Araştırma Makalesi

Çok İşlemcili Gerçek Zamanlı Sistemlerde Zamanlama Algoritmaları

Muhammet Kürşat Yıldırım*[1], Lütfi Süder[1], Ahmed Abd Alkader[2], Abdullah Elewi[1]

[1] *Mersin Üniversitesi, Bilgisayar Mühendisliği Bölümü, Mersin, Türkiye*
[2] *Hatay Mustafa Kemal Üniversitesi, Hassa Meslek Yüksekokulu, Hatay, Türkiye*

Anahtar Kelimeler:
Gerçek zamanlı sistemler
Çok işlemci
Zamanlama
MCRTsim

ÖZ

Gerçek zamanlı sistemler hayatımızın birçok alanında çok önemli bir rol oynar. Bu sistemlerin düzenli çalışabilmesi ve performans bakımından daha verimli olabilmesi için kullanılan bazı algoritmalar vardır. Bu çalışmada çok işlemcili gerçek zamanlı sistemler üzerinde çalışan zamanlama algoritmalarından olan, Evrensel Erken Biten Önce (Global Earliest Deadline First-GEDF), Bölmeli Erken Biten Önce (Partitioned Earliest Deadline First-PEDF), Bölmeli Monoton Oran (Partitioned Rate Monotonic-PRM) ve Evrensel Monoton Oran (Global Rate Monotonic-GRM) algoritmaları ele alınmıştır. Bu algoritmalar işlemciye gelen görevlerin hangi sıra ile işleme konulacağını belirlemektedir. Bu algoritmalarla ilgili bazı örnekler, açık kaynaklı MCRTsim uygulamasında çalıştırılmıştır ve alınan karşılaştırmalı sonuçları verilmiştir.

Scheduling Algorithms in Multiprocessor Real-Time Systems

Keywords:
Real-time systems
Multiprocessor
Scheduling
MCRTsim

ABSTRACT

Real-time systems play a crucial role in many aspects of our lives. There are some algorithms used for these systems to work regularly and to be more efficient in terms of performance. In this study, the schduling algorithms for multiprocessor real-time systems are discussed. These algorithms include: Global Earliest Deadline First (GEDF), Partitioned Earliest Deadline First (PEDF), Partitioned Rate Monotonic (PRM) and Global Rate Monotonic (GRM) algorithms. These algorithms determine the order in which the tasks coming to the processor will be processed. Comparative results of these algorithms using open-source MCRTsim application have been given.

***Sorumlu Yazar**

*(m.kursatyildirim33@gmail.com) ORCID ID 0000-0001-7346-8868
(lutfisuder1992@gmail.com) ORCID ID 0000-0002-6949-0824
(akader@mku.edu.tr) ORCID ID 0000-0002-0538-7924
(elewi@mersin.edu.tr) ORCID ID 0000-0001-9774-5292

e-ISSN:2717-8579
Geliş Tarihi: 31/08/2020; Kabul Tarihi: 07/10/2020

1. GİRİŞ

Gerçek zamanlı sistemler kullanımı oldukça yaygın olan ve her geçen gün daha da yaygınlaşan sistemlerdir. Verilen görevlerin tam zamanında yerine getirilmesi için gerçek zamanlı sistemlere ihtiyaç vardır. Çünkü bir işin tam zamanında yapılması gerekiyorsa bu iş, zaman açısından kritik bir iştir. Başka bir deyişle, gerçek zamanlı sistemler, sistemin gerçek zamana tabi olduğu anlamına gelir, yani yanıt, belirli bir zamanlama kısıtlaması dahilinde garanti edilmeli veya sistem belirtilen zaman sınırını karşılamalıdır. Gündelik yaşantımızda güvenlik (radar sistemleri), ulaşım (otomobillerdeki fren sistemleri), haberleşme (cep telefonları) gibi birçok alanda gerçek zamanlı sistemler kullanılmaktadır.

Günümüzde işlemcilerin artan maliyetlerinden dolayı bir gömülü sistemde birden fazla görev ve iş eş zamanlı olarak çalıştırılmaya başlanmıştır. Bu görevlerin doğru, eksiksiz ve güvenilir bir şekilde çalıştırılması ve kaynakların doğru kullanılması çok önemlidir.

Çok işlemcili gömülü sistemler en az iki veya daha fazla işlemciden oluşur. Çok işlemcili zamanlamada, çeşitli işlemlerin aynı anda çalışması için yükü paylaşan birden fazla işlemci (CPU) vardır.

Çok işlemcili bir sistemde ayarlanan gerçek zamanlı bir görevin son tarihlerini karşılamak için bir zamanlama algoritması gerekir. Bu zamanlama algoritması, sistemdeki her görevin dağıtımın uygun bir biçimde yapılmasından (dağıtma problemi) ve ne zaman ve hangi sırayla bu görevlerin çalıştırılmasının (zamanlama problemi) belirlenmesinden sorumludur. Yaşanılan genel problemler şunlardır:

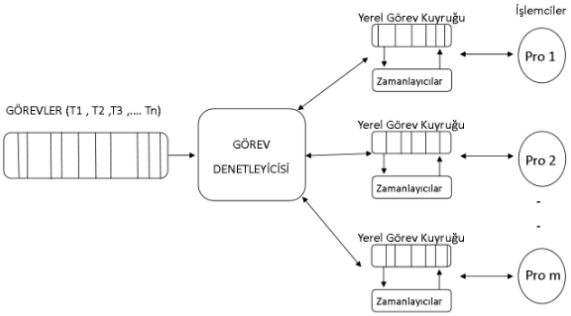
- Tek bir işlemci için bazı araştırma sonuçları her zaman birden fazla işlemci için uygulanamaz (Dhall ve Liu, 1978; Garey ve Johnson, 1979).
- Birden fazla işlemcide farklı zamanlama anormallikleri ortaya çıkabilir (Baker, 2003; Andersson, 2003; Ha ve Liu 1994).
- Dağıtma problemine çözüm olarak yüksek hesaplama karmaşıklığına sahip algoritmalar kullanmak gerekir.

Yukarıdaki problemlerin giderilebilmesi için literatürde bulunan çeşitli algoritmalar kullanılabilir. Bu çalışmada en sık kullanılan algoritmalardan olan GEDF ve GRM algoritmalarından bahsedilmiş olup, MCRTsim uygulaması üzerinde çalıştırılan simülasyon sonuçları ve karşılaştırmaları verilmiştir. MCRTsim uygulaması, Java programlama dili ile geliştirilmiş ve açık kaynaklı bir uygulamadır. Tek işlemcili, çok işlemcili ve çok çekirdekli işlemcilere sahip gerçek zamanlı sistemler için açık kaynaklı bir görev zamanlama simülatörü sunar (Wu ve Huang, 2017). Yaptığımız çalışmada MCRTsim uygulamasının yeniden geliştirilip derlenmiş olan versiyonu kullanılmıştır (URL-1).

Bu çalışmada, ikinci bölümde zamanlama algoritmalarının (GRM, GEDF, PEDF, PRM) yapısı ve çalışma mantıkları anlatılmıştır. Gerekli matematiksel denklemler verilmiştir. Üçüncü bölümde ise bu algoritmalarla ilgili örnekler MCRTsim uygulaması üzerinde simule edilmiştir. Sonuç bölümünde bu simülasyon sonuçları değerlendirilmiş ve çıkarımlarda bulunulmuştur.

2. ZAMANLAMA ALGORİTMALARI

Zamanlama algoritmaları sistemlerin kritik seviyedeki görevleri en doğru şekilde işlemelerini sağlamak için oluşturulan algoritmalardır. Bu algoritmalar evrensel (global) ve bölmeli (partitioned) olarak iki şekilde işlenmektedir (Davis ve Burns, 2011). Bölmeli algoritmalarda verilen bir görevi hangi işlemci üstlenmişse o görevi sadece o işlemci icra etmektedir. Bölmeli zamanlama algoritmalarının genel yapısı Şekil 1.(a)'da gösterilmiştir. Evrensel algoritmada ise bir görev farklı farklı işlemciler tarafından icra edilebilmektedir (Zapata ve Alvarez, 2005; Guan, 2016). Evrensel zamanlama algoritmalarının genel yapısı Şekil 1.(b)'de gösterilmiştir. Burada dikkat edilmesi gereken nokta ise bir görevin farklı parçaları aynı anda farklı farklı işlemcilerde işlenemez.



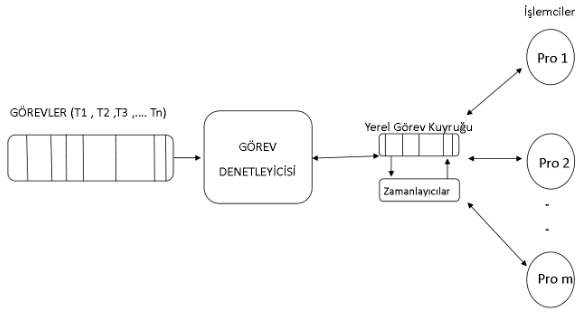
Şekil 1.(a) Bölmeli zamanlama algoritmalarının genel yapısı

Global zamanlamada, bir görev hazır kuyruğuna konur ve tüm işlemciler tarafından paylaşılan ve her an m yüksek öncelikli görev, yürütme için seçilen m işlemci tarafından işleme konur.

Global zamanlama algoritmalarının tasarım düzeni şu şekildedir:

- Görevlerin(tasks) işlemcilere aktarılması sisteme yüksek yük getirir.
- Bir işlemciden diğerine aktarılmasına izin verildiğinden, işlemciler yüksek olma eğilimindedir. Bu iletişim kanalları paylaşılan iletişim hafızasının kullanılmasını sağlar.
- Bu tasarım düzeninde Dhall Etkisi oluşabilir ve görevler küçük kullanımlı şekilde planlanamayabilir (Dhall ve Liu, 1978).
- Bu tasarım düzeninde bazı programlama anomalileri oluşabilir (Lauzac ve ark., 1998;

Andersson ve Jonsson, 2000; Andersson, 2003).



Şekil 1.(b) Evrensel zamanlama algoritmalarının genel yapısı

Bölmeli algoritmaları gelen görevlerin işlemciye hangi öncelik sırasına göre yerleştirileceğini belirlemede kullanılan algoritmalardır. Bu algoritmaların doğru çalışması için görevlerin işlemciye aynı anda gelmesi gerekmektedir. Literatürde birçok bölme algoritması bulunmaktadır. Bu yazıda daha çok ilk bulunan yere yerleştirme (First Fit-FF), en iyi yerleştirme (Best Fit–BF), en kötü yerleştirme (Worst Fit–WF) algoritmaları (Zapata ve Alvarez, 2005; Elewi ve ark., 2014) ve bunların türevleri incelenecek ve MCRTsim uygulaması üzerinden analiz edilecektir. Bu algoritmaların en büyük göreve öncelik vererek yerleştirmesine de FFD (FF Decreasing), BFD (BF Decreasing) ve WFD (WF Decreasing) denmektedir.

İlk Bulunana Azalan Yerleştirme (FFD): FFD algoritmasında, öğeler ilk önce azalan boyutta sıralanır ve sonra C(Bj) + s(ai) ≤ l denklemine göre sığacağı ilk yere (en düşük indeks) yerleştirilir. Aksi takdirde, ilk elemanı ai olan yeni bir boş alan başlatılır (Guochuan ve Enyu, 1998).

En İyi Azalan Yerleştirme (BFD): BFD algoritmasında, öğeler ilk önce azalan boyutta sıralanır ve çalıştırılır.

En Kötü Azalan Yerleştirme (WFD): WFD algoritmasında, öğeler ilk önce azalan sırada sıralanır ve daha sonra boşluğu en büyük olan işlemciye yerleştirilir ve çalıştırılır.

Bölme algoritmaları zamanlama algoritmaları ile uygun şekilde çalışabilmektedirler. Özellikle Erken Biten Önce (Earliest Deadline First-EDF) zamanlama algoritması şu şekilde grup oluşturabilmektedir (Beitollahi ve Deconinck, 2006):

$$A = \{NF, BF, FF, \dots\} x(EDF\ Algoritması)$$

2.1. Evrensel Monoton Oran (Global Rate Monotonic-GRM)

Evrensel Monoton Oran (GRM) zamanlama algoritması en sık kullanılan algoritmalardan bir tanesidir. Bu algoritmada ilk işlenecek görev en küçük periyotlu görevdir. İstisnai bir durum olarak, görevin işlemciye varış zamanı öncelik durumunu değiştirmektedir. GRM algoritmasında işlemci bir görevi icra ederken daha yüksek öncelikli bir görev gelirse o göreve geçiş yapan (preemptive) bir algoritmadır. Gerekli durumlarda bir görevden diğer göreve geçişi olmayacak şekilde (nonpreemptive) de ayarlanabilir. RM algoritmasının hatasız çalışması için 1973 yılında Liu ve Layland tarafından ortaya çıkarılan denklem 1 (U işlemci kullanım yüzdesi, n görev sayısı, C hesaplanma süresi ve P periyot olmak üzere) deki şartı sağlaması gerekmektedir. Bu şart tek işlemcili durumlarda geçerlidir (Liu ve Layland, 1973).

$$U = \sum_{i=1}^{n} \left(\frac{C_i}{P_i}\right) \leq n(\sqrt[n]{2} - 1) \qquad (1)$$

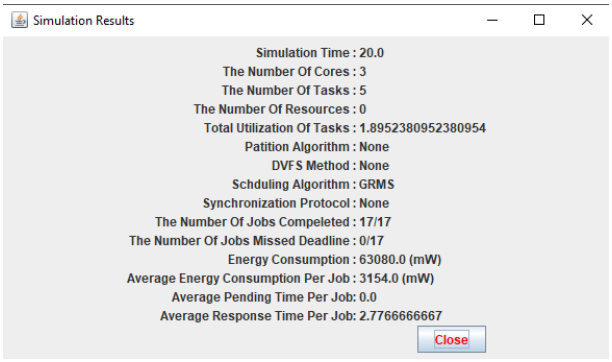
Aşağıda bulunan denklem 2'ye göre, son teslim tarihleri olan bir dizi periyodik görev, m işlemciyle, $m \geq 2$ şartını sağlayacak şekilde preemptive global monoton oran (GRM) algoritması kullanılarak programlanırsa görevlerin başarıyla tamamlanması garantidir (Baker, 2003).

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq \frac{m}{2}(1 - \alpha) + \alpha \qquad (2)$$

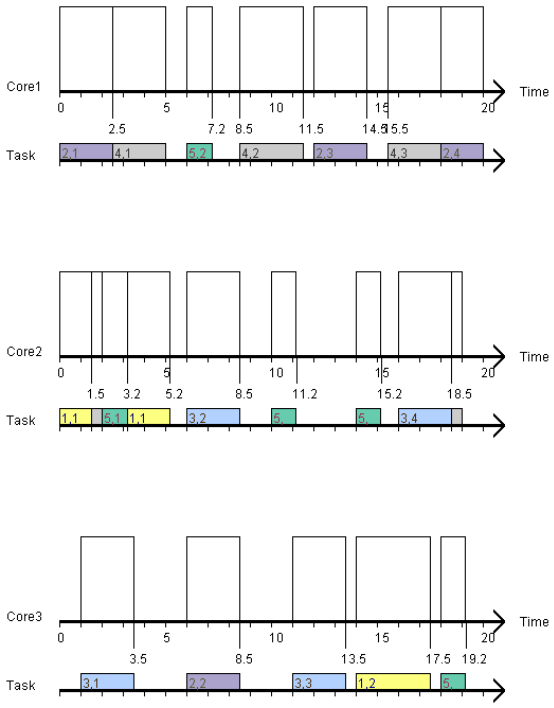
$$\alpha = \max\left\{\frac{C_i}{P_i} \mid i = 1, \dots, n\right\}$$

Bu kısımda GRM algoritmalarının nasıl çalıştığı MCRTsim uygulaması kullanılarak gösterilmiştir. Burada görevler $T_n(r, C, P)$ şeklinde gösterilmiştir. Bu gösterimde r görevin işlemciye varış zamanını, C görev süresini ve P görevin periyodunu göstermektedir. Örnek olarak $T_1(0, 3.5, 14)$, $T_2(0, 2.5, 6)$, $T_3(1, 2.5, 5)$, $T_4(1.5, 3, 7)$, $T_5(2, 1.2, 4)$ görevleri çalıştırılmıştır. Denklem 2'deki formülden yararlanılarak işlemci sayısı 3 olarak belirlenmiştir.

Aşağıda Şekil-2.(a)'da ki sonuca (Total Utilization of Tasks) baktığımız da toplam kullanım 1.895 olarak görülmüştür. Böylece 2,25'den küçük olduğu için denklem 2'yi sağlamış ve simülasyon başarıyla çalışmıştır.



Şekil 2. (a). GRM Algoritması Simülasyon Sonucu



Şekil 2. (b). GRM Algoritması İşlemci Görev Grafiği

Ayrıca Şekil-2.(b)'deki işlemci görev grafiğine baktığımızda bütün işlemcilere görev dağılımının yapıldığı görülmüştür. Bu noktada dikkat edilmesi gereken diğer bir husus ise; bir görev başladığı bir işlemci tarafından sonlandırılmadan diğer işlemciye aktarılabilmektedir. Örneğin Şekil-2.(b)'deki işlemci zaman grafiğine baktığımızda; 4 numaralı görevin ilk periyodu 2 numaralı işlemci ile başlayıp 1 numaralı işlemci ile devam etmiştir.

2.2. Evrensel Erken Biten Önce (Global Earliest Deadline First-GEDF)

Bir dizi periyodik görevin zamanlanması için kullanılan en yaygın dinamik öncelikli zamanlama yöntemi EDF algoritmasıdır (Palamut ve ark., 2019). Bu algoritmada öncelikle periyot süresi kısa olan işlem yapılmaktadır. Periyot sürelerine göre sıraya koyulmaktadır. Bu algoritmada en önemli noktalardan biri ise öncelik sıralaması yapılıp yapılmayacağına karar verilebilmesidir. Yani önceliği yüksek olan bir görev diğer görevin geçici olarak durdurulmasına ve kendisinin icra edilmesine (preemptive) sebep olabilir ya da tam tersi olarak da çalışabilir. Yani en kısa görev süresi olan görev bitene kadar yeni görev hiçbir şekilde icra edilmez(non preemptive).

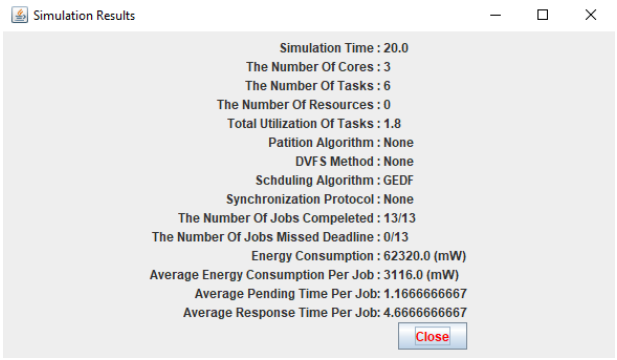
1973 yılında Liu ve Layland tarafından yapılan çalışmaya göre eğer görev zamanlamaları denklem 3'de ki eşitliği (U işlemci kullanım yüzdesi, n görev sayısı, C hesaplanma süresi ve P periyot olmak üzere) sağlıyorsa EDF tek işlemcili durumlarda sorunsuz çalıştırılabilir (Liu ve Layland, 1973).

$$U = \sum_{i=1}^{n} \left(\frac{C_i}{P_i}\right) \leq 1 \qquad (3)$$

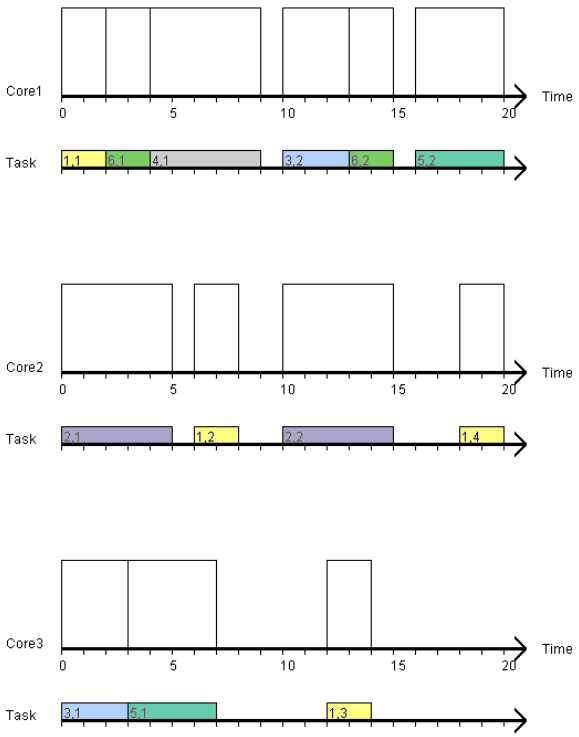
Yukarıda belirtilen denklem 3'de ki kurala göre, bir dizi periyodik görev, T_1, T_2, ...T_n tüm periyotların eşit olması kaydıyla, m işlemci üzerinde preemptive GEDF algoritması kullanılırsa görevlerin başarıyla tamamlanması garantidir (Goossens ve ark., 2003).

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq m(1-\alpha)+\alpha \qquad (4)$$
$$\alpha = \max\left\{\frac{C_i}{P_i} \mid i = 1, \dots, n\right\}.$$

Bu çalışmada PEDF algoritması ile çeşitli parçalama algoritmalarının kombinasyonları ve GEDF algoritması MCRTsim programında çalıştırılmıştır. Örnek olarak; $T_1(0, 2, 6)$, $T_2(0, 5, 10)$, $T_3(0, 3, 10)$, $T_4(0, 5, 20)$, $T_5(0, 4, 16)$, $T_6(0, 2, 12)$ görevleri çalıştırılmıştır. Simulasyon sonuçları Şekil-3 ve MCRTsim uygulaması simülasyon sonuçları başlığı altında yorumlanmıştır.



Şekil 3. (a). GEDF Algoritması Simülasyon Sonucu

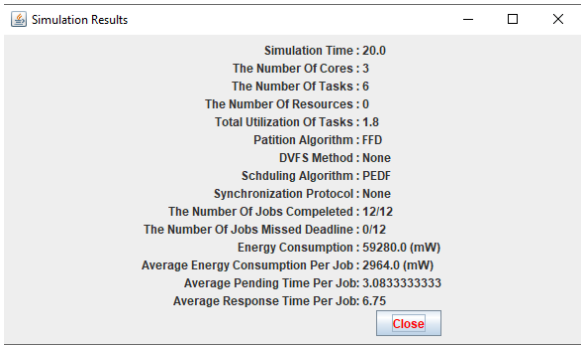


Şekil 3. (b). GEDF Algoritması İşlemci Görev Grafiği

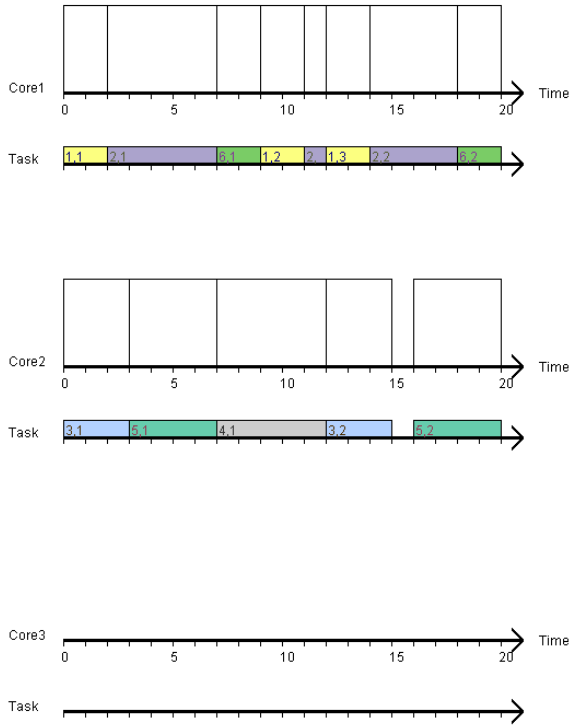
2.3. Bölmeli Erken Biten Önce (Partitioned Earliest Deadline First-PEDF)

PEDF-WFD algoritmaları bütün işlemcileri kullanma gereği duyarken PEDF-FFD ve PEDF-BFD algoritmaları sadece 2 işlemciyi kullanarak görevleri başarıyla icra edebilmiştir.

Aşağıda Şekil-4.a'da PEDF-FFD algoritmasının başarılı şekilde çalıştığı görülmüştür. Diğer algoritmalar ile karşılaştırma sonuçları Tablo 1.'de verilmiştir.

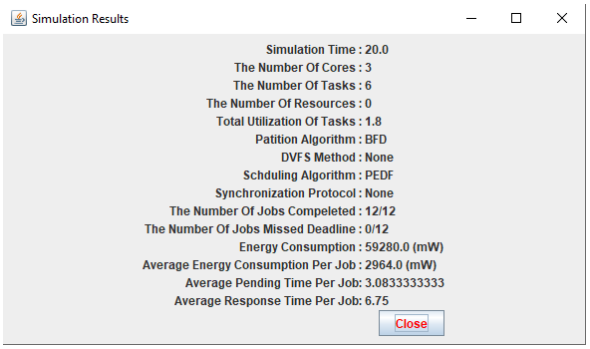


Şekil 4. (a). PEDF-FFD Algoritması Simülasyon Sonucu

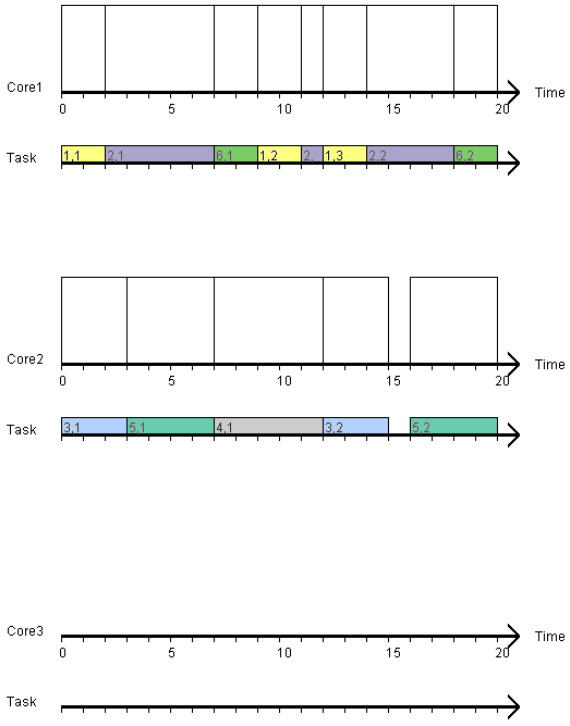


Şekil 4. (b). PEDF-FFD Algoritması İşlemci Görev Grafiği

Aşağıda Şekil-5.a'da PEDF-BFD algoritmasının başarılı şekilde çalıştığı görülmüştür. Diğer algoritmalar ile karşılaştırma sonuçları Tablo 1.'de verilmiştir.

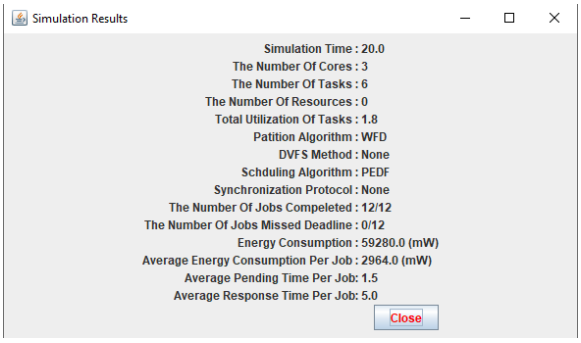


Şekil 5. (a). PEDF-BFD Algoritması Simülasyon Sonucu

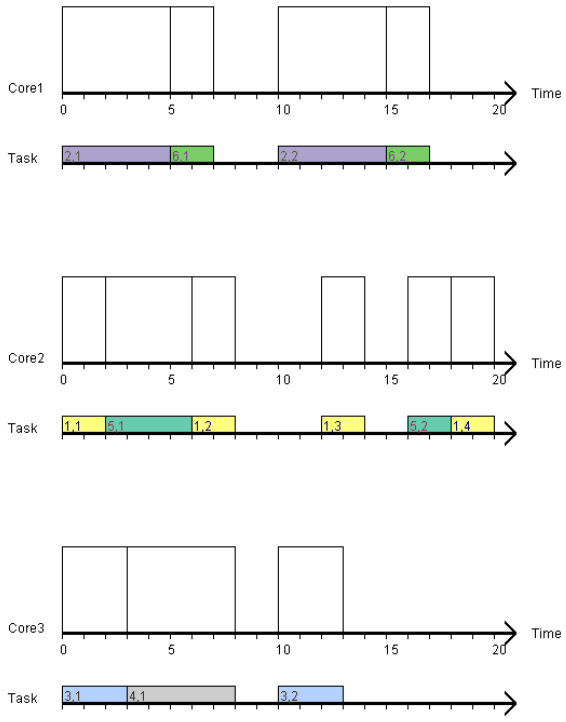


Şekil 5. (b). PEDF-BFD Algoritması İşlemci Görev Grafiği

Aşağıda Şekil-6.a'da PEDF-WFD algoritmasının başarılı şekilde çalıştığı görülmüştür. Diğer algoritmalar ile karşılaştırma sonuçları Tablo 1.'de verilmiştir.



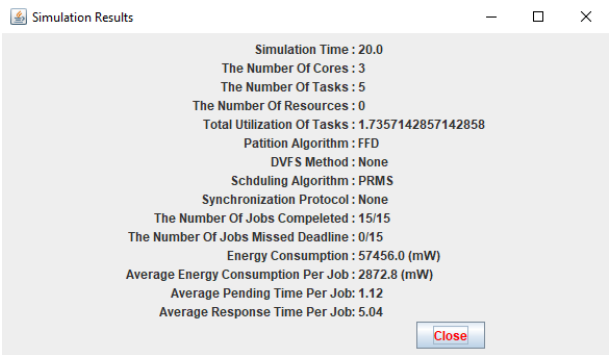
Şekil 6. (a). PEDF-WFD Algoritması Simülasyon Sonucu



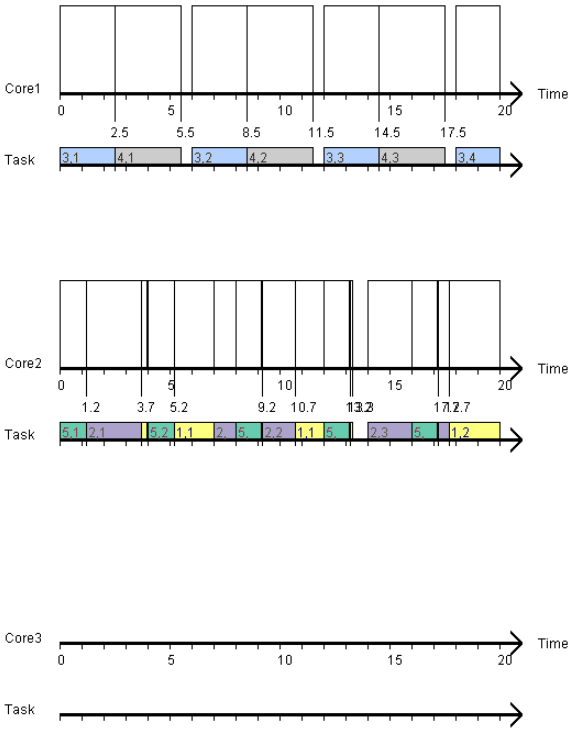
Şekil 6. (b). PEDF-WFD Algoritması İşlemci Görev Grafiği

2.4. Bölmeli Monoton Oran (Partitioned Rate Monotic -PRM)

PRM algoritması FFD, BFD, WFD algoritmaları ile çalıştırılmıştır, ayrica Şekil-7, Şekil-8 ve Şekil-9'da algoritmasının çalıştığı görülmüştür. Örnek görev olarak da T_1 (0, 3.5, 15), T_2(0, 2.5, 7), T_3(0, 2.5, 6), T_4(0, 3, 7), T_5(0, 1.2, 4) görevleri kullanılmıştır. Simülasyon sonuçlarına baktığımızda WFD algoritmasının daha kısa sürede cevap verebildiği görülmüştür. İşlemci görev grafiğine baktığımızda WFD algoritmasının 3 işlemciyi de kullandığı görülmüştür. BFD ve FFD algoritmasının ise sadece 2 işlemciyi kullandığı görülmüştür. Bitene kadar yeni görev hiçbir şekilde icra edilmez (non preemptive).



Şekil 7. (a). PRM-FFD Algoritması Simülasyon Sonucu PRM-FFD algoritması başarılı çalışmıştır.

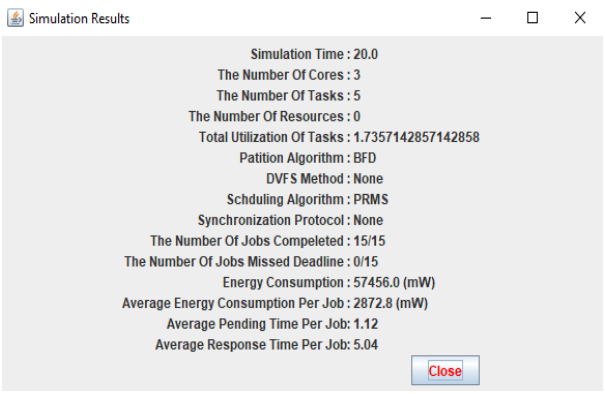


Şekil 7. (b). PRM-FFD Algoritması İşlemci Görev Grafiği

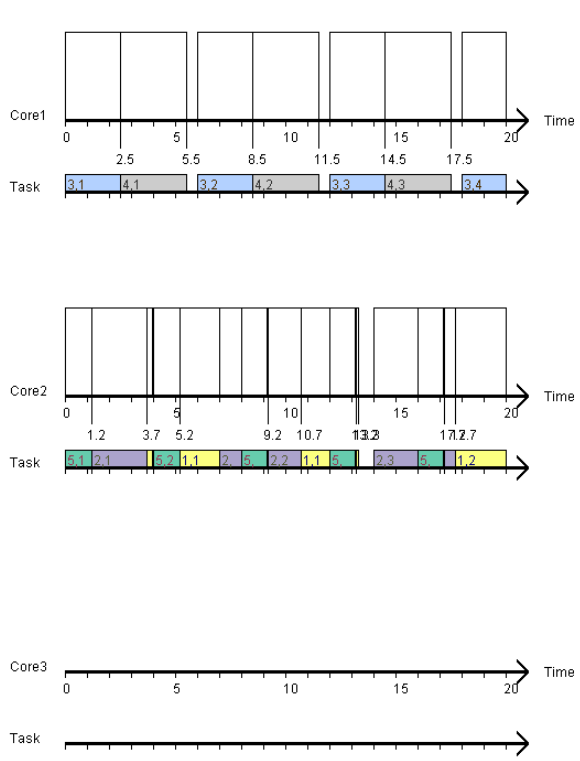
Yukarıda Şekil-7.a'da PRM-FFD algoritmasının başarılı şekilde çalıştığı görülmüştür. Ayrıca Şekil-7.b'de bu görevleri başarabilmesi için sadece iki işlemcinin yeterli olduğu görülmüştür. Diğer algoritmalar ile karşılaştırma sonuçları Tablo 2.'de verilmiştir.

PRM-BFD algoritması PRM-FFD algoritması ile aynı sonucu vermiştir. Aşağıda Şekil-8.a'da PRM-BFD algoritmasının başarılı şekilde çalıştığı görülmüştür. Diğer algoritmalar ile karşılaştırma sonuçları Tablo 2.'de verilmiştir

Aşağıda Şekil-9.a'da PRM-WFD algoritmasının başarılı şekilde çalıştığı görülmüştür. Şekil-9.b'ye baktığımızda ise bu algoritmanın aynı işi yapabilmesi için üç işlemci kullandığı görülmüştür. Diğer algoritmalar ile karşılaştırma sonuçları Tablo 2.'de verilmiştir.

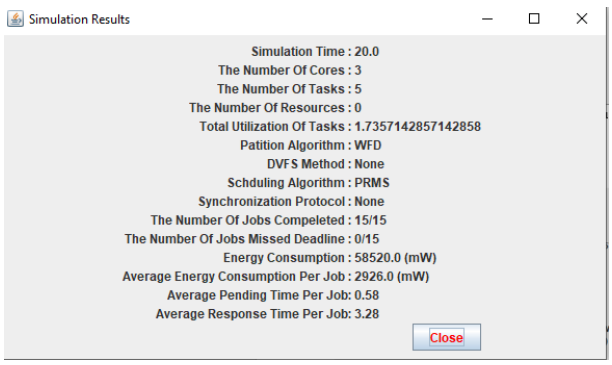


Şekil 8.(a). PRM-BFD Algoritması Simülasyon Sonucu

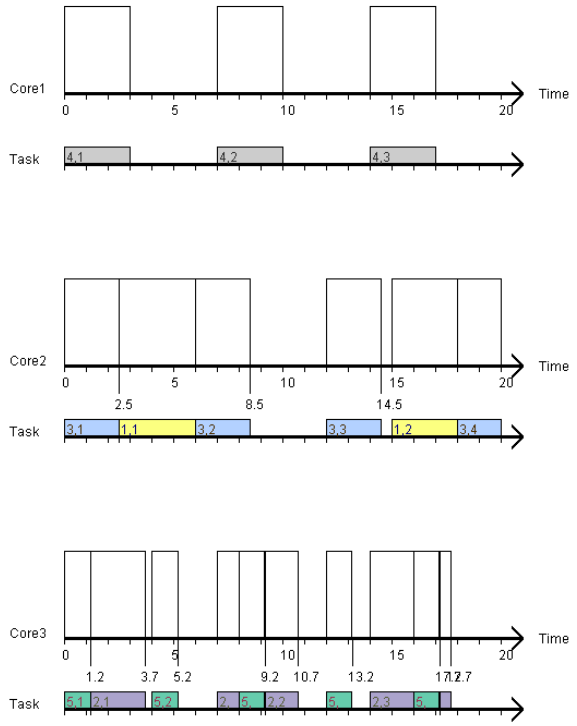


Şekil 8. (b). PRM-BFD Algoritması İşlemci Görev Grafiği

MCRTsim uygulamasında draw butonuna tıklayarak simülasyon sonucu grafiksel olarak gisterilebilmektedir. Çizim yaptırıldığında AllCores butonu tıklanarak aşağıda belirtilen işlemci görev grafikleri elde edilmektedir. Bu grafiklerde hangi işlemcinin hangi işi yüklendiği ve hangi sürelerde çalıştırdığı gözlemlenebilmektedir. Grafikte belirtilen task satırında birinci rakam kaç numaralı görev olduğunu, ikinci rakam ise o görevi kaçıncı defa çalıştırdığını göstermektedir. Örneğin 5,1 gösteriminde 5 görev numarası, 1 ise kaçıncı defa çalıştığını göstermektedir.



Şekil 9. (a). PRM-WFD Algoritması Simülasyon Sonucu



Şekil 9. (b). PRM-WFD Algoritması İşlemci Görev Grafiği

3. MCRTsim UYGULAMASI SİMÜLASYON SONUÇLARI

GRM algoritması, GEDF ve PEDF algoritmaları için yukarıdaki şekillerde verilen örnekler yapılmıştır. Bu örneklerle ilgi çıkarımlar bu kısımda anlatılmıştır.

3.1. Evrensel Monoton Oran (GRM) Algoritması Simülasyon Sonuçları

GRM algoritması için verilen denklem 2'deki kural gözetilerek 3 işlemcili simülasyon yapılmıştır. Simülasyon sonuçlarına göre denklem 2'de sağlanması gereken m≥2 koşulu başarıyla yerine getirilmiştir.

GRM algoritması için Şekil-2.(a)'daki Simulasyon sonucuna bakıldığında GRM algoritmasının bütün görevleri başarılı bir şekilde yerine getirdiği görülmüştür. Şekil-2.(b)'deki GRM Algoritması İşlemci Görev Grafiğine baktığımızda 3 işlemciyi de aktif olarak kullandığı görülmektedir. Ayrıca dikkat edilirse GRM algoritmasının Şekil-2.(b)'deki 4 numaralı görevin 1. icrası esnasında bu görevin bir kısmını 1. işlemciye yaptırırken diğer bir kısmını 2. işlemciye yaptırmaktadır.

3.2. Evrensel Erken Biten Önce (GEDF) Algoritması Simülasyon Sonuçları

GEDF algoritmasının Şekil-3.(a)'da ki Simulasyon sonucuna baktığımızda görevi başarılı bir şekilde tamamladığı görülmüştür. Tablo 1. deki ortalama cevap süresi karşılaştırmalarına göre GEDF

algoritmasının diğer PEDF algoritması kombinasyonlarına göre daha kısa sürede cevap verdiği görülmüştür. Şekil-3.(b)'deki GEDF Algoritması İşlemci Görev Grafiğine baktığımızda üç işlemcinin de aktif kullanıldığı ve bu algoritmanın nonpreemtive olarak çalıştığı görülmüştür. Bu yüzden bir görevi alan işlemci o görevi bitirene kadar yeni görev almamış ve elindeki görevi başka işlemciye devretmemiştir. Biten görev yeni görev olarak başka işlemcilere aktarılabilmiştir.

3.3 Bölmeli Erken Biten Önce (PEDF) Algoritması Simülasyon Sonuçları

PEDF algoritmasının hangi parçalama algoritmasında daha verimli çalıştığını anlamak için yukarıda belirtilen FFD, BFD ve WFD parçalama algoritmaları kullanılmıştır. Şekil-4.(a)'da ki PEDF-FFD Algoritması Simülasyon Sonucuna baktığımızda PEDF-FFD algoritmasının tüm görevleri başarılı bir şekilde icra ettiği görülmüştür. Ayrıca Şekil-4.(b)'deki PEDF-FFD Algoritması İşlemci Görev Grafiğine baktığımızda iki işlemcinin bu görevi yapması için yeterli olduğu görülmüştür. Şekil 5.(a).'da ki PEDF-BFD Algoritması Simülasyon Sonucuna baktığımızda bu parçalama algoritmasının da başarılı bir şekilde tüm görevleri icra ettiği görülmüştür. Şekil-5.(b)'deki PEDF-BFD Algoritması İşlemci Görev Grafiğine baktığımızda ise bu algoritmanın da aynı şekilde iki işlemci ile bu görevi başardığı görülmüştür. Şekil-6.(a)'da PEDF-WFD Algoritması Simülasyon Sonucuna baktığımızda bu algoritmanın da tüm görevleri başarılı şekilde icra ettiği görülmüştür. Fakat Tablo 1.'de de gösterildiği gibi PEDF-WFD algoritması PEDF-BFD ve PEDF-FFD algoritmalarına göre daha kısa sürede işlemi tamamlamıştır. Şekil-6.(b)'de PEDF-WFD Algoritması İşlemci Görev Grafiğine baktığımızda bu algoritmanın diğer iki algoritmadan farklı olarak üç işlemci kullandığı görülmüştür.

Tablo 1. GEDF ve PEDF Algoritmaları Görev Tamamlama Başarısı Sonuçları

	GEDF	PEDF-FFD	PEDF-BFD	PEDF-WFD
Görev Sayısı	6	6	6	6
Toplam Kullanım	1.8	1.8	1.8	1.8
Yapılan İş Sayısı	13	12	12	12
Tamamlanan İş Sayısı	13	12	12	12
Hatalı İş Sayısı	0	0	0	0
Ortalama Bekleme Süresi/İş Başına	1.17	3.08	3.08	1.5
Ortalama Tepki Süresi/İş Başına	4.67	6.75	6.75	5.0

3.4. Bölmeli Monoton Oran (PRM) Algoritması Simülasyon Sonuçları

Simülasyon sonuçlarına baktığımızda WFD algoritmasının daha kısa sürede cevap verebildiği görülmüştür. Şekil-9.(b)'deki İşlemci Görev Grafiğine baktığımızda WFD algoritmasının 3 işlemciyi de kullandığı görülmüştür. Şekil 8.(b)'de PRM-BFD algoritmasının ve Şekil-7.(b)'de PRM-FFD algoritmasının ise sadece 2 işlemciyi kullandığı görülmüştür. Ayrıca Şekil-9.(b)'de bir görevin hangi işlemci ile başladıysa sürekli o işlemci tarafından çalıştırıldığı görülmüştür.

Tablo 2'de GRM ve PRM algoritmalarının aynı örnek görevler için MCRTsim'de çalıştırılmasıyla elde edilen değerleri karşılaştırmalı olarak verilmiştir.

Tablo 2. GRM ve PRM Algoritmaları Görev Tamamlama Başarısı Sonuçları

	GRM	PRM-FFD	PRM-BFD	PRM-WFD
Görev Sayısı	5	5	5	5
Toplam Kullanım	1.895	1.736	1.736	1.736
Yapılan İş Sayısı	17	15	15	15
Tamamlanan İş Sayısı	17	15	15	15
Hatalı İş Sayısı	0	0	0	0
Ortalama Bekleme Süresi/İşBaşına	0	1.12	1.12	0.58
Ortalama Tepki Süresi/İş Başına	2.78	5.04	5.04	3.28

4. SONUÇ

Yapılan bu çalışma sonucunda RM algoritması için evrensel algoritmanın cevap süresinin parçalı algoritmaya göre daha kısa olduğu görülmüştür. Ayrıca GRM algoritmasının işlemcide sürekli görev dağılımını değiştirdiği ve 3 işlemciyi de kullandığı gözlemlenmiştir. Fakat PRM-FFD ve PRM-BFD algoritmaları sadece 2 işlemciyi kullanmıştır. Bu da GRM algoritmasının daha kısa sürede cevap vermesi ve daha verimli çalışmasını sağlamıştır.

GEDF algoritması aynı süre içerisinde PEDF algoritmasına göre daha fazla görevi icra etmiştir. Ayrıca GEDF algoritmasının PEDF algoritmasına göre cevap verme süresi daha kısa olmuştur. Bu durumda GEDF algoritmasının daha verimli olduğu gözlemlenmiştir.

Parçalama algoritmalarından PEDF-WFD algoritması 3 işlemciyi kullanmıştır. Fakat PEDF-BFD ve PEDF-FFD algoritmaları 2 işlemciyi kullanmıştır. Bu da PEDF-WFD algoritmasının daha hızlı cevap vermesini sağlamıştır. Böylece PEDF-WFD algoritması daha verimli çalışmıştır.

Bir sonraki çalışmalarda ise bu algoritmaların enerji verimlilik değerlendirilmesi yapılabilir. Hangi algoritmanın aynı iş için daha fazla enerji tükettiği tespit edilebilir. Bu sonuçlara göre de hangi alanlarda hangi algoritmanın kullanılmasının daha mantıklı olabileceği tespit edilebilir.

KAYNAKÇA

Andersson, B. (2003). Static Priority Scheduling in Multiprocessors. In *PhD Thesis, Department of Comp.Eng., Chalmers University, 2003.*

Andersson, B. ve Jonsson, J. (2000). Fixed-Priority Preemptive Multiprocessor Scheduling: To Partition or not to Partition. In *IEEE Int'l Conference on Real Time Computing Systems and Applications, Dec. 2000.*

Baker, T. P. (2003). Multiprocessor EDF and Deadline Monotonic Schedulability Analysis. In *IEEE Real-Time Systems Symposium, Dec, 2003.*

Beitollahi, H. ve Deconinck, G. (2006). Fault-Tolerant Partitioning Scheduling Algorithms in Real-Time Multiprocessor Systems. In *12th Pacific Rim International Symposium on Dependable Computing (PRDC'06).*

Dhall, S. K. Ve Liu, C. L. (1978). On a Real-Time Scheduling Problem. In *Operations Research, vol. 26, number 1, 127-140, 1978.*

Garey, M. R. ve Johnson, D. S. (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness. In *W. H. Freeman, New York, 1979.*

Goossens, J., Funk, S., & Baruah, S. (2003). Priority-Driven Scheduling of Periodic Task Systems on Multiprocessor. In *Real Time Systems, vol 25, 187-205, 2003.*

Guan, N. (2016). Techniques for Building Timing-Predictable Embedded Systems. In *Springer International Publishing Switzerland, pp. 15, 2016.*

Guochuan, Z. ve Enyu, Y. (1998). The FFD Algorithm for the Bin Packing Problem with Kernel Items. In *Applied Mathematics-A Journal of Chinese Universities, pp. 2, 1998.*

Ha, R. ve Liu, J. (1994). Validating Timing Constraints in Multiprocessor and Distributed Real-Time Systems. In *Int'l Conf. on Distributed Computing Systems, pp. 162-171.*

Davis, R. I. ve Burns, A. (2011). A survey of hard real-time scheduling for multiprocessor systems. In *ACM Comput. Surv. 43, 4, Article 35, 44 pages.*

Lauzac, S., Melhem, R. & Mosse, D. (1998). Comparison of Global and Partitioning Schemes for Scheduling RM Tasks on a Multiprocessor. In *Euromicro Workshop on Real-Time Systems, June 1998.*

Liu, L. ve Layland, J. W. (1973). Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM, 1973.*

Palamut, S., Gönültaş, T., Elewi A., & Avaroğlu, E. (2019). Task Scheduling Algorithms and Resource Access Protocols. In *2019 International Artificial Intelligence and Data Processing Symposium (IDAP), Malatya, Turkey, pp. 1-6.*

Wu, J., ve Huang, Y. (2017). MCRTsim: A simulation tool for multi-core real-time systems. In *2017 International Conference on Applied System Innovation (ICASI), 461–464.*

Zapata, O. ve Alvarez, P. (2005). EDF and RM Multiprocessor Scheduling Algorithms: Survey and Performance Evaluation. In *Dept. Seccion de Computacion, CINVESTAV-IPN, Mexico City, Tech. Rep. CINVESTAV-CS-RTG-02*.

Elewi A., Shalan M., Awadalla M., & Saad E. M. (2014). Energy-efficient task allocation techniques for asymmetric multiprocessor embedded systems. In *ACM Transactions on Embedded Computing Systems (TECS) 13 (2s), 1-27*.

URL-1: https://github.com/SalihPalamut/MCRTsim [Erişim Tarihi: 15.06.2020]