Gerçek-Zamanlı Geribesleme Denetimli Görev Planlaması Tekniğinin RT-Linux'ta Uygulanması [1]

Tolga Ayav, Sinan Yılmaz
İzmir Yüksek Teknoloji Enstitüsü, Bilgisayar Mühendisliği Bölümü, 35430, Urla İzmir
Ege Üniversitesi, Bilgisayar Mühendisliği Bölümü, 35100, Bornova İzmir
TolgaAyav@iyte.edu.tr, Yilmaz@staff.ege.edu.tr

Özetçe

Bu çalışma kesinleşmemiş hesaplamalar yöntemine dayalı olan gerçek-zamanlı geribesleme denetimli bir görev planlaması tekniğinin ("Real-Time Feedback Control Scheduling"). Gerçek-Zamanlı Linux (RT-Linux) üzerindeki uygulamasını sunmaktadır. Zorunlu ve seçmeli alt-görevlerden oluşan ve geribesleme denetimli "rate-monotonic" tekniğine göre planlanan iki-sürümlü görev modeli seçilmiştir. Oransal-Entegral-Türev (PID) denetim, seçmeli alt-görevlerin çalıştırılması veya reddedilmesinde gerekli geribesleme stratejisini sağlamak için kullanılmıştır. Oluşturulan bu model RT-Linux işletim sistemine uygulanmış ve sistem başarımı yapay bir iş yükü altında sınanarak önerilen görev planlama modelinin sistemi öngörülen AİB kullanım oranı düzeyinde kararlı halde çalıştırabildiği gösterilmiştir.

Abstract

Implementation of Real-Time Adaptive Scheduling Technique on RT-Linux

This paper presents an RT-Linux implementation of real-time feedback control scheduling technique based on imprecise computations. We consider two-version tasks: one defined as mandatory and the other defined as optional for scheduling according to a feedback control rate-monotonic algorithm. A Proportional-Integral-Derivative (PID) control action provides the feedback strategy for deciding on the execution or rejection of the optional sub-tasks. The proposed model is implemented and evaluated on RT-Linux. The experimental results show that the proposed model can successfully keep the system stable at a desired level of CPU utilization.

1. Giriş

Gömülü sistemlerde geleneksel gerçek-zamanlı görev planlaması tekniklerinin çoğu iş yükünün ve çalışma-zamanı ortamının durağan ve kararlı olduğu varsayımına, bunun sonucu olarak da bu iş yükünün ve çalışma-zamanı ortamının tasarım sürecinde tamamıyla belirlenmiş olması esasına dayanmaktadır. Bu yaklaşımla birlikte iş yükünün modellemesinde olası en kötü durumların gözönünde bulundurulması gerçek-zamanlı sistemlerde boş bir kapasite doğurmaktadır (örn."rate-monotonic" görev planlaması tekniğinde işlemci kaynağının %69'unun kullanılması gibi).

[12], [3], [4] ve [5]'e göre gelecek kuşak gerçek-zamanlı sistemler, daha açık ve önceden tahmin edilemeyen ortamlarla karşı karşıya kalacak, bunun sonucunda da özel hedefe yönelik ve güvenlik açısından kritik fonksiyonların zaman kısıtlamalarını karşılamanın yanısıra daha karmaşık ve uyarlamacı yapılara sahip olacaktır. Örneğin, akıllı robot sistemlerinin iş yükünde, robot denetimi algoritmaları (algılayıcı yorumlama, hareket plan-

[1] Bu çalışma İzmir Yüksek Teknoloji Enstitüsü ve Institute Aéronautique et Spatial tarafından ortak yürütülen "Uydu Haberleşme ve Bağımlı Laboratuvarlar Projesi" kapsamında desteklenmektedir.

lama, ters kinematik ve dinamik algoritmaları vb.) değişken çalışma süreleri nedeniyle yüksek değişim ve aşırı yüklenme durumları oluşturmaktadır [7].

Geleneksel algoritmaların gerçek-zamanlı sistemlerdeki bu yeni gereksinimlere yeterli cevap veremeyeceği düşüncesiyle mevcut araştırmalar, yukarıda anılan bu boş kapasitenin değerlendirilmesi ve öngörülmeyen ortamlarda güvenilir çalışma sağlanma amacıyla uyarlamacı yaklaşımlar getirilmesi üzerine odaklanmaktadır. Bu çalışmanın da üzerinde duracağı geribesleme denetimli görev planlaması tekniği [12], [4], [11] ve [6]'da incelenmiş olmasına karşın yine bu çalışmalarda ortaya atılan ve yanıt bekleyen bir çok soruyu da beraberinde getirmektedir. Bu sorulardan bazıları kontrol sistemindeki denetlenen değişken, değiştirilen değişken ve ayar noktasının ne olması gerektiği, PID denetleinin yeterli olup olmadığı veya doğru denetim algoritmasının ne olması gerektiği, kontrol sisteminin kararlılık kriterlerinin ne olduğu ve geribesleme denetim ve deneticinin sisteme getirdiği ek yükün ne derece etkili olduğudur.

Bu çalışmada, yeni bir geribesleme denetimli "Rate-Monotonic" (RM) görev planlama tekniği sunulacak ve bu tekniğin RT-Linux işletim sistemine uygulaması anlatılacaktır. Bölüm 2.1'de kesinleşmemiş hesaplamalar yönteminden kısaca bahsedilecektir. Bölüm 2.2 ve 2.3'te çalışmada kullanılan görev ve iş yükü modelleri sunulacak, ardından geribesleme denetimli "rate-monotonic" görev planlama ayrıntılı olarak anlatılacaktır. Çalışmada denetim stratejisi olarak PID seçilmiştir. Bundaki amaç bu stratejinin getireceği ek yükün az olması ve daha önceki çalışmalarda sunulan PID denetimin sistemi kararlı halde çalıştırmadaki yeterliliğidir. Bölüm 3'te sistemin RT-Linux üzerindeki uygulamasının ayrıntıları verilmiş, son olarak da bölüm 4'te denemeler ve elde edilen sonuçlar aktarılmıştır.

2. Geribesleme Denetimli "Rate-Monotonic" Görev Planlaması Mimarisi

Bu bölümde, RM görev planlaması ve geribesleme denetim tekniklerini bütünleştiren geribesleme denetimli "rate-monotonic" (GK-RM) görev planlaması konusu incelenecektir. İlk olarak, bu bütünleşik tekniğin dayandığı kesinleşmemiş hesaplamalar yöntemine kısaca değinilecek, daha sonra denemelerde kullanılan görev modelleri ve iş yükü tanıtılacak, son olarak da GK-RM görev planlaması ayrıntılı olarak sunulacaktır.

2.1. Kesinleşmemiş Hesaplamalar Yöntemi

Gerçek-zamanlı sistemlerin aşırı yüklendigi durumlarda meydana gelebilecek planlama hatalarını karşılamanın bir yolu tüm *önemli* görevlerin zamanında tamamlanmasıdır. Diğer bir deyişle, işletim sistemi AİB'ni tüm görevlere eşit olarak atayacağına, bu kaynağı önemli görevler ("zorunlu"), ve ikincil ("seçmeli") görevler olarak kümelenecek görev sınıflarına dayanarak anahtarlayabilir. Bu yaklaşımda programcıların, zorunlu görevleri daima kendileri için tanımlanan son zamandan önce çalışmalarını bitirmek zorunda olanlar, seçmeli görevleri ise yeterli AİB kaynağı olduğunda çalıştırılacaklar biçiminde tanımlamaları gereklidir. Seçmeli görevler AİB'nin yüklenme durumlarında olası zamanlama hatalarının önlenmesi için atlanabilecektir. Herhangi bir görevin tüm kısıtlamalara uygun bir biçimde tamamıyla çalıştığında ürettiği sonuç tam veya kesin ("precise") sonuç olarak adlandırılabilir. Eğer görev atlanıyor veya sadece bir kısmı çalıştırılıyorsa görev tam olarak sonlandırılmaz ve elde edilen ara sonuç kesinleşmemiş ("imprecise") olarak adlandırılır. Örneğin, bir radardan gelen sinyalin işlenmesinde kullanılan filtreleme görevi aşırı yüklenme sırasında bir veya birkaç örnekleme dönemi boyunca çalıştırılmaz ise, bir önceki değer veya sensörden gelen işlenmemiş değer kullanılabilir. Kesinleşmemiş hesaplamalar yönteminin "Milestone", "Sieve" ve "Multiple-version" olarak adlandırılan uygulama yöntemleri [1] ve [2]'de ayrıntılı olarak incelenmiş olup aşağıda kısaca bahsedilecektir.

2.1.1. Uygulama Yöntemleri

Bir görevin ürettiği ara sonucun doğruluğu görev çalıştırılmaya devam ettikçe artıyorsa, bu görev tekdüze ("monotone")'dir. Eğer görev tamamlanmadan önce sonlandırılırsa bu görevden elde

edilen ara sonuç, sonlandırmadan önce üretilen tüm ara sonuçların içerisinde en iyisi olacaktır. Tekdüze görevler sayısal hesaplamalar, istatistiksel tahmin, buluşsal arama, sıralama ve veritabanı sorgulamaları için uygundur. Bir tekdüze görevden ara sonuç elde edebilmek için çalıştığı sürece bu görevin ürettiği ara sonuçların uygun anlarda saklanması ve görev süresinden önce durdurulduğunda kullanıma hazır durumda olması gerekliliğini "Milestone" olarak adlandırılıran bu yöntemin en büyük sakıncası ara sonuçların tutulmasının sisteme getireceği ek yüktür.

Bazı uygulamalarda tüm görevlerin tekdüze olması mümkün değildir. Bu durumda, "sieve" yöntemi uygulanabilir. Bu yöntemde her bir görev yukarıda radar örneğinde olduğu gibi ya tam olarak çalıştırılır yada atlanır. "Sieve" yönteminde görevler 0/1 kısıtlamasını karşılar ve daha az esnek bir görev planlamasına neden oluştururlar.

"Milestone" ve "sieve" yönteminin uygulanamadığı durumlarda ise "Multiple-version" yöntemi uygulanabilir. Bu yönteme göre görev en az iki sürümden oluşur: birincil sürüm ve ikincil sürüm(ler). Görevlerin birincil sürümü istenilen kesin sonucu üretir ancak daha uzun çalışma süresine sahiptirler. İkincil sürümler daha kısa süre çalışırlar ancak kesinleşmemiş sonuçlar üretirler. Geçici bir aşırı yüklenme durumunda, eğer tüm birincil sürümlerin zamanında tamamlanması olanaksız ise, sistem bazı görevlerin birincil sürümleri yerine ikincillerini çalıştırabilir. Daha esnek bir görev planlaması için fazla sayıda sürüm kullanılması ise tüm sürümlerin saklanmasının getireceği ek yük nedeniyle sakınca yaratabilir. Önceki tanımlamalar doğrultusunda, ikincil sürüm yalnız zorunlu alt-görevi içermekteyken, birincil sürüm hem zorunlu hem de seçmeli alt-görevi kapsar. Bu çalışmada ikili-sürüm yöntemi kullanılmıştır.

Görev planlaması açısından bakılacak olursa, kesinleşmemiş hesaplamalar yönteminin tam ve etkin çalışabilmesi için tüm zorunlu görevlerin sınırlı ve öngörülebilir kaynak ihtiyacına sahip olmaları gerekir ve zorunlu görevlere çalışmalarını belirlenen süre içinde tamamlayabilmeleri için yeterli işlemci zamanının verilmesi gerekir. Daha sonra sistem geri kalan AİB zamanını mümkün olduğunca çok sayıda seçmeli görevi çalıştırmak için kullanabilir. Belirli bir başarımı garantilemek için, zorunlu görevler daha geleneksel bir algoritmaya göre planlanır. Seçmeli görevler için ise daha dinamik bir politika uygulanabilir. Bu çalışma, zorunlu görevler için "rate-monotonic", seçmeli görevler içinse geribesleme denetimli görev planlaması tekniklerini önermektedir.

2.2. Görev Modelleri ve Çalışma Yükü

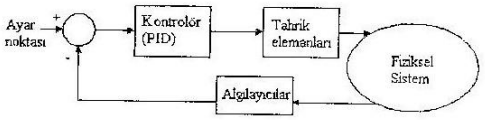
Geribesleme denetimli görev planlamasını uygulayabilmek için gerçek zamanlı sistemlerin klasik görev tanımlaması bir kaç küçük eklemeyle geliştirilebilir. Örneğin, q görevin oluşturduğu küme $\{\tau_1, \tau_2, ..., \tau_q\}$ olsun ve her bir görev, son zaman d_i (periyodik görevlerde periyot T_i olarak ele alınabilir), öncelik p_i, çalışma süresi C_i ve seçmeli kısım için izin biti e_i gibi bazı parametrelerle tanımlansın. Daha önce söylendiği gibi τ_i zorunlu alt-görev M_i ve seçmeli alt-görev O_i'den oluşmaktadır. M_i ve O_i'nin çalışma süreleri ise $C_{i,m}$ ve $C_{i,o}$ olsun. Bu durumda $C_{i,m} + C_{i,o} = C_i$. Klasik katı gerçek-zamanlı sistemler $C_{i,o} = 0, \ \forall i$ koşuluna sahip olup kesinleşmemiş hesaplama modelinin özel bir durumudur. Benzer şekilde, esnek gerçek-zamanlı sistemler ise $C_{i,m} = 0, \ \forall i$ özel durumudur. İkili-sürüm yönteminde görevler aşağıdaki yapıda oluşturulurlar:

```
τi {
   Zorunlu_Görevi();

   eğer(τi->ei==DOĞRU)
      Seçmeli_Görevi();
}
```

Eğer bir görev, zorunlu ve seçmeli olarak iki kesime ayrılamıyorsa aşağıda gösterildiği gibi iki sürümlü olarak da yazılabilir. Birinci sürüm, zorunlu ve seçmeli alt-görevleri içerir ve ikincisinden daha fazla çalışma süresi gerektirir.

```
τi {
   eğer(τi->ei==DOĞRU)
      Görev_Sürüm_Ai();
   değilse
      Görev_Sürüm_Bi();
}
```



Şekil 1: Geribesleme Denetim Sistemi

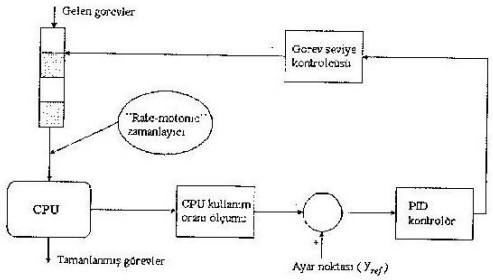
Daha sonra açıklanacağı gibi görevlerin, e_i durum değişkenlerine göre hangi sürümü çalıştıracağına veya seçmeli alt-görevin işletime alınıp alınmayacağına karar verilir. e_i değişkenleri ise görev düzey denetleyicisi tarafından kullanılır. Bu çalışmadaki tüm deneylerde, tablo 1'de yer alan parametreler kullanılmıştır. Ayrıca, tüm görevler periyodik olarak tanımlanmış ve çalışma süreleri $C_{i,m}$ ve $C_{i,o}$ birörnek olarak alınmıştır.

Tablo 1: Deneylerde kullanılan iş yükü ve sistem parametreleri

Değişken	Değerler
q (görev sayısı)	20
$C_{i,m}, C_{i,o}$	birörnek[15μs, 25μs]
$T_{i\in\{1,\ldots,q\}}$	{ 1s,2s,2s,2s,100ms,120ms, 130ms,170ms,12ms,12.2ms, 13.4ms,18.6ms,1.2ms,8.8ms, 6.6ms,9.5ms,100μs,110μs, 120μs,130μs}
T	100 ms
K_p, K_i, K_d	0.1, 2, 0.1
IW, y_{ref}	100, 0.7

2.3. GK-RM Görev Planlaması

"Rate-monotonic" görev planlaması tekniği, durağan öncelikli ve kaynakları geri alabilen (static priority driven preemptive) yöntemler sınıfında yer alır. Bu yöntemde her göreve çalışma sıklığıyla orantılı olan bir öncelik atanır. Görev planlayıcı tüm hazır görevler içerisinde en-yüksek önceliği olanı seçer ve AİB'nin kullanımını ona verir. "Rate-monotonic" literatürde önemli yeri olan, iyi tanımlanmış bir tekniktir ve AİB kullanım oranı aşağıdaki eşitsizliği sağladığı sürece tüm zorunlu alt-görevlerin son zamanlarından önce çalışmalarını



Şekil 2: GK-RM Görev Planlama

tamamlamasını garanti eder (Periyodik görevler için $f_i = 1/T_i$ alınır) [9].

$$\sum_{i=1}^{q} C_{i,m} f_i \leq q(2^{1/q} - 1) \qquad (1)$$

"Rate-monotonic" RT-Linux çekirdeğinin varsayılan görev planlama yöntemidir ve aşağıdaki tanımlandığı gibi çalışır.

```
Görevler= {τ1, τ2, ··· , τq},
τi = {di, Ci, pi, ei, ··· },
E={[tT, (t+1)T] zaman aralığında
kuyrukta hazır konumdaki görevler}.

''Rate-Monotonic'' Görev Planlayıcı{
  E kümesini hesapla;
  eğer(E==∅)
    Yeni_Görev=Std._Linux_Çekirdek;
  değilse
    Yeni_Görev=τarg(max i∈{1,2,...,q} pi);
  Yeni_Görev'e geç;
}
```

Öte yandan, seçmeli görevlerden hangilerinin veya kaçının çalıştırılacağına karar verilmesi çok daha karmaşık bir sorundur. Kapalı-döngü görev planlama olarak anılan geribesleme denetimli görev planlaması bu sorunun çözümü için ortaya atılan, klasik kontrol teorisine dayanan bir yöntemdir. "Rate-monotonic" ve "earliest-deadline-first" gibi algoritmalar ise açık-döngü düzeninde çalışırlar. Açık-döngü, tasarım sırasında öncelikler ve görev planlamaları bir kere belirlendikten sonra bunların üzerinde çalışma-zamanında geribeslemeye

dayalı herhangi bir değişiklik yapılamayacağını belirtir. Bu algoritmalar iş yükünün önceden modellenebildiği durağan ve devingen sistemlerde iyi başarım göstermekle birlikte önceden tahmin edilemeyen devingen sistemlerde başarısız olmaktadırlar. Geribesleme denetimli görev planlaması şekil 1'de görülen bir denetici, denetlenen fiziksel sistem, uyarıcı ve algılayıcı elemanlardan oluşan tipik bir denetim sistemine karşılık gelmektedir.

Kontrol teorisini uygulayabilmek için ilk olarak kontrol sisteminin elemanlarının bu çalışmada sunulan sistemdeki karşılıklarını bulmak gerekir. Bu uyarlama işinde çeşitli seçenekler bulunmakla birlikte [4], çalışmada denetlenen değişken olarak AİB kullanım oranı seçilmiştir. Katı gerçekzamanlı sistemlerde görevlerin zaman sınırlarını aşmalarına kesinlikle izin verilmediğinden, AİB kullanım oranının da %100'e ulaşmaması istenir. Bunun gerekçesi ise: geribesleme olarak ölçülen AİB kullanım oranı, %100'e ulaştığında sistemin ne kadar yüklü olduğuna ilişkin gerçek bilgiyi veremez ve bu da kontrol sisteminde doğrusal olmayan bir etki yaratır. Örneğin aşırı yüklenmiş bir sistemde gereksenen AİB kaynağı %150 olabilir ancak ölçülebilen değer en fazla %100 olacağından, ayar noktasının %99 olması durumunda kontrol sistemindeki hata değeri de %-1 olacaktır. Aslında hata değerinin %99-%150=%-51 olması gerekirken %-1 olması deneticinin gerekli tepkiyi verememesine sebep olur. Bu sebeple, ayar noktası (y_{ref}) %100'den yeteri kadar uzak seçilmelidir [2][10]. Bu çalışmada y_{ref} %70 olarak alınmıştır.

AİB kullanım oranının verilen ayar noktasında tutulabilmesi için denetim birimi ve çıkışını girdi olarak alan görev düzey deneticisi bir sonraki örnekleme aralığında çalıştırılacak seçmeli görevlerin sayısını ayarlar. Bu denetim mekanizmasının getireceği ek yükün kabul edilebilir düzeyde tutulabilmesi için denetim biriminin çıkışı $1/T$ sıklığında güncellenir. T örnekleme dönemi veya denetim dönemi olarak anılır. Denetim dönemi bu çalışmada sabit $100ms$ olarak alınmıştır. T'nin değişken olmasına ve o anki iş yüküne göre uyarlamacı bir yaklaşımla ayarlanmasına ilişkin bir çalışma [13]'te sunulmaktadır. Şekil 2 geri besleme denetimli "rate-monotonic" görev planlayıcı sisteminin yapısını göstermektedir. Burada görülen denetim ünitesinde Oransal-Entegral-Türev ("Proportional-Integral-Derivative" [PID]) işlevi iki nedenle kullanılmıştır. Birincisi, uygulanması kolay olan bu fonksiyonun sisteme getireceği ek yük azdır. İkincisi, [3], [4] ve [12]'de belirtildiği gibi PID denetim, sistemi kararlı halde tutabilmek için yeterlidir. PID denetim fonksiyonu aşağıdaki gibi tanımlanır.

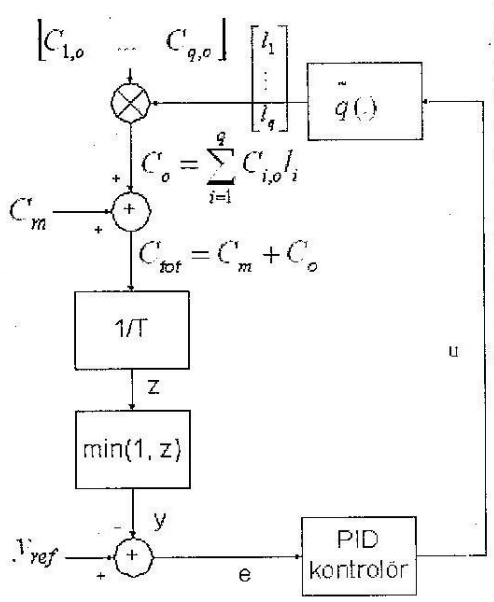
```
PID Denetici{
   e_t=Ayar noktası-AIB kullanım oranı;
   çıkış=K_p e_t + K_i Σ_{i=t-IW}^{t} e_i + K_d(e_t - e_{t-1});
   eğer(çıkış < -1/(2q)) çıkış=-1/(2q);
   eğer(çıkış > 1 + 1/(2q)) çıkış=1 + 1/(2q);
   e_{t-1}=e_t;
}
```

Bir görev AİB'nin denetimini aldığında ilk olarak zorunlu alt-görevini çalıştırır, sonra görev yapısında yer alan e_i değişkenine göre seçmeli alt-görevini çalıştırıp çalıştırmayacağına karar verir. Bu değişkenler görev düzey denetleyicisinin gözetimi altındadır. Görev düzey denetleyici, aşağıda gösterildiği gibi $[0, 1]$ aralığında değişen PID çıkışını alır ve bu değere göre seçmeli alt-görevlerini çalıştırması gereken görevlerin e_i değişkenlerini var ("1") konumuna, çalıştırmayacak olanlarınkini ise yok ("0") konumuna getirir.

```
Görev Seviye Deneticisi{
   döngü i : 1'den q'ya
      eğer(PID Çıkışı ≥ (2p_i - 1)/(2q))
         τ_i → e_i=1;
      değilse
         τ_i → e_i=0;
   döngü sonu
}
```

Sistemin çalışması şekil 3'te verilen matematiksel model yardımıyla daha net olarak açıklanabilir. Bu model bölüm 2.2'de açıklanan ve tablo 1'de verilen işyükü modeline göre oluşturulmuştur. Modelde görülen tüm sinyallerin bir denetim döneminde alacakları değerler aşağıda hesaplanacaktır. Bu hesaplamalarda $t \in \mathbb{N}$ kesikli zaman endeksidir, gerçek zaman $\tilde{t}$ ise Tt'ye eşittir. Çalışma dönemi $T_i(t)$ ile belirtilen görev τ_i'nin bir denetim dönemindeki çalışma sayısı $N_i(t) = T/T_i(t)$'dir. Bir denetim dönemindeki toplam görev sayısı

$$N(t) = \sum_{i=1}^{q} N_i(t) = \sum_{i=1}^{q} \frac{T}{T_i(t)}. \qquad (2)$$



Şekil 3: GK-RM görev planlayıcısının sistem modeli

τ_i'nin zorunlu ve seçmeli olmak üzere iki alt-görevden oluştuğu ve bu alt-görevlerin çalışma sürelerinin $C_{j,m}$ ve $C_{j,o}$ olduğu gözönüne alındığında, bu görevin t zamanındaki zorunlu ve seçmeli tüm alt-görevlerinin toplam çalışma süreleri şu şekilde hesaplanır:

$$C_{i,m,T}(t) = \sum_{j=1}^{N_i(t)} C_{j,m}(t),\ C_{i,o,T}(t) = \sum_{j=1}^{N_i(t)} C_{j,o}(t) \quad (3)$$

Bir denetim dönemindeki tüm görevlere ait zorunlu alt-görevlerin toplam çalışma süresi ise aşağıda verilmiştir:

$$C_m(t) = \sum_{i=1}^{q} C_{i,m,T}(t) \quad (4)$$

Tüm seçmeli alt-görevlerin toplam çalışma süresi görev planlama politikasına bağlıdır. Bu çalışmada seçmeli alt-görevler "sieve" yöntemine göre planlandığı için bu alt-görevler denetleyicinin çıkışına bağlı olarak tümüyle çalışacak yada işletim dışı kalacaklardır. Bu yapı, $l^k(t) \in \{0,1\}^q$, yani $l^k(t) = [1,1,...,1,0,0,...,0]^T$ vektörü ile tanımlanabilir. Öyle ki PID ve görev düzey denetleyicileri tarafından üretilen bu vektördeki k sayıda "1", bir sonraki dönemde çalıştırılacak olan ve en yüksek önceliklere sahip seçmeli alt-görevlere karşılık gelmektedir. Buna göre bir sonraki dönemde $\tau_1, \tau_2, ..., \tau_k$ görevlerinin seçmeli alt-görevleri çalışır. Formül (5), $t+1$ zamanında çalışacak olan tüm seçmeli alt-görevlerin toplam çalışma süresini $\tau_{1,o}, \tau_{2,o}, ...\tau_{q,o}$ görevlerinin $p_1 > p_2 > ... > p_q$ olacak şekilde sıralandığı varsayımıyla vermektedir.

$$C_o(t) = \left[\ C_{1,o,T}(t)...C_{q,o,T}(t)\ \right] \cdot \begin{bmatrix} l_1(t) \\ l_2(t) \\ \vdots \\ l_q(t) \end{bmatrix}^{(k)}$$

$$= \sum_{i=1}^{q} C_{i,o,T}(t).l_i(t) = \sum_{i=1}^{k} C_{i,o,T}(t) \quad (5)$$

Şekil 3'teki $z(t)$ ve AİB kullanım oranı $y(t)$ ise

$$z(t) = \frac{C_m(t) + C_o(t)}{T},\ y(t) = min\{1,\ z(t)\} \quad (6)$$

fonksiyonlarıyla gösterilir. Hata sinyali $e(t)$, ayar noktası y_{ref} ile ölçülen AİB kullanım oranının arasındaki farktır:

$$e(t) = y_{ref} - y(t) \quad (7)$$

Denetici bu hata sinyalini işleyerek $[0,1]$ aralığında bir değer üretir. Denetici PID ise bu durumda $u(t)$ şu formüle göre hesaplanır [8]:

$$u(t) = K_p e(t) + K_i \sum_{i=t-IW}^{t} e(i) + K_d(e(t) - e(t-1)) \quad (8)$$

Ardından, PID denetleyicisinin çıkışı şekil 3'te görülen $\tilde{q}(.)$ bloğu tarafından nicelendirilir. Bu işlem sözde-kodu daha önce verilen görev düzey denetleyicisi tarafından gerçekleştirilir. Nicelendiricinin fonksiyonu formül (9)'da verilmiştir. Nicelendirici $l^{(k)}(t)$ vektörünü elde etmede kullanılır ki bu vektör de önceden söylendiği gibi seçmeli alt-görevlerden hangilerinin çalıştırılacağını belirtir.

Örneğin $k = 3$ ise $l^{(3)} = [1, 1, 1, 0, ..., 0]^T$ olacak ve en yüksek p_1, p_2 and p_3 önceliklerine sahip 3 adet seçmeli alt-görev bir sonraki denetim döneminde çalıştırılacaktır. Görev düzey denetıcisinin çalışması aşağıda verilmiştir.

$$l^{(k)} = \left[\begin{array}{l} \left\{ \begin{array}{ll} 1 & \text{eğer} -\frac{1}{2q} \leq u < \frac{1}{2q} \\ 0 & \text{değilse} \end{array} \right\} \\ \left\{ \begin{array}{ll} 1 & \text{eğer} -\frac{1}{2q} \leq u < \frac{3}{2q} \\ 0 & \text{değilse} \end{array} \right\} \\ \vdots \\ \left\{ \begin{array}{ll} 1 & \text{eğer} -\frac{1}{2q} \leq u < \frac{1+2q}{2q} \\ 0 & \text{değilse} \end{array} \right\} \end{array} \right] \tag{9}$$

Bir başka deyişle çalıştırılacak seçmeli alt-görevlerin sayısı ise aşağıdaki gibi saptanır.

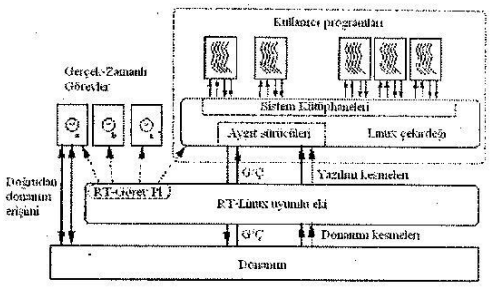
$$k = \left\{ \begin{array}{ll} 0 & \text{eğer} -\frac{1}{2q} \leq u < \frac{1}{2q} \\ 1 & \text{eğer} \frac{1}{2q} \leq u < \frac{3}{2q} \\ \vdots & \\ q & \text{eğer} \frac{2q-1}{2q} \leq u < \frac{2q+1}{2q} \end{array} \right. \tag{10}$$

3. GK-RM Tekniğinin RT-Linux Sisteminde Uygulanması

Bu çalışmada önerilen görev planlama tekniğini uygulayabilmek için RT-Linux işletim sistemi test ortamı olarak seçilmiştir. Bu bölümde ilk olarak RT-Linux ile ilgili çok kısa bilgi verilecek, daha sonra uygulamanın ayrıntıları anlatılacaktır.

3.1. RT-Linux Hakkında Bilgi

Gerçek-zamanlı sistemler açısından standart Linux'ta yaşanan problem, seçilemez kılınan kesmelerin yaratacağı öngörülemeyen gecikmelerdir. Bu problem Linux çekirdeği ile donanım kesme denetimi arasına yerleştirilen bir öykünüm programı ile aşılmıştır [14]. Linux kaynak kodunun içerisinde yer alan tüm *cli*, *sti* ve *iret* komutları *S_CLI*, *S_STI* ve *S_IRET* makroları ile değiştirilmiştir. Bu durumda oluşan tüm kesmeler ilk olarak öykünüm programı tarafından yakalanır. Linux kaynak kodunda bir kesme seçilemez kılındıysa, bunun yerine öykünme programındaki bir değişken sıfırlanır. Bir kesme oluştuğunda program önce bu değişkeni kontrol eder. Eğer bu değişken bir değere sahipse, Linux



Şekil 4: RT-Linux işletim sisteminin yapısı

bu kesmeyi seçilir kılmıştır demektir ve bunun sonucunda Linux kesme işleyicisi çalıştırılır. Aksi durumda kesme işleyicisi çalıştırılmaz yerine askıda bekleyen tüm kesmelerin bilgisinin tutulduğu bir değişkendeki ilgili bit belirlenir. Linux'un kesme işleyici üzerinde doğrudan bir denetimi olmadığından, gerçek-zamanlı kesmelerin işlenmesi Linux çekirdeğinin çalışmasından etkilenmez. Şekil 4 RT-Linux yapısını göstermektedir (RT-Linux ile ilgili daha detaylı bilgi için bknz. [14]). RT-Linux'ta gerçek-zamanlı görevler çekirdek birleşeni olarak ve yüksek başarım amacıyla çekirdeğin adres alanında tanımlanır. Bunun en büyük sakıncasıysa görevlerdeki en küçük bir yanlışın sistemin kilitlenmesine neden olabilmesidir. RT-Linux periyodik ve kesmelere bağlı çalıştırılabilen görevleri destekler ve çekirdek birleşeni olarak yüklenebilen küçük bir görev planlayıcısına sahiptir. Bu nedenle de görev planlayıcısının yeni bir sürümüyle değiştirilmesi oldukça kolaydır.

3.2. Uygulama

GK-RM planlayıcısının uygulanmasında ilk gereksinim duyulan AİB kullanım oranının ölçümü için aşağıdaki program parçası `rtl_schedule()` fonksiyonunun içerisine eklenmiştir:

```
now = sched -> clock -> gethrtime(
sched->clock ); /*fonksiyonun
başında system saati (nano saniye
cinsinden) alınır.*/
/*Görev planlama kararı burada
```

```
verilir*/
sched -> rtl_current -> tin-
sec += ( now-oldnow ); /*tin-
sec, görev yapısının içerisinde
tanımlanmıştır*/
now2 = sched -> clock -> geth-
rtime( sched->clock );
scheduler_overhead += ( now2-now
); /*Bu satır planlayıcının ek
yükünün hesaplanması içindir*/
oldnow = sched -> clock -> geth-
rtime(sched -> clock);
/*Görev değişimi (''task-
switching'') burada yapılır*/
```

Her denetim döneminde (çalışmada $100ms$ alınmıştır) bir kere çalışmak üzere yazılan "cpu_util" isimli periyodik iş parçacığı aşağıdaki programı çalıştırarak AİB kullanım oranını hesaplar:

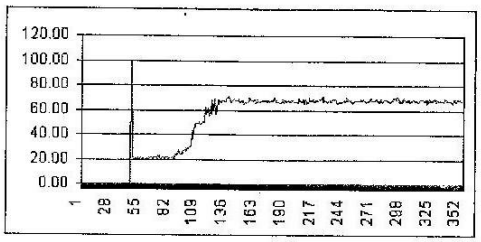
```
total_rt=linuxtime=0;
for(t=sched->rtl_tasks,i=0;
  t;
  t=t->next,i++){
 if(t==&sched->rtl_linux_task)
  linuxtime+=(float)t->tinsec;
 else total_rt+=(float)t->tinsec;

 t->tinsec = 0;
}
CPU_utilization = total_rt / (to-
tal_rt + linuxtime);
```

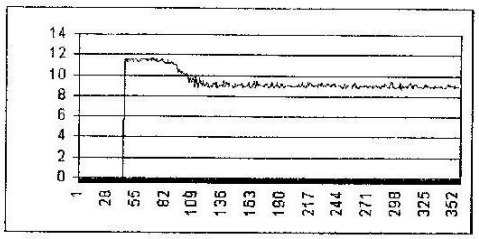
PID ve görev düzey deneticisi için de $100ms$ periyotlu birer iş parçacığı yazılmış olup, bunların programları daha önce verildiğinden burada daha fazla bir açıklamaya gerek duyulmamıştır.

4. Deneyler ve Sonuçlar

Sistem, periyotları tablo 1'de belirtilen 20 adet bağımsız periyodik görev kullanılarak test edilmiştir. Görevler, çalışma-zamanında birörnek dağılımdan üretilen çalışma sürelerine sahip olup, bu süre boyunca boş döngülerle işlemciyi oyalamaktadır. Şekil 5 bu denemeden elde edilen AİB kullanım oranını göstermektedir. Görüldüğü



Şekil 5: AİB kullanım oranı (ayar noktası=%70)



Şekil 6: Görev planlayıcının, PID hesaplaması ve AİB kullanım oranı ölçümü de dahil olmak üzere sisteme getirdiği ek yük

gibi, $t = 42$ anında 20 periyodik görev bir anda sisteme yüklenmiş, yaklaşık $t = 105$ anında ise sistem kararlı duruma ulaşarak %70 AİB kullanım oranını yakalamıştır. Tüm görevlerin bir anda yüklenmesindeki amaç, kontrol teorisinde sıkça uygulanan ve karşılaşılabilecek en kötü durumu ifade eden rampa girişe karşı sistemin tepkisinin ölçülmesidir. Şekil 6 ise görev planlayıcısının sisteme getirdiği ek yükü göstermektedir. "cpu_util" ve PID denetici iş parçacıklarının ek yükleri de sırasıyla %0.008 ve %0.019 olarak ölçülmüş olup, bunların görev planlayıcının ek yükü yanında ihmal edilebilir olduğu görülmüştür.

5. Gelecek Çalışmalar

Bu çalışmada, kesinleşmemiş hesaplamalar yöntemine dayalı GK-RM görev planlama tekniğinin RT-Linux işletim sistemine uygulanması sunulmuş ve deney sonuçları bu yaklaşımın AİB

kullanım oranını istenilen noktada kararlı halde başarıyla tutabildiğini göstermiştir. Bunun yanında sistem kararlığının teorik çalışmalarla ve daha gerçekçi iş yükleri altında incelenmesi gereklidir. Önerilen bir başka çalışma konusu da yine daha gerçekçi iş yükleri altında GK-RM görev planlayıcısının başarımının geleneksel görev planlama algoritmalarıyla karşılaştırmalarının yapılmasıdır.

6. Türkçe-İngilizce Sözlük

Ayar noktası:	Setpoint
Birörnek	Uniform
AİB kullanım oranı:	CPU Utilization
Görev planlayıcısı:	Scheduler
İş yükü:	Workload
Kesinleşmemiş hesaplamalar:	Imprecise Computations
Nicelendirici:	Quantizer
Son zaman:	Deadline
Öykünüm:	Emulation

Kaynakça

[1] **J.W.S. Liu, K.J. Lin, W.K. Shih, A.C.S. Yu, J.Y. Chung, W. Zhao**, "Algorithms for scheduling imprecise computations", IEEE Computer, May (1991) 58 – 68. (5) (1987).

[2] **Jane W.S. Liu, Wei-Kuan Shih, Kwei-Jay Lin, Riccardo Bettati and Jen-Yao Chung**, "Imprecise Computations", Proceedings of the IEEE, Vol. 82, No.1, January 1994.

[3] **Chenyang Lu, John A. Stankovic, Gang Tao, Sang H. Son**, "Design and Evaluation of a Feedback Control EDF Scheduling Algorithm". 20^{th} IEEE Real-Time Systems Symposium, 1999, pg. 56-67.

[4] **Chenyang Lu, John A. Stankovic, Gang Tao, Sang H. Son**, "Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms", Real-Time Systems Journal Special Issue on Control Theoretical Approach to Real-Time Computing 23(1/2):85-126 September, 2002.

[5] **Chenyang Lu, John A. Stankovic, Tarek F. Abdelzaher**, "Performance Specifications and Metrics for Adaptive Real-Time Systems", IEEE Real-Time System Symposium (RTS2000), December 2000.

[6] **D.A.Lawrence, J.Guan, S.Mehta, L.R. Welch**, "Adaptive Scheduling via Feedback Control for Dynamic Real-Time Systems", 20^{th} International Performance, Computing and Communications Conference, April 2001.

[7] **G. Beccari, et. al.**, "Rate Modulation of Soft Real-Time Tasks in Autonomous Robot Control Systems", Euro Micro Conference on Real-Time Systems, June 1999.

[8] **Gene F. Franklin, J. David Powell**, Abbas Emami-Naeini, 2002, Feedback Control of Dynamic Systems, Prentice Hall.

[9] **Liu, C.L., Layland, J.W.**, 1973. "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment". JACM. 20(1): 40-61.

[10] **Tolga Ayav, Sinan Yilmaz**, "Neuro-Fuzzy Controller in Real-Time Feedback Schedulers", In Proceedings of the 10^{th} Workshop on Nonlinear Dynamics of Electronic Systems, pp. 3.37-3.40, Izmir, Turkey, 2002.

[11] **J. Eker, P. Hagander, K. Arzen**, "A Feedback Scheduler for Real-Time Controller Tasks", In IFAC Control Engineering Practice, 2000.

[12] **John A. Stankovic, Chenyang Lu, Sang H. Son and Gang Tao**, "The Case for Feedback Control Real-Time Scheduling", In Proceedings of the 11^{th} Euromicro Conference on Real-Time Systems, pp. 11-20, York, UK, 1999.

[13] **Giorgio Buttazo and Luca Abeni**, "Adaptive rate control through elastic scheduling", Proceedings of the 39^{th} IEEE Conference on Decision and Control, Sydney, Australia, Dec. 2000.

[14] **Michael Barabanov**, "A Linux-based Real-Time Operating System", Master Thesis, New Mexico Institute of Mining and Technology, June 1997.