5 LLF-SBU İşlem Süreci Organizatörü ile elde edilen işlemci boş zamanı ve kaynak kullanım oranı (Başlangıç: U=0,25)

4.5 TEST 3: %50 Kaynak Kullanım Oranı

Çizelge 4.11 SBU Algoritmasının sağladığı kaynak kullanım oranının artırılması özelliğinin testi için üretilen görev kümesi (%50 Kaynak Kullanım Oranı)

Görev	**Periyot**	**Kapasite**	**Kullanım Oranı (U)**
G1	8	1	0.125
G2	16	2	0,125
G3	16	2	0,125
G4	32	4	0,125
Toplam Kaynak Kullanım Oranı			0.50

Çizelge 4.11'de belirtilen görev kümesi için temel periyot değeri (LCM (8, 16, 16, 32)) 32'dir ve simülasyon aralığı olarak temel periyot değerinin katları olan 32, 64, 128, 256 ve 512 değerleri kullanılmıştır.

Belirtilen görev kümesinin EDF ve LLF algoritmalarının kullanıldığı işlem süreci organizatörleri ile elde edilen kaynak kullanım oranı %50'dir ve %50'lik kullanılmayan (ID) işlemci zamanı kalmıştır. Organize edilebilirlik değeri %100'dür.

Organize edilebilirlik değerinin aynı kalacağı ve daha yüksek kaynak kullanım oranlarının sağlanabileceği bir organizayon şemasının elde edilmesi amaçlanmıştır. Çizelge 4.12, Şekil 4.6 ve Şekil 4.7'de SBU algoritması kullanılarak elde edilen sonuçlar görülmektedir.

Çizelge 4.12 SBU-EDF ve SBU-LLF İşlem Süreci Organizatörleri Kullanılarak Elde Edilen Kapasite ve Kaynak Kullanım Oranı Artışı (Başlangıç: U= 0,50)

	SIM	ID	U	CS	PR
SBU EDF	32	14	56,25	9	1
	64	17	73,44	19	2
	128	21	83,60	43	4
	256	29	88,67	91	4
	512	45	91,21	187	4

	SIM	ID	U	CS	PR
SBU LLF	32	12	62,50	17	7
	64	13	79,69	47	23
	128	16	87,50	93	44
	256	25	90,23	201	95
	512	41	91,99	417	199

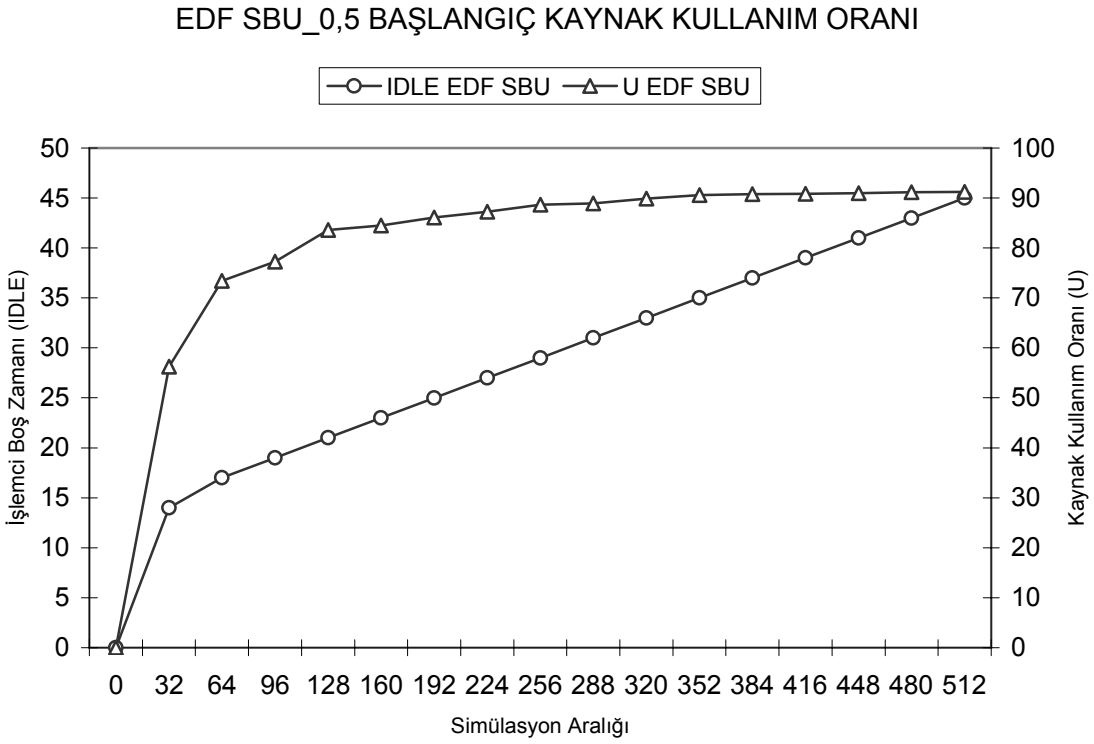
	SIM	ID	U	CS	PR
EDF	32	16	50	9	1
	64	32	50	18	2
	128	64	50	36	4
	256	128	50	72	8
	512	256	50	144	16

	SIM	ID	U	CS	PR
LLF	32	16	50	13	5
	64	32	50	26	10
	128	64	50	52	20
	256	128	50	104	40
	512	256	50	208	80

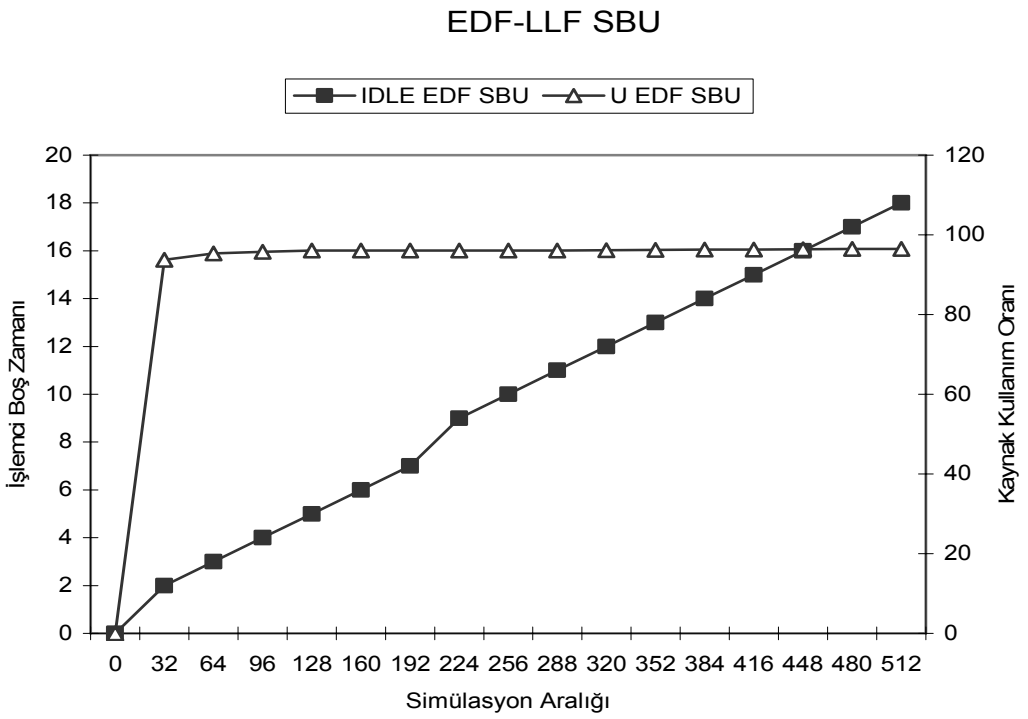
Çizelge 4.12'de görüldüğü üzere SBU algoritmasının EDF ve LLF algoritmaları ile birlikte kullanılması durumunda amaçlanan sonuçlara ulaşılmış ve %100 organize edilebilirlik değeri yanında %50'ye karşılık %90 civarında kaynak kullanım oranına ulaşılmıştır. Simülasyon aralığı arttıkça sistemdeki kaynak kullanım oranının (U) artış hızı azalmakta bu ise organize edilebilirlik değerinin %100'ün altına düşmesini önlemektedir. Tasarlanan sistem kullanılarak kaynak kullanım oranı artırılırken önceliklendirme ve kontekst anahtarlama parametrelerinde EDF algoritması için belirgin bir değişiklik söz konusu olmamakla birlikte LLF algoritması için ciddi bir artış söz konusudur. Bu ise *ertelenebilirlik eşitliği (laxity tie)* adı verilen durum sebebiyle oluşmakta ve sistem

kaynakları aynı öncelik değerine sahip iki görev arasında görevler tamamlanana kadar değiş tokuş edilmektedir. (SBU-LLF için 0-512 simulasyon aralığında gerçekleştirilen CS : 417 , PR : 199 iken LLF için aynı aralıkta CS : 208, PR : 80'dir).

Şekil 4.6 ve 4.7'de başlangıçta %50 kaynak kullanım oranına sahip olan bir görev kümesi için EDF-SBU ve LLF-SBU algoritmaları kullanılarak kaynak kullanım oranında elde edilen artış ve işlemci boş zamanı için elde edilen değerler görülmektedir. İşlemci boş zamanındaki artış simülasyon aralığındaki artıştan daha azdır, dolayısıyla sistemde kaynak kullanım oranında bir artış elde edilebilmektedir.



Şekil 4.6 EDF-SBU İşlem Süreci Organizatörü ile Elde Edilen İşlemci Boş Zamanı ve Kaynak Kullanım Oranı (Başlangıç: U=0,50)



Şekil 4.7 LLF-SBU İşlem Süreci Organizatörü ile Elde Edilen İşlemci Boş Zamanı ve Kaynak Kullanım Oranı (Başlangıç: U=0,50)

5. SONUÇLAR VE DEĞERLENDİRME

Katı kısıtlara sahip gerçek zamanlı sistemlerde karşılaşılabilecek dinamik durumlarla baş edebilmek için bir takım yöntemlerin geliştirilmesi gerekmektedir. Dinamik sistemlerin ihtiyaçlarına cevap verebilmek için sistemde yer alan tüm parametrelerin dikkate alınması ve gerçek dünya problemlerinin taleplerine cevap aranması gerekmektedir.

Bu tez çalışmasında, gerçek dünya sistemlerinin kapasite güncelleme taleplerine cevap verebilmek ve tek işlemcili sistemlerde yazılımsal arıza kaldırılabilirliği tesis edebilmek için kapasite parametresi üzerinde yoğunlaşılmıştır. Uygun bir görev kümesi üretilerek EDF ve LLF dinamik işlem süreci organizasyonu algoritmalarının değişen kapasite durumunda sergilediği davranışlar incelenmiş ve sistemin dengesiz ve organize edilebilirliğin daha düşük olduğu tespit edilmiştir.

Sistemin farklı kapasiteler için dengesiz davranmasını engellemek, organize edilebilirlik değerini artırmak ve tam kullanımın altında kullanıma sahip olan sistemler için daha yüksek oranlarda kaynak kullanımı sağlayarak arıza kaldırılabilirliği tesis edebilme imkânı sağlamak maksadıyla dengeli, organize edilebilirlik değeri daha yüksek ve kaynak kullanım oranını artıracak şekilde kapasite güncelleme işlemi sağlayan işlem süreci organizasyon şeması önerilmiştir.

Önerilen sistemle gerçekleştirilen test ve simülasyon işlemleri sonucunda farklı geliş modelleri için organize edilebilirlik değerini (SCH) ve kaynak kullanım oranını (U) artıran bir organizasyon şeması (SBU), EDF ve LLF dinamik işlem süreci organizasyonu algoritmaları ile birlikte kullanılarak amaçlanan sonuçlara ulaşılmıştır. %100'ün altında kaynak kullanım oranına sahip tüm sistemlerde sistem boş kalma zamanında sağlanan azalmaya paralel olarak kapasite değerlerindeki artışla birlikte kaynak kullanım oranının artırılması sağlanmıştır. Bu ise sistem üzerinde yazılımsal arıza kaldırılabilirliğin sağlanabilmesi için ihtiyaç duyulan bir özelliktir.

Dinamik sistemler araştırmacıların üzerine yoğunlaşabileceği çok sayıda, farklı ve değişkenlik gösterebilen ihtiyaçlara sahiptir. Bu tez çalışması kapsamında üzerinde yoğunlaşılan kapasite güncelleme işlemi; aşama aşama gerçekleştirilen testler ve karşılaşılan sorunların çözümü şeklinde geliştirilerek işlem süreci organizasyon şemalarında kapasite değerinin belirlenmesinde anlık olarak hesaplanan sistem iş yükü, sistem boş zamanı, kaynak ihtiyacı parametreleri kullanılarak kapasitenin artırılması ve azaltılması ile daha yüksek kaynak kullanım oranlarının sağlanmasına imkân vermiştir. Sistem üzerinde belirtilen parametrelerin durumuna göre, organize edilebilirliğin en son tamamlanma zamanının geciktirilmesi nedeniyle düşürülmemesi maksadıyla, kapasite değerlerinde zaman zaman azalma gerçekleştirilmekle birlikte hiçbir zaman başlangıç kapasite değerlerinin altında bir kapasite değeri atanmamaktadır.

Gerçekleştirilen çalışma, katı kısıtlara sahip gerçek zamanlı bir sistemde dinamik olarak ihtiyaç duyulabilecek kapasite güncelleme işleminin gerçekleştirilmesini sağlamakla birlikte önerilen işlem süreci organizasyon şeması ile kapasite değerlerinde ve dolayısıyla sistemin kaynak kullanım oranında bir artış sağlamaktadır. Sabit kapasite yerine kapasite değerlerinde artış sağlayan sistemin, tek işlemcili sistemlere yazılımsal arıza kaldırılabilirlik özelliğinin ilave edilmesinde kullanılabileceği değerlendirilmektedir. Tek işlemcili sistemlerde, donanımsal arıza kaldırılabilirliği sağlayacak şekilde her görevin yedek kopyalarının saklandığı ilave yedek bir işlemci bulunmadığından yedek kopyaların ya da hata oluşması durumunda görevin yeniden aktivasyonunu sağlayacak şekilde artırılmış kapasite değerlerine ihtiyaç vardır. Sistemde kritik görevlerin gerçekleştirilmesi durumunda ise, sistem güvenilirliğinin arıtırılması maksadıyla arıza kaldırılabilirlik özelliğine ihtiyaç duyulmaktadır. Önerilen şema kullanılarak tek işlemcili bir sistem için yazılımsal arıza kaldırılabilirlik özelliğinin ilave edilebileceği ve güvenilirlik değerinin artırılmasının sağlanabileceği bir sistem tesis edilmiştir.

KAYNAKLAR

Lauzac, S., Melhem, R. and Mosse, D. 2003. An Improved Rate-Monotonic Admission Control and Its Applications. IEEE Transactions on Computers, Vol.52 (3).

Marti, P., Villa, R., Fuertes, J. and Fohler, 2001. G. On Real-Time Control Tasks Schedulability.

Marti, P., Fuertes, J., Fohler G., and Ramamritham, K. 2001. Jitter Compensation for Real-Time Control Systems.

Lauzac, S., Melhem, R. and Mosse, D. 1998. Adding Fault-Tolerance to P-Fair Real-Time Scheduling.

Ghosh, S., Melhem, R. and Mosse, D. 1994. Fault-Tolerant Scheduling on a Hard Real-Time Multiprocessor System.

Dong, L., Melhem, R. and Mosse, D. 1998. Time Slot Allocation for Real-Time Messages with Negotiable Distance Constraints.

Alvarez, P., Aydın, H., Mosse, D. and Melhem, R. 2000. Scheduling Optional Computations in Fault-Tolerant Real-Time Systems.

Lauzac, S., Melhem, R. and Mosse, D. 1998. Comparison of Global and Partitioning Schemes for Scheduling Rate Monotonic Tasks on a Multiprocessor.

Zhang, H. and Ferrari, D. 1993. Rate-Controlled Static-Priority Queueing.

Naedele, M. 1999 A Fault-tolerant Scheduling Scheme For Hybrid Tasks In Distributed Real-time Systems IEEE Transactions on Computers

Chen, Y. and Xiong, G. 2002 Imprecise Computation Fault-Tolerant Rate-Monotonic Scheduling. Computer Science and Engineering College Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02) IEEE

Hong, Y. and Goo, H. 2005 A Fault-Tolerant Scheduling Scheme for Hybrid Tasks in Distributed Real-Time Systems Dogguk University Proceedings of the Third IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems(SEUS'05) IEEE 2005

Liu, D. and Lee, Y. 2003 An Efficient Scheduling Discipline for Packet Switching Networks Using Earliest Deadline First Round Robin. Department of Computer Science and Engineering Arizona State University IEEE

Chen, Y. and Xiong, G. 2002 Fault-Tolerant Earliest Deadline First Scheduling with Resource Reclaim. Computer Science and Engineering College Proceedings of the Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02) IEEE

Kim, H., Lee, S. and Jeong, B. 2000 An Improved Feasable Shortest Path Real-Time Fault-Tolerant Scheduling Algorithm. School of Electronics and Information, Kyung Hee University Supprted by Ministry of Information and Communication of Korea, IEEE

Tsuchiya, T., Kakuda, Y. and Kikuno, T. 1995 Fault-Tolerant Scheduling Algorithm for Distributed Real-Time Systems. Department of Information and Computer Sciences Faculty of Engineering Science, Osaka University IEEE

Al-Omari, A., Somani, A. and Manimaran, G. 2001 A New Fault-Tolerant Technique for Improving Schedulability in Multiprocessor Real-Time Systems. Dependable Computing & Networking Lab. Dept. Of Electrical and Computer Engineering IOWA State University IEEE

Satyanarayana, N. and Mall, R. 2001. Fault-Tolerant Scheduling in Distributed Real-Time Systems. PAL IEEE

Peng, X. 2003 Fault-Tolerant Scheduling for Asymmetric Communications Electronic Engineering School of Engineering & Applied Science, Aston University The Institution of Electrical Engineers, Printed and Published by IEEE, Michael Faray House, IEEE

Stewart, D. And Khosla, P. 1992 Real-Time Scheduling of Sensor-Based Control Systems. (in Real-Time Programming, ed. By W. Halang and K. Ramamritham) (Tarrytown, New York: Pergamon Press Inc.)

Locke, J. 1997 "Designing real-time systems" In IEEE International Conference of Real-Time Computing Systems and Applications (RTCSA '97), Taiwan (Invited talk).

Lehoczky, J. L. Sha and Ding, 1989. "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior", in Proceedings 10th IEEE Real-Time Systems Symposium, Santa Monica, CA, pp.166–171.

Stewart, D. and Khosla, P. 1992, "Real-Time Scheduling of Sensor-Based Control Systems", in Real-Time Programming, ed. By W. Halang and K. Ramamritham, Tarrytown, New York: Pergamon Press Inc.

Leung J.Y.T. and Merril, M.L., 1980. "A note on preemptive scheduling of periodic, Real Time Tasks", Information processing Letters, Vol. 11, num. 3.

Liu, C. L. and Layland, J. W. 1973. "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment", 20(1):46-61, Journal of the Association for Computing Machinery.

Baruah, S. K., Mok, A. K. and Rosier, A. K., 1990. "Preemptively Scheduling Hard-Real-Time Sporadic Tasks on One Processor", Proceedings of the 11th Real-Time Systems Symposium, pp. 182–190.

Goossens, J. and Macq, C. 2001. "Limitation of the Hyper-Period in Real-Time Periodic Task Set Generation" 9th Conference RTS embedded systems 2001, Paris (France). Page 133–148.

Buttazzo, G. January 2005. "Rate Monotonic vs. EDF: Judgment Day." Real-Time Systems, Vol. 29, Issue 1, pp. 5–26.

George, L. Rivierre, N. and Spuri, M. September 1996. "Preemptive and Non-Preemptive Real Time Uni-Processor Scheduling." INRIA Research Report number 2966.

Anonymous, http://beru.univ-brest.fr/~singhoff/cheddar/ , The Cheddar Project, Real Time Scheduling Alayzer, Erişim Tarihi: 02/01/2007

EKLER

EK 1 EDF-LLF RC İşlem Süreci Organizatörleri

EK 2 EDF-LLF SBU İşlem Süreci Organizatörleri

EK 1 EDF-LLF RC İşlem Süreci Organizatörleri

*/*Kapasite matrisinde yer alan 1 dizinin kullanımına yönelik kod parçacığı*/*

Start_Section:

```
 i : integer;
 dynamic_priority : array (tasks_range) of integer;
 T1_ptr : integer;
 T2_ptr : integer;
 T3_ptr : integer;
 T4_ptr : integer;

 T1_randomcapacity_array: array (tasks_range) of integer;
 T2_randomcapacity_array: array (tasks_range) of integer;
 T3_randomcapacity_array: array (tasks_range) of integer;
 T4_randomcapacity_array: array (tasks_range) of integer;

 T1_randomcapacity_array(0):=1;   T2_randomcapacity_array(0):=2;
 T1_randomcapacity_array(1):=1;   T2_randomcapacity_array(1):=1;
 T1_randomcapacity_array(2):=1;   T2_randomcapacity_array(2):=1;
 T1_randomcapacity_array(3):=1;   T2_randomcapacity_array(3):=2;

 T3_randomcapacity_array(0):=3;   T4_randomcapacity_array(0):=2;
 T3_randomcapacity_array(1):=5;   T4_randomcapacity_array(1):=6;
 T3_randomcapacity_array(2):=4;   T4_randomcapacity_array(2):=4;
 T3_randomcapacity_array(3):=4;   T4_randomcapacity_array(3):=4;
```

Priority_Section:

```
 for i in tasks_range loop
  if (tasks.capacity(i)>1) then
   if (tasks.rest_of_capacity(i) = 1)
```

```
   then
     if (i=1) then
         tasks.capacity(i):=T2_randomcapacity_array(T2_ptr);
         T2_ptr:=(T2_ptr+1) mod 4;
     end if;
     if (i=2) then
         tasks.capacity(i):=T3_randomcapacity_array(T3_ptr);
        T3_ptr:=(T3_ptr+1) mod 4;
     end if;
     if (i=3) then
         tasks.capacity(i):=T4_randomcapacity_array(T4_ptr);
         T4_ptr:=(T4_ptr+1) mod 4;
     end if;
   end if;
  end if;
end loop;

put(simulation_time);
put(dynamic_priority,0,3);
put(tasks.capacity,0,3);
```

/*Önceliklendirme değerinin ve değerlendirmenin yapıldığı bu bölüm EDF ve LLF algoritmaları için farklılık arz etmektedir. Aşağıda öncelikle EDF sonrasında ise LLF algoritmasının kod parçacığı yer almaktadır.*/

For EDF Preemption:

```
dynamic_priority := tasks.start_time
+ ((tasks.activation_number-1)*tasks.period)
+ tasks.deadline;
```

Election_Section:

```
     return min_to_index(dynamic_priority);
```

For LLF Preemption:

```
dynamic_deadline := tasks.start_time
+ ((tasks.activation_number-1)*tasks.period)
+ tasks.deadline;
 laxity:=dynamic_deadline-tasks.rest_of_capacity;
```

Election_Section:

```
    return min_to_index(laxity);
```

EK 2 EDF-LLF SBU İşlem Süreci Organizatörleri

Start_Section:

```
i : integer;
w : integer;
wrk :integer;
s : integer;
P : integer;
P := 128;        /*Farklı simülasyon aralıkları için parametre güncellenmektedir */
b : integer;

r0 : integer;
r1 : integer;                    Uniform random değerler atama
r2 : integer;
r3 : integer;

gen0 : random;
uniform(gen0, 1, r0);
gen1 : random;
uniform(gen1, 2, r1);            Uniform random değer aralıkları
gen2 : random;
uniform(gen2, 2, r2);
gen3 : random;
uniform(gen3, 2, r3);

dynamic_priority : array (tasks_range) of integer;
```

Priority_Section:

```
for i in tasks_range loop
   if (tasks.capacity(i)>1) then
      if (tasks.rest_of_capacity(i) = 1) then
       if (i=0) then
        w := ((P/tasks.period(i+1))*tasks.capacity(i+1))-
       (tasks.capacity(i+1)*(tasks.activation_number(i+1)-1))
       + ((P/tasks.period(i+2))*tasks.capacity(i+2))-
       (tasks.capacity(i+2)*(tasks.activation_number(i+2)-1))
       + ((P/tasks.period(i+3))*tasks.capacity(i+3))-
       (tasks.capacity(i+3)*(tasks.activation_number(i+3)-1));
       b := ((P-simulation_time)-w);

       wrk := ((P/tasks.period(i))*tasks.capacity(i))-
       (tasks.capacity(i)*(tasks.activation_number(i)));

        if (wrk>0) then
          if ( b>=wrk) then
            if((b-wrk)>wrk/2) then
                tasks.capacity(i) := tasks.capacity(i) +1;   /*Capacity Incrementation*/
           else
                tasks.capacity(i):= tasks.capacity(i);
           end if;
         else
           put(i);

       while(tasks.capacity(i+1) > 2) loop                  /*Capacity Backoff */
         tasks.capacity(i+1) := tasks.capacity(i+1)-1;
       end loop;

       while(tasks.capacity(i+2) > 2) loop                  /*Capacity Backoff */
         tasks.capacity(i+2) := tasks.capacity(i+2)-1;
       end loop;
```

```
        while(tasks.capacity(i+3) > 4) loop                /*Capacity Backoff */
          tasks.capacity(i+3) := tasks.capacity(i+3)-1;
        end loop;

  end if;
  else
  tasks.capacity(i) := tasks.capacity(i);
end if;

put(simulation_time);
put(w);
put(b);
put(wrk);
put(tasks.activation_number,0,3);
put(tasks.capacity,0,3);
put(tasks.deadline,0,3);

end if;

   if (i=1) then
   if (i=2) then          /*Görev 2, Görev 3, ve Görev 4 için aynı kod parçacığı,*/
   if (i=3) then          /*bu kısımda yer almaktadır. */

end loop
```

/*Önceliklendirme değerinin ve değerlendirmenin yapıldığı bu bölüm EDF ve LLF algoritmaları için farklılık arz etmektedir. Aşağıda öncelikle EDF sonrasında ise LLF algoritmasının kod parçacığı yer almaktadır.*/

For EDF Preemption:

```
dynamic_priority := tasks.start_time
```

+ ((tasks.activation_number-1)*tasks.period)
+ tasks.deadline;

Election_Section:

return min_to_index(dynamic_priority);

For LLF Preemption:

dynamic_deadline := tasks.start_time
+ ((tasks.activation_number-1)*tasks.period)
+ tasks.deadline;
laxity:=dynamic_deadline-tasks.rest_of_capacity;

Election_Section:

return min_to_index(laxity);

ÖZGEÇMİŞ

Adı Soyadı : Orhan Fikret DUMAN
Doğum Yeri : Ankara
Doğum Tarihi : 30.08.1981
Medeni Hali : Bekar
Yabancı Dili : İngilizce, Almanca

Eğitim Durumu (Kurum ve Yıl)

Lise : Maltepe Askeri Lisesi , 1995-1999
Lisans : İstanbul Üniversitesi Bilgisayar Mühendisliği , 1999-2003
Yüksek Lisans : Ankara Üniversitesi Bilgisayar Mühendisliği , 2004-2008

Çalıştığı Kurum/Kurumlar ve Yıl

Jandarma Muhabere Ana Dp. Ve Fb. K.lığı : 2003-2004
Jandarma Genel Komutanlığı : 2004-2008

Yayınları (SCI ve diğer)

R. SAMET, O.F.DUMAN, "Methods for Improving Fault-Tolerant Scheduling Techniques for Real-Time Systems" MS2006 Proceedings of the International Conference on Modeling and Simulation, 28-30 August 2006, Konya, Turkey, pp. 643-647, 2006.

R. SAMET, O.F.DUMAN, "Behaviours of Real-Time Schedulers Under Capacity Modification And a Steady Scheme With Bounded Utilization", Journal of Scheduling, Manuscript Number: JOSH273, 2007.

R. SAMET, O.F.DUMAN, "Analysis of The Costs for Preemptive Dynamic Real Time Schedulers and Comparison with Steady Scheme With Bounded Utilization", Draft Manuscript, 2008.