نمذجة وتطوير نظام معالجة تفرعية لجدولة وحشر مهام غير دورية ومزامنتها مع مهام دورية باستخدام خوارزميات عدة

د.م. عمار علي زقزوق

أستاذ مساعد في قسم هندسة التحكم الآلي والحواسيب

كلية الهندسة الميكانيكية والكهربائية – جامعة البعث

ملخص البحث

يقدم هذا البحث نمذجة وتطوير نظام معالجة تفرعية لتنفيذ مهام عدة على التوازي حسب درجة الأولوية باستخدام خوارزميات جدولة مهام دورية ومهام غير دورية بتقنيات مختلفة، وذلك اعتماداً على شروط ومعايير تختلف من خوارزمية لأخرى، سعياً في استغلال زمن المعالج.

اعتمد البحث على جدولة مهام دورية باستخدام خوارزمية الشريط الدوار (Round Robin) وخوارزمية الزمن الأقل خمولاً (Least Slake Time) وخوارزمية معدل التواتر (Rate Monotonic)، مع حشر مهام غير دورية ضمن هذه المهام الدورية باستخدام خوارزمية الخلفية (Background) وخوارزمية مخدم الانتخاب (Pooling Server) وخوارزمية المخدم المخادع (Deferrable Server).

تم تقويم أداء النظام المطور من خلال اختبار الخوارزميات المستخدمة ومقارنة النتائج التي تم الحصول عليها مع نتائج أبحاث أخرى، وقد أظهرت هذه النتائج فاعلية هذا النظام في الاستغلال الأعظمي لزمن المعالج وتقليل زمن المعالجة الكلي، وذلك من خلال التوصل إلى الآتي:

– أهمية استخدام خوارزمية Background في جدولة المهام غير الدورية، لما لها من دور في استغلال وقت المعالج الضائع أثناء حشرها مع مهام دورية.

– تحافظ خوارزمية Deferrable Server على ميزانية المعالج في حال كان رتل المهام غير الدورية فارغاً، وذلك بعكس خوارزمية Pooling Server التي تعمل على استنزاف الميزانية في حال عدم وجود مهمة غير دورية جاهزة للتنفيذ.

كلمات مفتاحية: خوارزميات الجدولة، المهام الدورية، المهام غير الدورية، استعمالية المعالج، نظم الزمن الحقيقي.

1- المقدمة:

بعد تقسيم نظام التشغيل للمهام، يجب أن يختار من بين هذه المهام المهمة الحالية من أجل تنفيذها على المعالج، ويتم ذلك وفقاً لأولويات محددة، إذ يمكن أن تعتبر المهمة ذات زمن التنفيذ الأطول هي ذات الأولوية الأعلى مثلاً، في حين تعتبر مجدولات أخرى أن المهمة ذات الزمن الحرج الأقل هي الأكثر أولوية، وتميل مجدولات أخرى إلى اعتبار أن المهمة ذات زمن الوصول الأقل هي الأعلى بالأولوية، وعلى أساس هذه التقنيات، تم تقسيم خوارزميات الجدولة إلى أنواع عدة.

الجدولة تعني عملية تقرير إذا كانت هناك مهمة أخرى ذات أفضلية مطلوب تنفيذها، وبالتالي وظيفة الجدولة هي السماح بتحقيق المتطلبات الزمنية عند تنفيذ التطبيقات، أي تنظيم عملية تنفيذ المهام لتلبية المتطلبات الزمنية لجميعها، وهذا يحدث في ثلاث حالات هي: إما أن تكون المهمة مقرر أن تنتظر لوقت معين للانتهاء من حدث ما أو لتحقيقه، أو عندما ترسل المهمة رسالة أو إشارة إلى مهمة أخرى، أو عندما ترسل المهمة إشارة إلى مهمة أخرى (حالة مقاطعة).

أما خوارزمية الجدولة فهي الطريقة الأفضل لحساب أفضلية المهمة في التنفيذ، وهناك نوعان أساسيان من خوارزميات الجدولة هما: النمط المباشر (online)، أي خلال تشغيل المهمة يتم إعادة الطلب ثانية لقبول الأفضليات، والنمط غير المباشر (offline)، يتم وفق إحصائيات سابقة لعملية تشغيل المهام [1].

في عام 2005، قارن الباحثان Arezou Mohammadi و Selim بين الخوارزميتين RM و LST، من خلال دراسة صفات وشروط كل منهما ومدى قابلية تطبيقهما [2].

فيما تركزت دراسة الباحث M. Kaladevi، في عام 2010، على مقارنة الخوارزميتين EDF وACO من حيث القدرة على العمل عند زيادة تحميل النظام [3].

في عام 2010، قدم الباحثان Yaashuwanth و Ramesh أداة تساعد في فهم عمل خوارزميات الجدولة الكلاسيكية مثل RM و DM و EDF بالاعتماد على بنية LabVIEW، ولكنهما لم يتطرقا إلى حشر مهام غير دورية مع المهام الدورية [4].

في عام 2017، قام الباحثان Sumit Kumar Nager و Nasib Singh Gill بالمقارنة بين الخوارزميتين RM و EDF، إذ وجدا أن RM أفضل من EDF من حيث

التعقيد الزمني [5].

في عام 2017، عملت مجموعة من الباحثين على تصميم نظام لجدولة المهام الدورية وحشر مهام غير دورية، إذ تم تضمين عدد من خوارزميات جدولة المهام الدورية بالإضافة إلى خوارزميتين لحشر المهام غير الدورية هما Background و Polling Server [6].

مما سبق، نلاحظ أن أغلب الأبحاث ركزت على دراسة أداء جدولة المهام الدورية ومقارنتها، والبعض منها قام بحشر مهام غير دورية، أما هذا البحث فيعمل على جدولة أنواع عدة من المهام كالمهام الدورية وغير الدورية معاً باستخدام خوارزميات متعددة ومناقشة الأداء بالنسبة لزمن المعالج.

2- أهمية البحث وأهدافه:

يعتبر الفهم الداخلي لعمل المعالج وكيفية ترتيب المهام وتنفيذها ضمنه وأولوياتها من أهم القضايا التي تواجه الباحثين والدارسين في مجال المعالجات وتصميمها وتطويرها، لذلك فهذا البحث يقدم الآتي:

- نظام يوصف عمل أهم خوارزميات الجدولة المختلفة، ما يساهم في فهم العمل الداخلي للمعالجات وتبسيطه.
- مقارنات بين تلك الخوارزميات، من حيث شروط تطبيقها ونتائجها وتحليل أدائها.

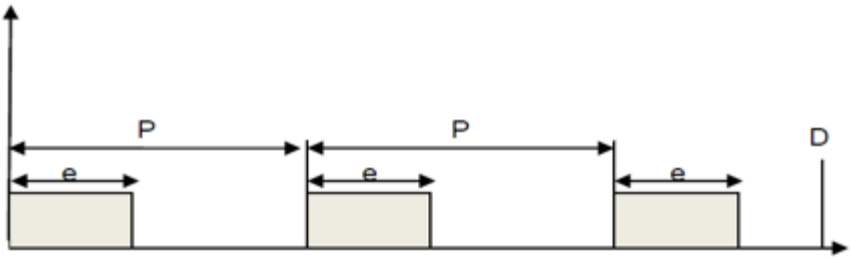
لذلك، فالبحث يهدف إلى دراسة نظام يدعم عمل أهم خوارزميات الجدولة في الزمن الحقيقي وتطويره، سعياً في توضيح كيفية عمل كل نوع من هذه الخوارزميات والفروق الأساسية فيما بينها، ومعرفة أين يمكن استخدام كل نوع من هذه الخوارزميات.

وقد افترضنا أننا نتعامل مع نظام يطلب منه تنفيذ سلسلة من المهام، ومناقشة الحالة المثالية عند حدوث مقاطعة حرجة من قبل مهمة أخرى لها أولوية أعلى من ناحية الزمن المنقضي أثناء عملية التبادل، كما أنه يتيح إمكانية أخذ هذا الزمن بالحسبان.

3- أساسيات البحث [1]:

1.3- المهام الدورية والمهام غير الدورية:

المهام الدورية: هي المهام التي تنطلق في فترات زمنية متتالية متساوية، وترجع إلى حالتها الخاملة عد انتهاء الطلب، كما هو مبين في الشكل (1).



الشكل (1): نموذج المهمة الدورية.

المهام غير الدورية: هي المهام التي تنطلق في فترات زمنية غير متساوية (عشوائية).

2.3- الواصفات الزمنية:

Proceedings: أحياناً لا يمكن تنفيذ مهمة معينة قبل نهاية تنفيذ مهمة أخرى، أي أن تنفيذ إحدى المهمتين مرتبط بتنفيذ المهمة الأخرى.

Arrival Time or Release Time: اللحظة التي تتكرر فيها مهمة دورية، وبالنسبة لمهمة غير دورية هي اللحظة التي وصل فيها الحدث.

Phase: زمن أول عملية تحرير للمهمة.

Period: الفترة الزمنية التي تفصل بين عملية تحرير حالي وتحرير تالي لمهمة.

Deadline: يمثل الزمن المسموح ضمنه للمهمة أن تنهي تنفيذها، فإن انقضى هذا الزمن ولم تنتهي المهمة فلا يعود هناك أي داعي لإكمال تنفيذها.

Response Time: اللحظة بين تحرير الإجراء وتنفيذه.

3.3- معايير قياس أداء خوارزميات الجدولة:

Utilization: نسبة استعمالية المعالج خلال عملية الجدولة، وتعني مقدار الزمن الذي بقي المعالج فيه يعمل بدءاً من لحظة بداية عملية الجدولة وحتى انتهائها، وكلما كان هذا الزمن أكبر كلما كانت عملية الجدولة أفضل، ويجب ألا تتجاوز قيمته درجة استعمالية المعالج القصوى (Utilization Max)، وإلا فإن المشكلة غير قابلة للجدولة، وهو يقاس بمجموع نسب أزمنة تنفيذ المهام (e_i) على دور كل منها (p_i) وفق العلاقة (1).

$$U = \sum_{i=1}^{n} \frac{e_i}{p_i} \qquad (1)$$

Utilization Max: درجة استعمالية المعالج القصوى، ويجب أن يبقى مستوى استعمالية المعالج أقل من هذا الحد.

Hyperperiod: الزمن الأعظمي اللازم لإتمام عملية الجدولة، أو هو المضاعف المشترك الأصغر للأدوار الذي فيه تتم عملية التحرير لجميع الإجرائيات.

4.3- خوارزميات الجدولة المستخدمة في البحث:

1.4.3- خوارزميات جدولة المهام الدورية:

Round Robin (RR): يجري تنفيذ الإجراءات تسلسلياً حتى تنتهي، إذ يخصص لكل إجراء شريحة (حصة) زمنية ثابتة ينفذ خلالها، وتستعمل ساعة ذات دور ثابت يساوي الحصة الزمنية لمقاطعة النظام دورياً. يستمر تنفيذ الإجراء حتى ينتهي تنفيذه أو تنتهي حصته الزمنية بحدوث مقاطعة الساعة. إذا لم ينتهي تنفيذ الإجراء، يحفظ سياقه ويوضع في نهاية رتل الإجراءات الجاهزة. بعد ذلك، يسترجع النظام سياق الإجراء التالي الموجود في بداية الرتل ويتابع تنفيذه. تعطي هذه الخوارزمية جدولة عادلة غير شفعية للإجراءات التي لها الأولوية نفسها، بتوزيع زمن المعالج عليها بالتساوي [7].

Least Slack Time (LST) or Least Laxity First: تحسب الأفضليات عند دخول مهمة جديدة، ويستخدم عامل الخمول في تحديد الأولويات، وتعطى المهمة الأقل ركوداً أفضلية عليا [8].

يعرف زمن الخمول على أنه ناتج طرح زمن التنفيذ (C) من الزمن الحرج (D) كما في العلاقة (2).

$$L = D - C \qquad (2)$$

Rate Monotonic (RM): في النظم ذات الأولوية الثابتة، يعطى لكل إجراء أولوية محددة نسبة لباقي الإجراءات، وهذه الأولوية تتناسب عكساً مع دوره، فأولوية الإجراء الذي له دور كبير تكون أصغر من أولوية الإجراء الذي له دور قصير [9].

2.4.3- خوارزميات جدولة المهام غير الدورية:

Budget: هي الميزانية أو ميزانية المعالج، وتمثل الزمن المتاح من قبل المعالج لتنفيذ المهام عليه. تعتمد عملية جدولة المهام غير الدورية مع المهام الدورية على كيفية استهلاك الميزانية

المتاحة من قبل المهام وكيفية إعادة ملء هذه الميزانية من جديد، وتسمى هاتان العمليتان بالاستهلاك والملء.

هناك خوارزميات عدة لتنفيذ هاتين العمليتين، سنستخدم منها الثلاث الآتية:

Background: يعني أن المهام غير الدورية تملك الأولوية الأدنى من بين جميع المهام، وبالتالي سيتم تنفيذ المهام الدورية أولاً ثم المهام غير الدورية، وقد يسبب هذا تأخير عملية تنفيذ المهام غير الدورية [1].

Pooling Server: تنشط المهمة الدورية في بداية الدور الذاتي للمخدم ضمن حدود سعته، وفي حال عدم وجود مهمة، يلغي المخدم نفسه حتى بداية الدور الثاني، ويخصص الوقت المحجوز للمهام الدورية، ولكن من سيئاته أنه يقلل من عامل استخدام المعالج [1].

Deferrable Server: هذا النوع من المخدمات يختلف عن السابق بحالة واحدة هي أنه يبقى موجوداً بعد إلغاء نفسه نتيجة لإنهاء المهمة غير الدورية، وبالتالي يمكنه تخديم مهام غير دورية خلال الفترة التي تفصله عن بداية الدور الثاني [10].

4- تصميم النظام المقترح:

يبين الشكل (2) الواجهة الأساسية للنظام المصمم المقترح.

يوضح الشكل (3) المخطط الصندوقي لمراحل عمل النظام المصمم المقترح.

5- النتائج ومناقشتها:

1.5- الجدولة وفق خوارزمية Round Robin:

لدينا ثلاث مهام تعطى بياناتها وفق الجدول (1)، وتكون قيم الأوزان كلها 1، أي أن لكل المهام الأولوية نفسها والحق ذاته باستعمال المعالج، وتتم الجدولة باستخدام مبدأ المشاركة الزمنية، إذ تنفذ المهام بالتناوب بمقدار فاصل زمني يساوي زمن الانتقال لكل مهمة.

يبين الشكل (4) نتيجة تطبيق خوارزمية Round Robin على البيانات المعطاة في الجدول (1).

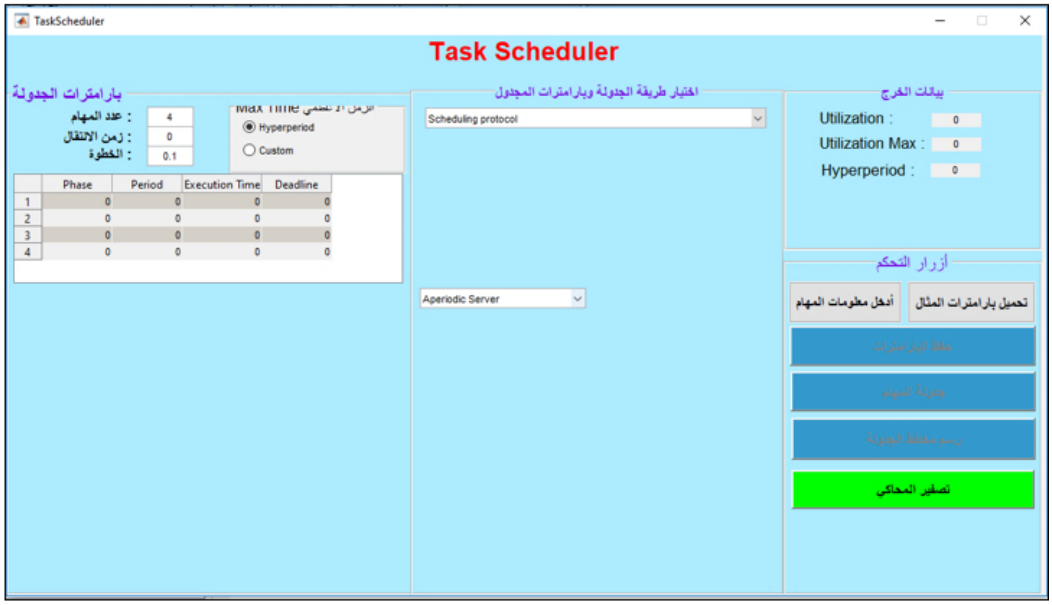
من الشكل (4)، نستنتج الآتي:

- أن المهمتان 1 و 3 بدأتا بالتنفيذ مباشرة وبشكل متناوب، أما المهمة 2 فقد انتظرت حتى الزمن 2 لأن زمن انطلاق هذه المهمة هو 2.

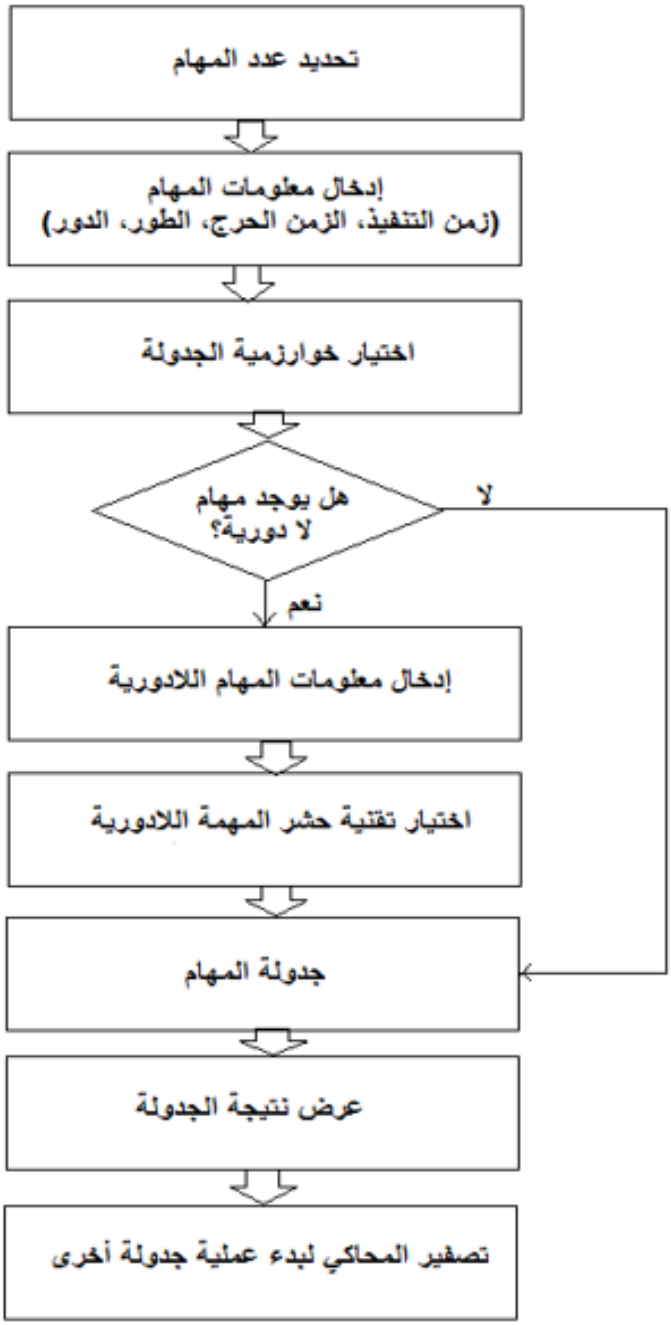
– يتم تخديم المهمة التي تصل أولاً في البداية ثم التالية، وهكذا حتى انتهاء المهام. تخدم المهمة الواحدة بمقدار زمن يسمى quantum، وهو زمن بالثواني يحدد من خلاله المقدار الزمني الذي يعطيه المعالج لكل مهمة خلال مرة التنفيذ الواحدة. يمكن ملاحظة أن المهمتان الأولى والثالثة تنفذان بطريقة المشاركة الزمنية، فبمجرد أن المهمة الأولى تنفذ بمقدار 0.1 ثانية تترك المعالج وتحل محلها المهمة الثالثة، وما إن تنفذ المهمة الثالثة بمقدار 0.1 تترك المعالج وتحل محلها المهمة الأولى مجدداً، علماً أن الزمن 0.1 هو زمن quantum.

الوزن	الزمن الحرج	زمن التنفيذ	الدور	زمن الوصول
1	3	1	3	0
1	3.5	1.5	3.5	2
1	6.5	0.5	6.5	0

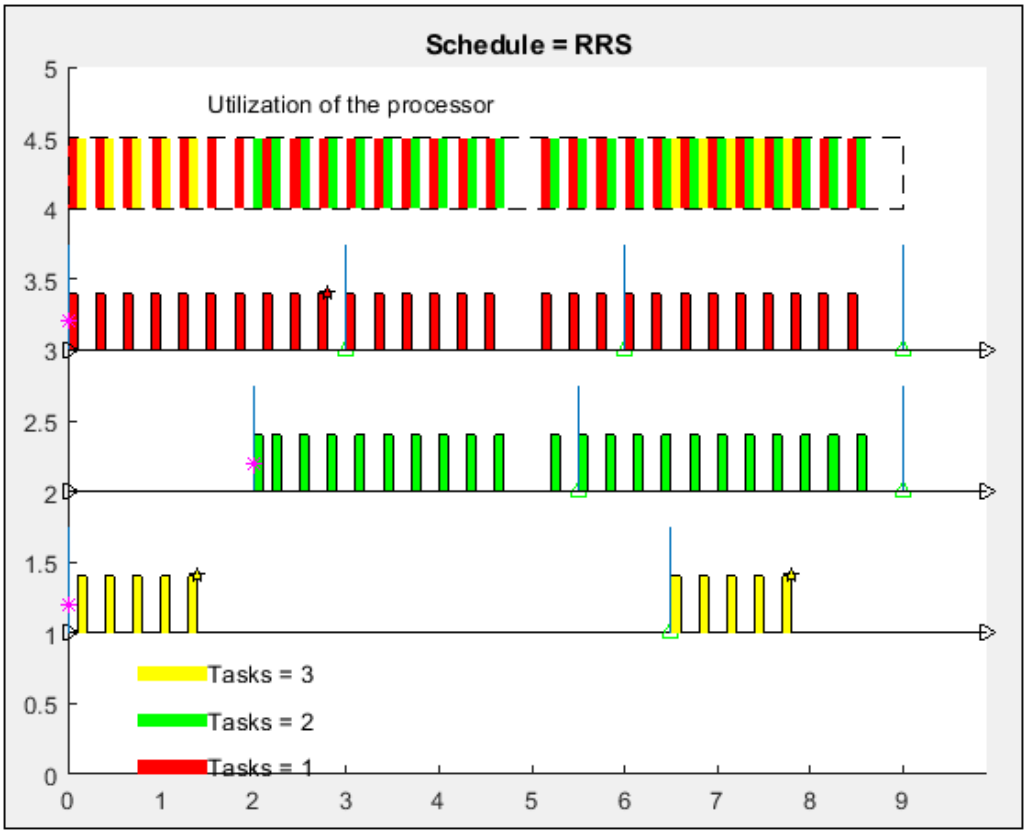
الجدول (1): البيانات المطبقة في خوارزمية Round Robin.



الشكل (2): الواجهة الأساسية للنظام المصمم المقترح.



الشكل (3): المخطط الصندوقي لمراحل عمل النظام المصمم المقترح.



الشكل (4): مخطط الجدولة باستخدام خوارزمية Round Robin.

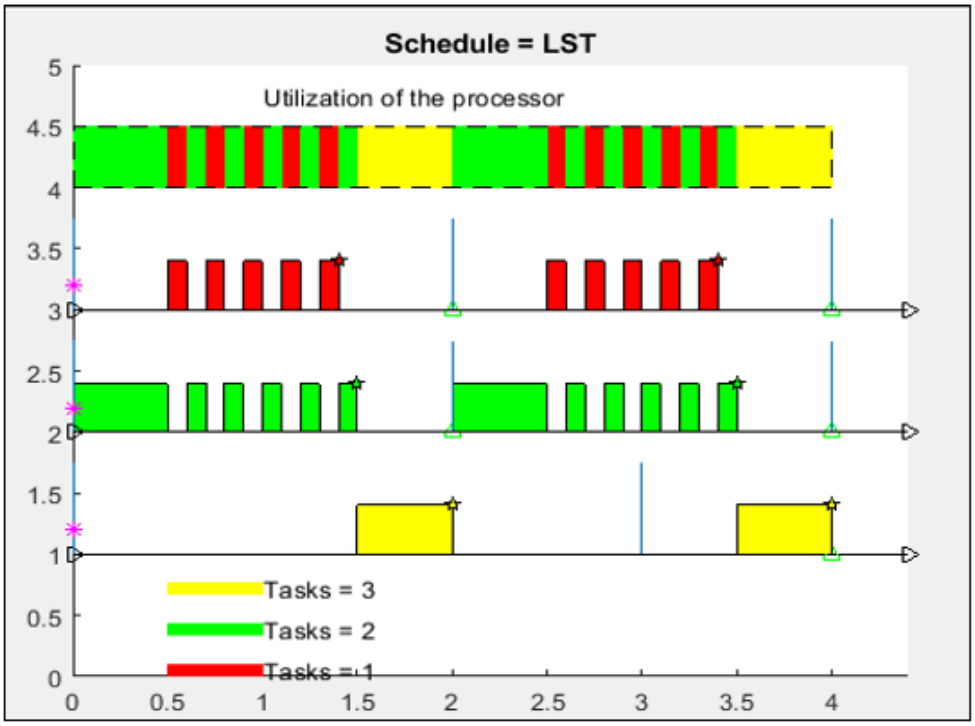
2.5- الجدولة وفق خوارزمية Least Slack Time:

لدينا ثلاث مهام تعطى بياناتها وفق الجدول (2).

الزمن الحرج	زمن التنفيذ	الدور	زمن الوصول
2	0.5	2	0
2	1	2	0
4	0.5	3	0

الجدول (2): البيانات المطبقة في خوارزمية Least Slack Time.

يوضح الشكل (5) مخطط الجدولة الناتج من تطبيق خوارزمية Least Slack Time على البيانات المعطاة في الجدول (2).



الشكل (5): مخطط الجدولة باستخدام خوارزمية Least Slack Time.

من الشكل (5)، نستنتج الآتي:

– تعتمد هذه الخوارزمية على حساب زمن الخمول في كل لحظة زمنية، وتعطى للمهمة ذات زمن الخمول الأقل الأولوية العليا، وفي حال تساوي زمن الخمول في لحظة ما، فتتم الجدولة عندها بالتناوب بمبدأ المشاركة الزمنية.

– بالعودة إلى مخطط الجدولة السابق، يتم تنفيذ المهام بناءً على زمن الخمول (المعطى بالعلاقة (2)) وفق اللحظات الزمنية الآتية:

- في اللحظة الزمنية t=0:

$$L_1 = 2 - 0.5 = 1.5$$

$$L_2 = 2 - 1 = \mathbf{1}$$

$$L_3 = 4 - 0.5 = 3.5$$

نلاحظ أن للمهمة الثانية زمن الخمول الأقل، وبالتالي لها الأولوية العليا وتبدأ بالتنفيذ.

- في اللحظة الزمنية t=0.5:

$$L_1 = 2 - 0.5 = \mathbf{1.5}$$
$$L_2 = 2 - 0.5 = \mathbf{1.5}$$
$$L_3 = 4 - 0.5 = 3.5$$

نلاحظ أن للمهمتين الأولى والثانية زمن الخمول ذاته، وبالتالي لهما الأولوية نفسها، ويتم التنفيذ بالتناوب بينهما.

- في اللحظة الزمنية t=1:

$$L_1 = 2 - 0.25 = \mathbf{1.75}$$
$$L_2 = 2 - 0.25 = \mathbf{1.75}$$
$$L_3 = 4 - 0.5 = 3.5$$

نلاحظ أن للمهمتين الأولى والثانية زمن الخمول نفسه، وبالتالي لهما الأولوية ذاتها، ويتم التنفيذ بالتناوب بينهما.

- في اللحظة الزمنية t=1.5:

$$L_1 = 2 - 0 = \mathbf{2}$$
$$L_2 = 2 - 0 = \mathbf{2}$$
$$L_3 = 4 - 0.5 = 3.5$$

نلاحظ أن المهمة الأولى (الحمراء) قد أتمت زمن تنفيذها البالغ 0.5sec، وكذلك المهمة الثانية (الخضراء) البالغ دورها 1sec، وبالتالي عليها انتظار بدء دور جديد، ونتيجة لذلك تبدأ المهمة الثالثة (الصفراء) بالتنفيذ.

- في اللحظة الزمنية t=2:

تكون المهمة الثالثة قد نفذت زمن تشغيلها البالغ 0.5sec، إذ تحدث مقاطعتها بسبب دخول المهمتين الأولى والثانية بالمنافسة على زمن المعالج من جديد، وذلك بسبب أنه في اللحظة t=2 تبدأ كل من المهمة الأولى والثانية دوراً جديداً، لذلك نحسب زمن الخمول لكل مهمة في هذه اللحظة:

$$L_1 = 2 - 0.5 = 1.5$$
$$L_2 = 2 - 1 = \mathbf{1}$$
$$L_3 = 4 - 0 = 4$$

نلاحظ أن للمهمة الثانية زمن الخمول الأقل، لذلك لها الأولوية بالتنفيذ.

3.5- تحليل أداء خوارزمية LST بوجود مهام غير دورية وفق خوارزمية Background:

بفرض لدينا ثلاث مهام دورية تعطى بياناتها وفق الجدول (3)، ونريد حشر مهام غير دورية مع تلك المهام الدورية باستخدام خوارزمية Background، بحيث تملك المهام غير الدورية البيانات المعطاة في الجدول (4).

الزمن الحرج	زمن التنفيذ	الدور	زمن الوصول
20	4	20	0
15	3	15	0
15	4	15	0

الجدول (3): بيانات المهام الدورية المستخدمة من قبل خوارزمية LST.

زمن التنفيذ	الطور
0.5	0
2	10

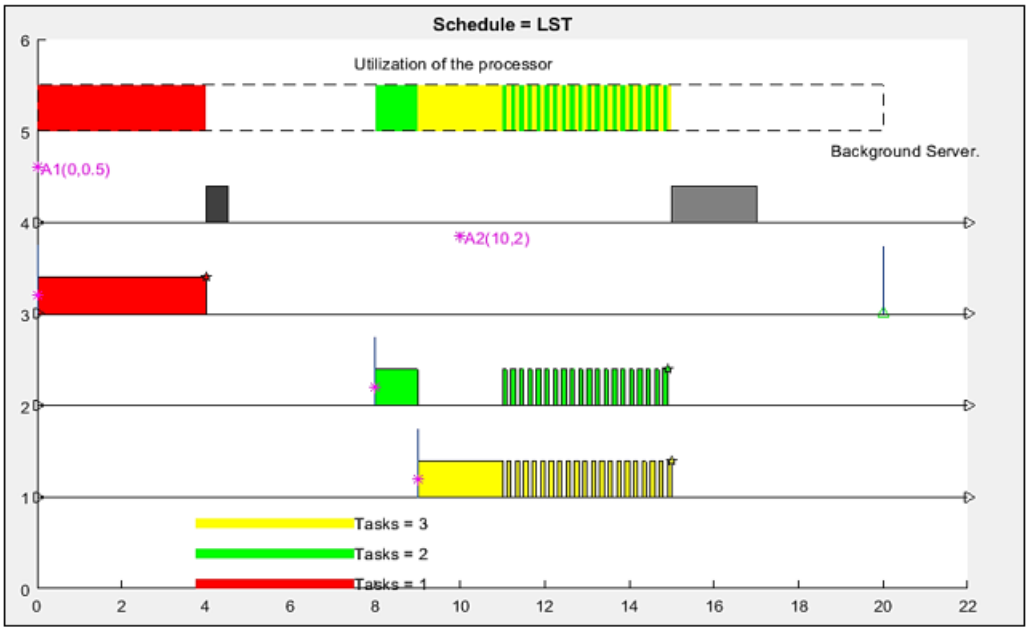
الجدول (4): بيانات المهام غير الدورية المستخدمة من قبل خوارزمية Background.

يظهر الشكل (6) مخطط الجدولة الناتج من تطبيق خوارزمية LST مع خوارزمية Background على البيانات المعطاة في الجدولين (3) و (4).

من الشكل (6)، نستنتج الآتي:

- بالنسبة للمهام غير الدورية باستخدام خوارزمية Background، فإنها لا تعمل على مقاطعة أية مهمة، وإنما تنتظر خلو المعالج من أية مهمة، أي تقوم بالتنفيذ في وقت فراغ المعالج.
- يكون المعالج في الفترة الزمنية الممتدة من الثانية 4 حتى الثانية 8 في حالة فراغ، وبما أن المهمة الأولى غير الدورية التي تملك زمن وصول 0 وزمن تنفيذ 0.5sec تكون جاهزة، لذلك يتم استغلال حالة فراغ المعالج وتنفذ هذه المهمة في اللحظة t=4 وتستمر لمدة 0.5sec.
- إن المهمة غير الدورية الثانية التي تملك زمن وصول 10 وزمن تشغيل 2sec قد وصلت منذ اللحظة t=10 ولكن لا يتاح لها التنفيذ حتى اللحظة t=15، أي لحظة انتهاء تنفيذ المهام

الدوريـة، إذ تنفذ بمقدار زمـن تشـغيلها البـالغ 2sec، ويـدخل المعـالج بعـدها بحالـة فـراغ حتـى اللحظة t=20.



الشكل (6): مخطط الجدولة باستخدام خوارزمية LST مع خوارزمية Background.

بالنتيجة، تظهر أهمية استخدام خوارزمية Background في جدولة المهام غير الدورية لما لها من دور في استغلال وقت المعالج الضائع أثناء حشرها مع مهام دورية.

4.5- تحليل أداء خوارزمية RM بوجود مهام غير دورية وفق خوارزمية Polling Server:

لدينا مهمتان دوريتان تعطى بياناتهما كما في الجدول (5).

الزمن الحرج	زمن التنفيذ	الدور	زمن الوصول
3	1	3	0
10	4	10	0

الجدول (5): بيانات المهمتين الدوريتين المستخدمة من قبل خوارزمية RM.

وذلك بفرض أن الخطوة تساوي 0.1 وزمن الانتقال يساوي 0.

نريد حشر مهمة غير دورية مع تلك المهام الدورية باستخدام خوارزمية Polling Server، بحيث تملك المهمة غير الدورية البيانات المبينة في الجدول (6).

زمن التنفيذ	الطور
0.8	0.1

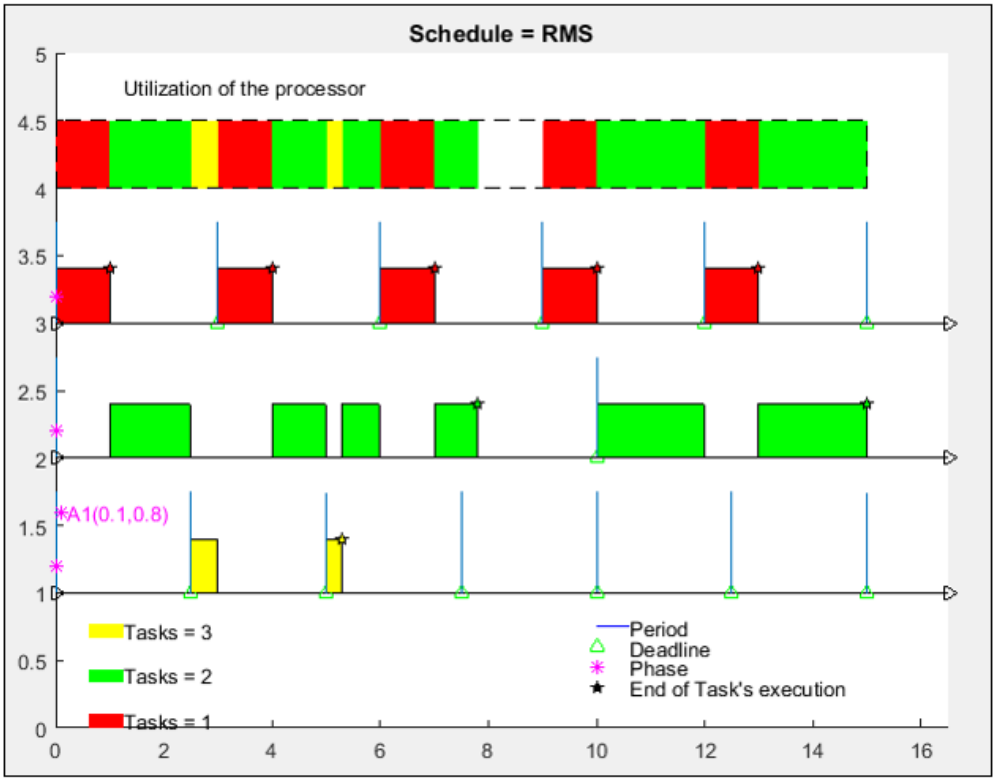
الجدول (6): بيانات المهمة غير الدورية المستخدمة من قبل خوارزمية Polling Server.

والتي نمثلها بوصفها مهمة دورية ببيانات الجدول (7).

الزمن الحرج	زمن التنفيذ	الدور	زمن الوصول
2.5	0.5	2.5	0

الجدول (7): تمثيل المهمة غير الدورية المستخدمة من قبل خوارزمية Polling Server بوصفها مهمة دورية.

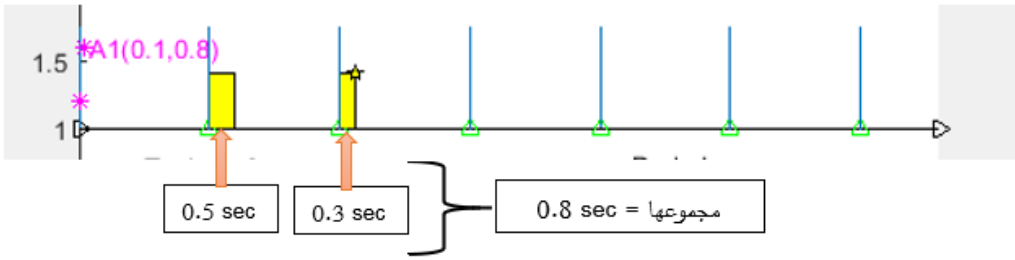
يكون مخطط الجدولة الناتج كما في الشكل (7).



الشكل (7): مخطط الجدولة باستخدام خوارزمية RM مع خوارزمية Polling Server.

من الشكل (7)، نستنتج الآتي:

بالنسبة للمهام الدورية، نستخدم هنا خوارزمية RM التي تقوم بالجدولة اعتماداً على أدوار المهمات، بحيث تأخذ المهمة ذات الدور الأقل الأفضلية العليا، وبما أن زمن الوصول صفري لجميع المهام، فإن كل المهام تدخل في حيز المنافسة في اللحظة 0، ولكن هنا المهمة غير الدورية هي المهمة الثالثة (اللون الأصفر) ولا يمكن البدء بتنفيذها إلا بعد مرور زمن دور كامل، أي بعد ثانيتين ونصف. لذلك، في بداية التنفيذ تتنافس المهمتان الأولى والثانية فقط. بسبب استخدام خوارزمية RM، فإن المهمة الأولى تنفذ أولاً لأنها تملك الدور الأقل، حيث دورها 3 ودور المهمة الثانية يساوي 10. تستمر المهمة الأولى بالتنفيذ لمدة 1sec، وبعدها تبدأ المهمة الثانية بالتنفيذ لمدة 1.5sec فقط، وذلك بمقاطعتها في اللحظة 2.5 بسبب انقضاء أول دور للمهمة غير الدورية ودخولها حيز المنافسة، وهنا تنفذ المهمة الثالثة (غير الدورية) لمدة 0.5sec فقط، مع العلم أن زمن تنفيذها هو 0.8sec، وذلك لأن المهمة الأولى تدخل في دور جديد بعد انقضاء دور كامل لها وتنفذ لمدة 1sec، وبعدها تدخل المهمة الثانية في التنفيذ حتى اللحظة 5، وعندها تعود المهمة الثالثة (غير الدورية) لتنفذ لمدة 0.3sec فقط، وبذلك يكون مجموع زمن تنفيذ المهمة غير الدورية: 0.8sec = 0.3sec + 0.5sec، كما هو مبين في الشكل (8).



الشكل (8): زمن تنفيذ المهمة غير الدورية بخوارزمية Polling Server مع المهمتين الدوريتين بخوارزمية RM.

5.5- تحليل أداء خوارزمية RM بوجود مهام غير دورية وفق خوارزمية Deferrable Server:

لدينا مهمتان دوريتان تعطى بياناتهما كما في الجدول (8). وذلك بفرض أن الخطوة تساوي 0.1 وزمن الانتقال يساوي 0.

الزمن الحرج	زمن التنفيذ	الدور	زمن الوصول
3.5	1.5	3.5	2
6.5	0.5	6.5	0

الجدول (8): بيانات المهمتين الدوريتين المستخدمة من قبل خوارزمية RM.

نريد حشر مهمة غير دورية مع تلك المهام الدورية باستخدام خوارزمية Deferrable Server، بحيث تملك المهمة غير الدورية البيانات المعطاة في الجدول (9).

زمن التنفيذ	الطور
1.7	2.8

الجدول (9): بيانات المهمة غير الدورية المستخدمة من قبل خوارزمية Deferrable Server.

والتي يتم تركيبها على مهمة دورية لها بيانات الجدول (10).

الزمن الحرج	زمن التنفيذ	الدور	زمن الوصول
3	1	3	0

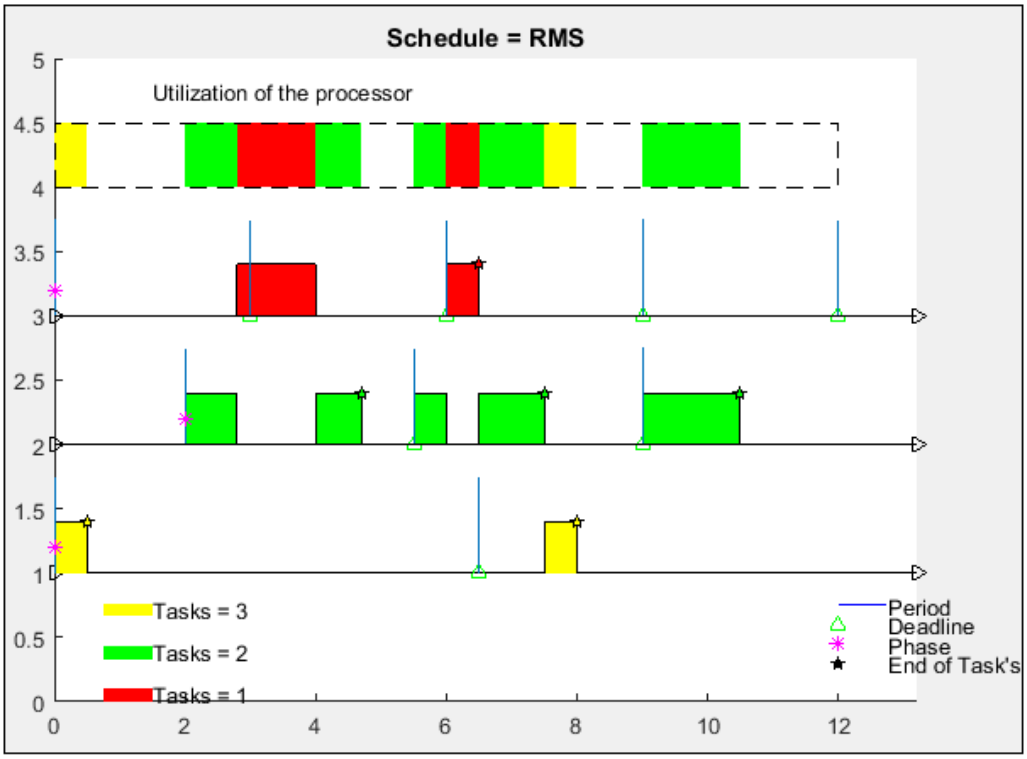
الجدول (10): تمثيل المهمة غير الدورية المستخدمة من قبل خوارزمية Deferrable Server بوصفها مهمة دورية.

يكون مخطط الجدولة الناتج كما في الشكل (9).

من الشكل (9)، نستنتج الآتي:

– في اللحظة الزمنية 2، تدخل المهمة الثانية (الخضراء) طور التنفيذ، التي دورها 3.5 وزمن تنفيذها 1.5، وتبقى في حيز التنفيذ حتى اللحظة الزمنية 2.8، إذ تدخل المهمة الأولى (غير الدورية) المنافسة على زمن المعالج، وبما أننا نستخدم خوارزمية RM، فإن المهمة ذات الدور الأقل تملك الأولوية العليا. بمقارنة الدور بين المهمتين الأولى والثانية، فإن المهمة الأولى التي دورها 3 لها الأولوية الأعلى، وبالتالي تتم مقاطعة المهمة الثانية وتبدأ المهمة الأولى بالتنفيذ، أي المهمة غير الدورية التي قمنا بتركيبها على مهمة دورية، وتستمر بالتنفيذ حتى اللحظة 4، والسبب في ذلك هو أن ميزانية المعالج المخصصة لتنفيذ المهمة غير الدورية تكون قد انتهت وعليها الانتظار حتى اللحظة 6 حتى تخصص لها ميزانية جديدة، وبذلك تكون هذه المهمة قد نفذت 1.2 من زمن تنفيذها الكلي ويبقى منه 0.5 حتى تنتهي من التنفيذ نهائياً، في هذه

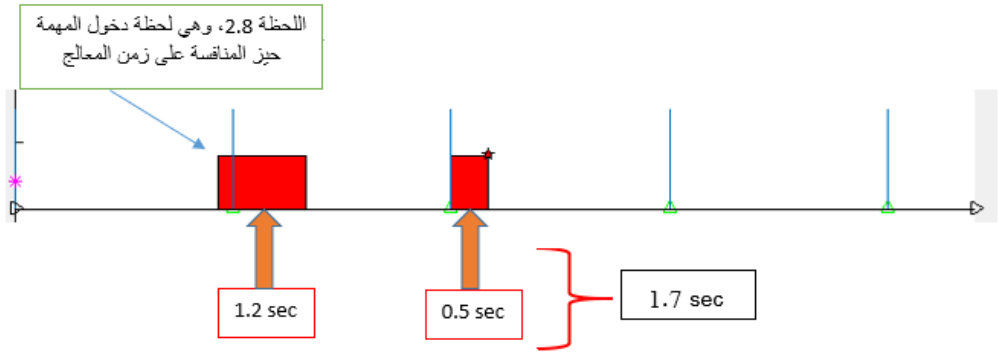
اللحظة تكمل المهمة الثانية التنفيذ (المهمة الثالثة لم تبدأ دوراً جديداً حتى اللحظة 6.5) وتستمر لمدة 0.7 ثانية، وبذلك تكون قد أتمت زمن تنفيذها الكامل: 1.5sec = 0.7 + 0.8.



الشكل (9): مخطط عملية الجدولة باستخدام خوارزمية Deferrable Server.

– يعود المعالج لحالة فراغ بسبب عدم جهوزية أية مهمة للتنفيذ، ويبقى بهذه الحالة حتى اللحظة 5.5، حتى يبدأ دور جديد للمهمة الثانية وتنفذ لمدة 0.5 ثانية، إذ في اللحظة 6 تكمل المهمة الأولى تنفيذها لمدة 0.5، وتتم مقاطعة المهمة الثانية لأنه في اللحظة 6 تكون المنافسة بين المهمة الأولى والثانية، وللمهمة الأولى الأولوية العليا لأن دورها أقل من المهمة الثانية (وفق خوارزمية RM).

– المهمة الأولى (الحمراء) هي المهمة غير الدورية، وزمن تنفيذها 1.7sec، كما هو مبين في الشكل (10).



الشكل (10): زمن تنفيذ المهمة غير الدورية بخوارزمية Deferrable Server مع المهمتين الدوريتين بخوارزمية RM.

6.5- مقارنة بين الخوارزميتين Deferrable Server و Pooling Server:

بالمقارنة مع الدراسة [1]، فإن خوارزمية Deferrable Server لجدولة المهام غير الدورية المستخدمة في نظامنا المقترح قد حققت نتائجاً أفضل من حيث المحافظة على ميزانية المعالج في حال كان رتل المهام غير الدورية فارغاً، وذلك بعكس خوارزمية Pooling Server التي تعمل على استنزاف الميزانية في حال عدم وجود مهمة غير دورية جاهزة للتنفيذ.

يبين الشكل (11) مخطط ميزانية المعالج للجدولة باستخدام خوارزمية Polling Server.

من الشكل (11) نستنتج الآتي:

أن الميزانية تملأ مع كل دور للعملية Polling Server وتستهلك مباشرة في حال عدم وجود مهام في الرتل غير الدوري، وكذلك عندما تدخل العملية غير الدورية مرحلة التنفيذ تبدأ الميزانية بالهبوط، ويمكن أن يتم استهلاكها بشكل كامل بمعدل 1، المهمة غير الدورية هي المهمة الثالثة (اللون الأصفر). يكون هذا النوع من الجدولة غير فعالاً من أجل المحافظة على الميزانية، إذ أنه يفرغ الميزانية ويستهلكها سواءً وجدت المهمة الدورية في الرتل أم لا.

يوضح الشكل (12) مخطط ميزانية المعالج للجدولة باستخدام خوارزمية Deferrable Server.

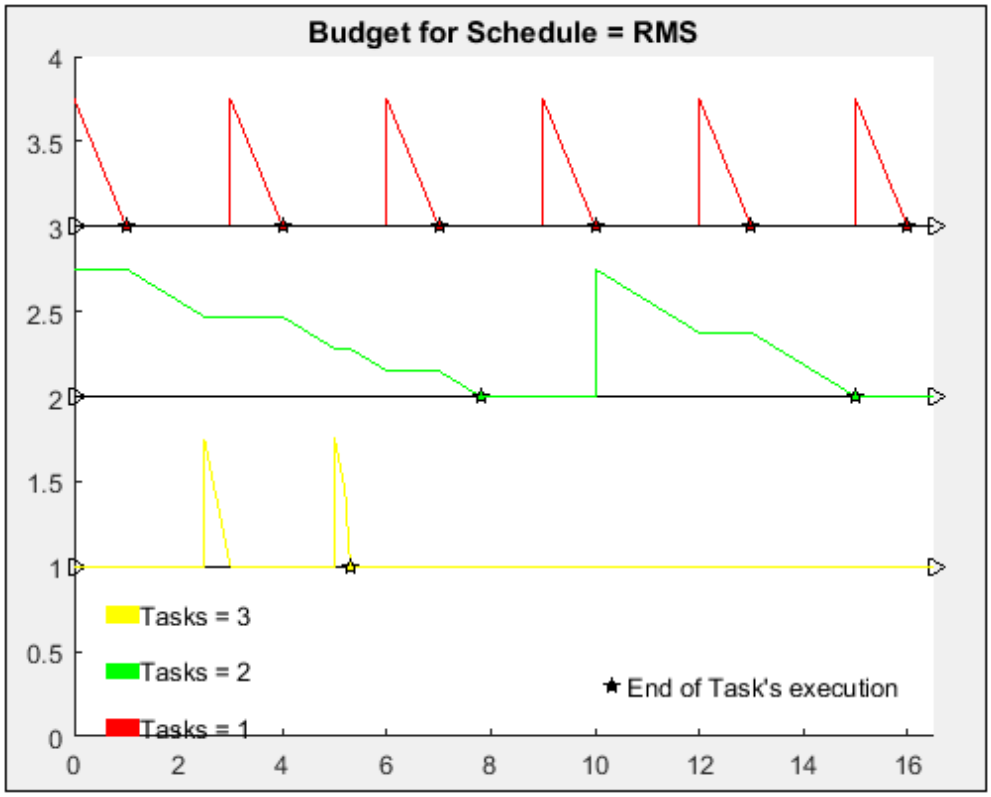
من الشكل (12)، نستنتج الآتي:

أنه لا يتم في هذه الحالة استهلاك الميزانية عندما لا يكون هناك أي عمل غير دوري جاهزاً للتنفيذ، أي أنه طالما لا توجد مهمة غير دورية قد نفذت بعد، فإن الميزانية لا تزال ممتلئة، لكن

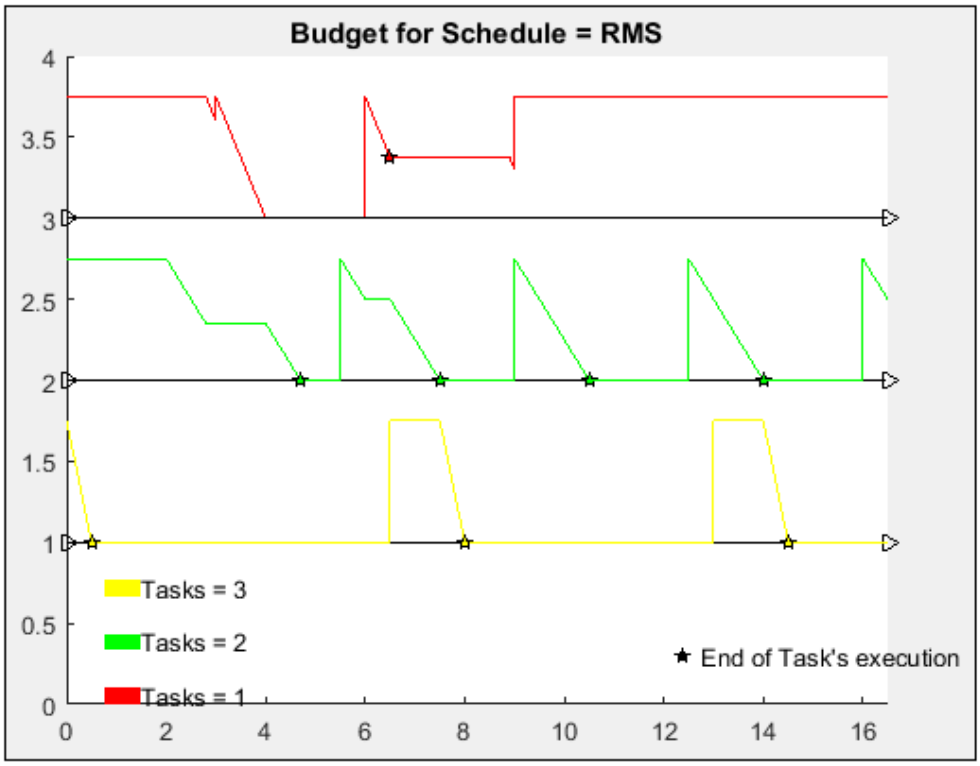
بمجرد المباشرة بتنفيذ مهمة تبدأ ميزانية زمن المعالج بالنقصان، وبالتالي نحافظ على الميزانية في حال عدم تنفيذ المهام غير الدورية، علماً أن المهمة غير الدورية هنا هي المهمة الأولى (اللون الأحمر).

بالنتيجة، نتوصل إلى أن خوارزمية Deferrable Server قد حققت نتائجاً أفضل من خوارزمية Polling Server، وذلك من حيث الحفاظ على ميزانية المعالج وعدم استنزافها.

يوضح الجدول (11) نتائج النظام المصمم المقترح مقارنة مع الدراسات السابقة.



الشكل (11): مخطط ميزانية المعالج للجدولة باستخدام خوارزمية Polling Server.



الشكل (12): مخطط ميزانية المعالج للجدولة باستخدام خوارزمية **Deferrable Server**.

البحث	الخوارزميات المستخدمة		ملاحظات
	مهام دورية	حشر مهام غير دورية	
[1]	DM EDF LST	Background Polling Server	حققت خوارزمية Polling Server أداءً أفضل من خوارزمية Background.
[5]	EDF RM	–	RM أفضل من EDF من حيث التعقيد الزمني.
[10]	RM DM	–	–

	EDF		
الدراسة الحالية	RR LST RM	Background	– استغلال الوقت الضائع في المعالج. – سهلة التطبيق.
		Polling Server	استنزاف ميزانية المعالج في حال عدم وجود مهمة غير دورية جاهزة للتنفيذ.
		Deferrable Server	حققت أفضل نتائج من حيث المحافظة على ميزانية المعالج وعدم استنزافها في حال كان رتل المهام غير الدورية فارغاً.

الجدول (11): نتائج النظام المصمم المقترح مقارنة مع الدراسات السابقة.

6- الاستنتاجات:

من النتائج التي حصلنا عليها، والتي حققت الغاية المرجوة من البحث، نستنتج الآتي:

– أهمية استخدام خوارزمية Background في جدولة المهام غير الدورية، لما لها من دور في استغلال وقت المعالج الضائع أثناء حشرها مع مهام دورية.

– تحافظ خوارزمية Deferrable Server على ميزانية المعالج في حال كان رتل المهام غير الدورية فارغاً، وذلك بعكس خوارزمية Pooling Server التي تعمل على استنزاف الميزانية في حال عدم وجود مهمة غير دورية جاهزة للتنفيذ.

7- الخاتمة والتوصيات:

قدم هذا البحث دراسة عملية لخوارزميات الجدولة بالنسبة للمهام الدورية مع إمكانية حشر مهام غير دورية بتقنيات مختلفة، وذلك من خلال تصميم نظام يقدم نمذجة عملية لجدولة تلك المهام لفهم آلية الجدولة بالتفصيل ومدى استغلال زمن المعالج.

يوصي البحث بمايلي:

- محاكاة أنواع أخرى من خوارزميات الجدولة، وخصوصاً فيما يتعلق بالعمليات غير الدورية.
- تطوير النظام المصمم المقترح ليتضمن خوارزميات تختار الطريقة الأفضل من بين خوارزميات الجدولة، وذلك من أجل عملية الجدولة.

8- المراجع:

[1]- د. م. محمد مصطفى حجازية، "نظم الزمن الحقيقي"، مطبوعات جامعة تشرين، 2013.

[2]- Arezou Mohammadi, Selim G. Akl, " Scheduling Algorithms for Real-Time Systems", Queen's University, No. 2005-499.

[3]- M. Kaladev, Dr. S. Sathiyabama, "A Comparative Study of Scheduling Algorithms for Real Time Task", International Journal of Advances in Science and Technology, Vol. 1, No. 4, 2010.

[4]- Yaashuwanth, C., R. Ramesh, "Web-Enabled Framework for Real-Time Scheduler Simulator (A Teaching Tool)", Second International Conference on,pp. 826-830. IEEE, 2010.

[5]- S. K. NAGER, N. S. GILL,"Comparative Study of RM and EDF Scheduling Algorithm in Real Time RTOS", IJCSMC, Vol. 6, Issue. 3, March 2017, pg.67 – 71.

[6]- F. F. Peronaglio, A. Manacero, R. S. Lobato, R. Spolon, "MODELING REAL-TIME SCHEDULERS FOR USE IN SIMULATIONS THROUGH A GRAPHICAL INTERFACE", Society for Modeling & Simulation International (SCS), 2017.

[7]- Zuber Patel, Upena Dalal, "DESIGN AND IMPLEMENTATION OF LOW LATENCY WEIGHTED ROUND ROBIN (LLWRR) SCHEDULING FOR HIGH SPEED NETWORKS", International Journal of Wireless & Mobile Networks (IJWMN) Vol. 6, No. 4, August 2014.

[8]- Myunggwon Hwang, Dongjin Choi, Pankoo Kim1, "Least Slack Time Rate First: an Efficient Scheduling Algorithm for Pervasive Computing Environment", Journal of Universal Computer Science, vol. 17, no. 6 (2011), 912-925.

[9]- Leena Das, Durga Prasad Mohapatra. Sourav Mohapatra, "Schedulability Analysis for Rate-Monotonic Algorithm in Parallel Real-Time Systems", International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, Number 16 (2017) pp. 5681-5689.

[10]- Jun Wu, Jian-Fu Li, "An Enhanced Real-Time Deferrable Server Scheduler for Xen Virtualization Systems", IAENG International Journal of Computer Science, 45:3, IJCS_45_3_05 (Advance online publication: 28 August 2018).

Modeling and developing a Parallel Processing System to Schedule, Insert and Synchronize an Aperiodic Tasks with a Periodic Tasks using Multi Algorithms

Dr. Eng. Ammar Ali Zakzouk

Assistant Professor in Automatic Control and Computers Engineering Department
Mechanical and Electrical Engineering Faculty
Albaath University

Abstract

This paper presents the modeling and development of a multi-tasking system to perform multiple tasks in parallel with priority degree, using periodic tasks and aperiodic tasks scheduling algorithms with different techniques, depending on the conditions and criteria that vary from one algorithm to another, in order to exploit the processor time.

The research was based on periodic tasks scheduling using the Round Robin algorithm, the Least Slack Time algorithm and the Rate Monotonic algorithm, with aperiodic tasks being inserted in these periodic tasks using the Background algorithm, the Pooling Server algorithm and the Deferrable Server algorithm.

The performance of the developed system was evaluated by testing the algorithms used and comparing the results obtained with the results of other researches. These results showed the effectiveness of this system in the maximal exploitation of the processor time and reduce the total processing time, by reaching the following:

- The importance of using the Background algorithm in the scheduling of aperiodic tasks, for its role in exploiting the lost time of the processor while they are inserted with periodic tasks.
- The Deferrable Server algorithm maintains the processor budget if the aperiodic tasks queue is empty, unlike the Pooling Server algorithm that drains the budget if there is no aperiodic task ready for execution.

Key-words: Scheduling Algorithms, Periodic Tasks, Aperiodic Tasks, Processor Utilization, Real Time Systems.