مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (40) العدد (4) 2018
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (40) No. (4) 2018

مقارنة أداء خوارزميات جدولة المهام العشوائية في نظم الزمن الحقيقي

الدكتور محمد حجازية*

خلدون فاقي**

(تاريخ الإيداع 24 / 6 / 2018. قُبِل للنشر في 30/ 7 / 2018)

□ ملخّص □

تشكل نظم الزمن الحقيقي اليوم النواة الأساسية لمعظم التطبيقات المستخدمة في مجالات تقنية المعلومات والاتصالات، كما أن سرعة تطور مثل هذه النظم جذب اهتمام الباحثين من أجل تحقيق أداء أمثل والتخلص قدر الإمكان من المشاكل والمساوئ التي تعاني منها بغية تحسين أدائها بما يتناسب مع حجم المهام الموكلة إليها.

توجد العديد من التحديات الأساسية التي تواجه نظم الزمن الحقيقي والتي تتجسد بالدرجة الأولى في مشكلة جدولة تنفيذ المهام على نوى المعالج في هيكلية المعالجات متعددة النوى حيث تم اقتراح العديد من الطرق منها ما اعتمد الطريقة العامة والتي تكون فيها أي مهمة قابلة للتنفيذ على أية نواة، أو الطريقة المجزأة التي تعتمد على تخصيص نواة معينة لكل مجموعة محددة من المهام، وهناك أيضاً الطريقة شبه المجزأة وهي عبارة عن هجين من الطريقتين السابقتين حيث يتم تخصيص مجموعة من المهام لكي تنفذ على نواة معينة في حين يسمح لمهام أخرى بالتنفيذ على أية نواة من نوى المعالج.

تم في هذا البحث مقارنة أداء خوارزميات جدولة المهام العشوائية على منصة متعددة النوىبهدف تحديد الخوارزمية الأفضل من ناحية مجموعة من البارامترات المعتمدة من قبل الباحثين في هذا المجال والتي بدورها تعطينا تفاصيل دقيقة حول جودة مثل هذه الخوارزميات عند تطبيقها على مجموعة من المهام العشوائية المولدة وفق التوزع الاحتمالي اللوغاريتمي الموحد.

تمت عملية المحاكاة على البرنامج simso والذي أثبت موثوقية أداء عالية بشهادة العديد من الباحثين في هذا المجال فضلاً عن كونه يقدم إمكانية توليد المهام وفق توزعات احتمالية معينة، ويحاكي تفاصيل دقيقة متعلقة بخصائص المهام العشوائية.

الكلمات المفتاحية: الجدولة، المهام العشوائية، المعالج متعدد النوى، التوزع الاحتمالي.

* أستاذ مساعد – قسم هندسة الحاسبات والتحكم الآلي– كلية الهندسة الميكانيكية والكهربائية– جامعة تشرين– اللاذقية– سورية.

الإيميل: mohammed.hejazieh2016@gmail.com

**طالب دراسات عليا (دكتوراه) – قسم هندسة الحاسبات والتحكم الآلي– كلية الهندسة الميكانيكية والكهربائية– جامعة تشرين– اللاذقية– سورية. الإيميل: khaldon.faqi@gmail.com

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (40) العدد (4) 2018
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (40) No. (4) 2018

Comparison of the performance of sporadic tasks scheduling algorithms in real time systems

Dr. Mohammed Hijazieh*
Khaldon Faqi **

(Received 24 / 6 / 2018. Accepted 30 / 7 / 2018)

□ ABSTRACT □

Nowadays, Real-time systems are considered as the core of most applications that used in Telecommunication and information technology areas. The rapid development of such systems has attracted researchers' attention to optimize performance and eliminate problems and disadvantagesas possible in order to improve their performance in proportion to the volume of tasks assigned to them.

There are many challenges facing real-time systems, mainly the problem of task scheduling on processor cores in the multi-core processor architecture. Several schemes have been proposed, including the global scheme, where any task can be executed on any core ,The partitionedscheme which depends on the allocation of a specific core for each set of tasks. There is also the semi-partitionedscheme, which is a hybrid of the two previous schemes, where a set of tasks is assigned to execute on a specific core while other tasks are allowed to be executed on any core of processor.

In this paper, we compare the performance of sporadic tasks scheduler algorithms on a multi-core platform in order to determine the best algorithm in terms of a set of parameters adopted by researchers in this field, which in turn gives us accurate details about the quality of such algorithms when applied to a set of sporadic tasks generated according to uniformed Logarithmic probability distribution.

The simulation is done using Simso simulator, which proved the reliability of high performance by the testimony of many researchers in this field, as it provides the possibility of generating tasks according to specific probability distributions, and simulates accurate details related to the characteristics of random tasks

Key Words: Scheduling, Sporadic Tasks, Multicore Processor, Probability Distribution.

***Associate Professor, Department of computer and automatic control Engineering, Faculty of Mechanical and electrical Engineering, Tishreen University, Lattakia, Syria.**
Email:mohammed.hejazieh2016@gmail.com
****Postgraduate Student , Department of computer and automatic control Engineering, Faculty of Mechanical and electrical Engineering, Tishreen University, Lattakia, Syria.**
Email:khaldon.faqi@gmail.com

مقدمة:

يُنفذ أي نظام يعمل بالزمن الحقيقي مجموعة من المهام، وتُعرّف المهمة بأنها وحدة التنفيذ الأساسية في البرنامج والتي ينتج عن تنفيذها إعطاء نتيجة معينة وتشكل خدمة أساسية من الخدمات التي يقدمها أي نظام أو تطبيق يعمل بالزمن الحقيقي.

تشكل المهام العشوائية Sporadic Tasks جزءاً مهماً من المهام في نظم التحكم البرمجية وهي تتواجد بكثرة في معظم أنظمة الزمن الحقيقي مثل الإنذار بحدوث حريق في منشأة معينة، ولكن المشكلة الأساسية عادة تكمن في سرعة الاستجابة لمثل هذه المهام عند ورودها، نظراً لكونها ترد بشكل عشوائي دون فواصل زمنية ثابتة أو موعد مسبق، لكن في بعض الأحيان يمكن التنبؤ بشكل تقريبي بموعد حدوث هذه المهام.

أهمية البحث وأهدافه:

الهدف الرئيسي من هذا البحث هو اختبار مدى قابلية جدولة المهام العشوائية وذلك عند تطبيق مجموعة من الخوارزميات المستخدمة في الجدولة على نظام تشغيل يعمل بالزمن الحقيقي والذي بدوره يتألف من مجموعة من المهام الدورية Periodic Tasks والمهام العشوائية Sporadic Tasks، وذلك من أجل الوصول إلى الخوارزميات القادرة على الاستجابة للمهام العشوائية عند ورودها وتنفيذها بالطريقة التي تضمن عدم تجاوز القيد الزمني Deadline المرتبط بهذا النوع من المهام.

تم استخدام المحاكي SIMSO لهذا الغرض، وهو يعتبر من المحاكيات الفعالة بشهادة العديد من الباحثين في هذا المجال كونه يحاكي تفاصيل دقيقة تتعلق بالمهام العشوائية مثل إمكانية التنبؤ بأزمنة الورود أو إمكانية إخضاع هذه الأزمنة لتوزع احتمالي معين.

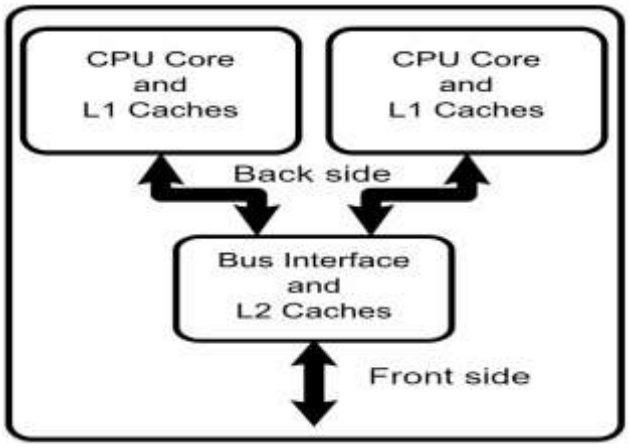
طرائق البحث ومواده:

نقدم في ما يلي نبذة مختصرة عن طبيعة نظام الزمن الحقيقي المعمول به من حيث خصائص المهام الموجودة فيه بالإضافة إلى الخصائص المتعلقة بالعتاد الصلب للنظام من معالج وذاكرة وغير ذلك.

1. معمارية المعالج Processor Architecture:

المعالج المستخدم في هذا النظام هو معالج متعدد النوى Multicore Processor، حيث تعتبر هذه المعمارية من النماذج الحديثة للمعالجات والتي تم التوجه لها بعد ظهور مشكلة عدم القدرة على زيادة تردد المعالج أحادي النواة إلى قيمة عالية تمكننا من الحصول على أداء عالٍ وفعال، حيث أن كل زيادة فوق التردد الأعلى الذي تم التوصل إليه تنتج عنها مشاكل إضافية تتعلق بظهور عدد من السعات الطفيلية التي تنشأ عادة في الدارات الالكترونية عند الترددات العالية، كما أن تحقيق مزامنة فعالة بين العناصر والبوابات في دارة المعالج بات أمراً صعباً بسبب التأخير الزمني الناتج عن عمل بعض البوابات المنطقية في المعالج.[8][4]

ويبين الشكل (1) معمارية المعالج متعدد النوى، حيث نلاحظ كيفية توضع النوى بالإضافة إلى ذاكرة الكاش بمستوياتها المختلفة.



الشكل (1) يبين معمارية المعالج متعدد النوى

2. نموذج المهام العشوائية Sporadic Tasks Model:

إن المهام العشوائية Sporadic Tasks المستخدمة في هذا البحث تتبع للنموذج التالي حيث نوصّف كل مهمة بمجموعة من البارامترات المميزة لها:[11][7]

1. زمن ورود المهمة t_A: ويوصف بأنه الزمن الذي يتم عنده رفع إشعار أو إعلام بورود حدث جديد في نظام الزمن الحقيقي، ويشترط عدم ورود حدثين في نفس اللحظة في هذا النوع من المهام، أي:

$$\forall A, B \in E \Rightarrow t_A \neq t_B \qquad (1)$$

2. الزمن الفاصل الأصغري (ε) بين كل ورودين متتالين لحدث واحد أو حدثين مختلفين وهو مقدار موجب دوماً، أي:

$$\forall A_1, A_2, B \in E \Rightarrow \left|t_{A_1} - t_{A_2}\right| \geq \varepsilon \ \ and \left|t_{A_1} - t_B\right| \geq \varepsilon \qquad (2)$$

3. زمن التنفيذ Execution Time: وهو زمن إشغال المعالج من قبل المهمة لكي تنفذ.
4. القيد الزمني Deadline: ويوصف بأنه الزمن الذي يتوجب على المهمة إنهاء التنفيذ قبل بلوغه.

4. برنامج المحاكاة المستخدم:

تم استخدام المحاكي SIMSOفي هذا البحث لكونه يتمتع بالخصائص التالية:

1. مفتوح المصدر Open Source: أي يمكن تعديل أي ملف من المكتبات المضمنة في هذا المحاكي بما في ذلك الملفات التي تصف عمل خوارزميات الجدولة.[17][6]
2. الواجهة الرسومية التفاعلية(Graphical User Interface) GUI: والتي تؤمن مجموعة من القوائم والخيارات التي تتيح ضبط إعدادات المحاكاة بشكل سريع وفعال.
3. يدعم إدخال قيم لبارامترات مفصلة ضمن خيارات المحاكاة، ونذكر من هذه البارامترات:

a. عدد دورات الساعة ضمن التعليمة الواحدة (Clock Per Instruction) CPI: والمقصود بذلك عدد دورات الساعة اللازمة لتنفيذ كل تعليمة من التعليمات التي تتضمنها المهمة.

b. عدد التعليمات Instructions: عدد التعليمات التي تتضمنها المهمة، حيث أن المهمة ممكن أن تتضمن أكثر من تعليمة في بعض الأحيان.

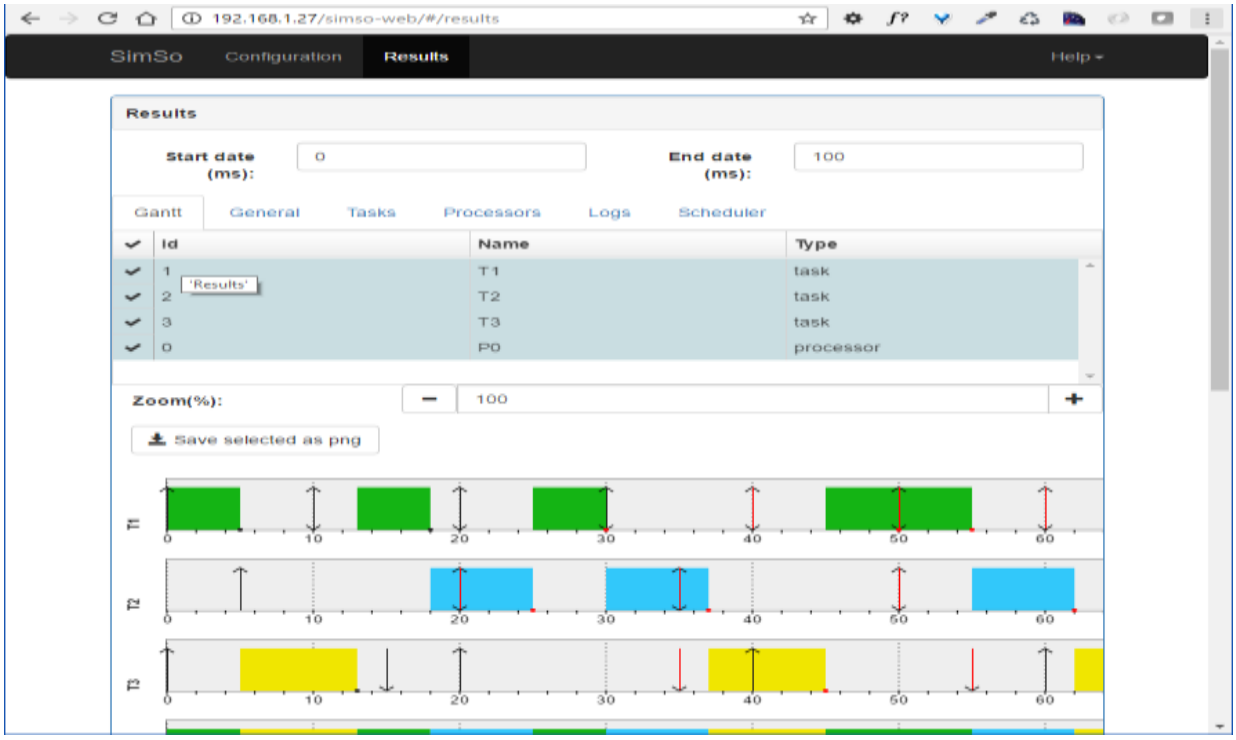
c. نسبة الوصول إلى الذاكرة Memory Access Rate: تمثل النسبة بين عدد التعليمات التي تتطلب الوصول إلى الذاكرة بالنسبة إلى عدد التعليمات الكلي، ويسمى بـ (MIX).[20][23]

d. مظهر مسافة الكدسة(Stack Distance Profile) SDP: ويتضمن توزع محتويات الكاش المختلفة والتي تسمى بـ (Cache Lines) ضمن ذاكرة الكاش وتقاس عن طريق عدد الـ Cache Lines المنفردة Unique التي تفصل بين وصولين متتاليين لنفس الـ Cache Line.

4. مكتوب بلغة الـ Python والتي تعد من اللغات البرمجية المتطورة، كما أنها تدعم البرمجة غرضية التوجه OOP (Object Oriented Programming).

5. كما أنه يدعم واجهة رسومية تفاعلية على الانترنت تتيح للمستخدم استخدام المحاكي دون الحاجة لتثبيت البرنامج على نظام التشغيل.

ويبين الشكل (3) الواجهة الرسومية التفاعلية على الانترنت.



الشكل (3) يبين الواجهة الرسومية للمحاكي SIMSO

النتائج والمناقشة:

تمت دراسة ثلاثة سيناريوهات مختلفة تتضمن العمل وفق ثلاث خوارزميات هي:

1. خوارزمية (Pseudo Deadline) PD2.
2. خوارزمية (Earliest Deadline First) EDF.
3. خوارزمية (Largest Local Remaining Execution time First) LLREF.

يتم في السيناريو الأول تنفيذ المحاكاة على نواتين، وفي السيناريو الثاني على أربع نوى، وفي السيناريو الثالث على ثمان نوى. مع الأخذ بعين الاعتبار البارامترات التالية:

1. كل نواة تحتوي على ذاكرةL1 كاش بحجم 4 kb وزمن وصول 1 cycle وكلفة ضياع 9 cycle.
2. L2 كاش مشتركة بين جميع الأنوية بحجم 64 kb وزمن وصول 10 cycle وكلفة ضياع 90 cycle.
3. كلفة الوصول إلى الـ RAM هي (100 cycles)، باعتبار أن (1 cycle = 1 nano second).
4. كلفة تبديل السياق (100 cycles).

كما يبين الجدول (2) مجموعة المهام العشوائية Sporadic Tasks مع قيم البارامترات الخاصة بها.

الجدول (2) يبين المهام العشوائية

المهمة Task	زمن التنفيذ WCET (ms)	أزمنة التفعيل Activation Dates(ms)	القيد الزمني Deadline (ms)	عدد التعليمات Instructions Count	نسبة التعليمات الذاكرية MIX
T1	2	5, 25, 40, 55	10	225000	0.3
T2	3	10, 30, 45, 65	12	260000	0.35
T3	3.6	20, 35, 60, 75	13	320000	0.4
T4	4	10, 40, 60, 80	15	350000	0.5

تمت مقارنة النتائج بالاعتماد على البارامترات التالية:

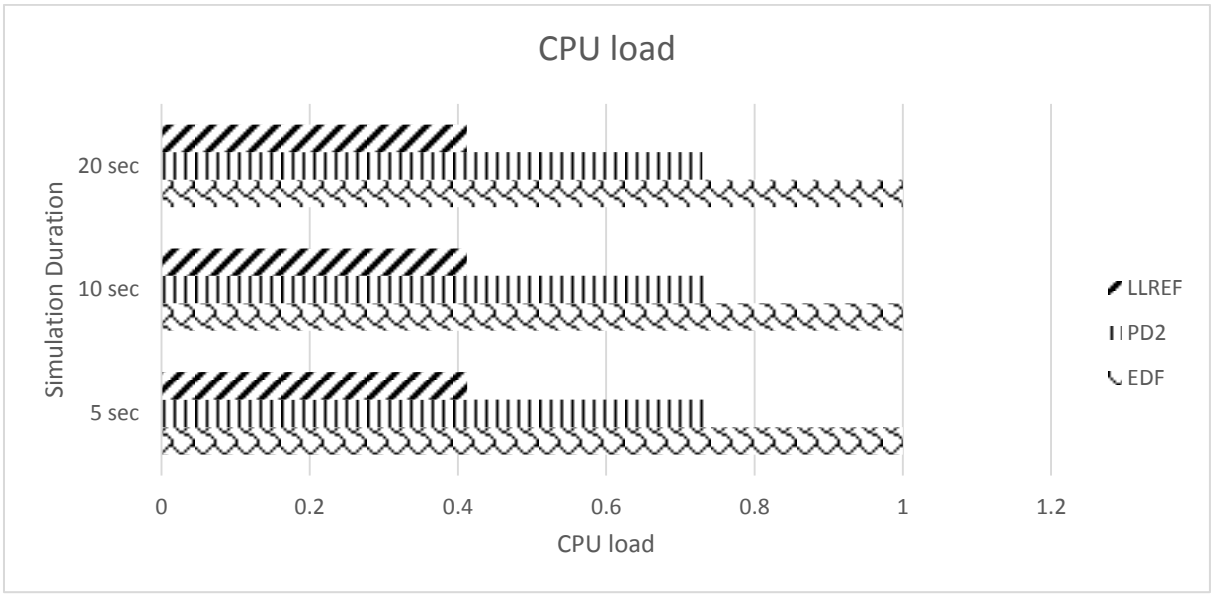
1. حمل المعالج CPU Load: ويمثل المقدار الزمني الذي يكون فيه المعالج مشغول بالتنفيذ بالنسبة إلى الزمن الكلي للمحاكاة.

2. أعباء عمليات تبديل السياق Context Switching Overheads: وتتضمن الزمن المنقضي ريثما يتم تبديل مهمة تنفذ حالياً بمهمة أخرى ذات أفضلية أعلى من المهمة الحالية وما يترتب على ذلك من حفظ حالة المهمة المستبعدة وتحميل مسجلات المعالج بالقيم الجديدة للمهمة الأخرى.

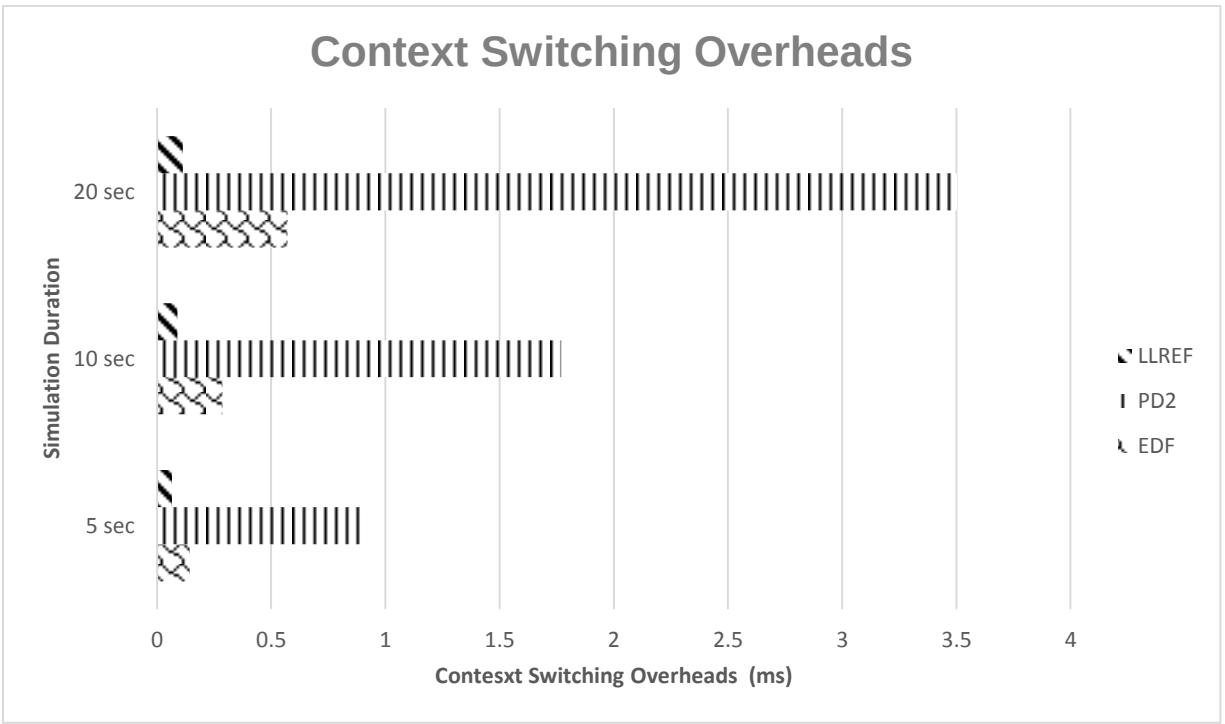
3. أعباء الجدولة Scheduling Overheads: وتتضمن الزمن المستهلك من قبل المعالج عند ورود حدث جديد يستدعي نداء المجدول لكي يقوم باتخاذ قرار الجدولة وما يترتب على ذلك من تنفيذ العالج لبعض العمليات حتى يتمكن من اتخاذ قرار الجدولة.

1. السيناريو الأول:

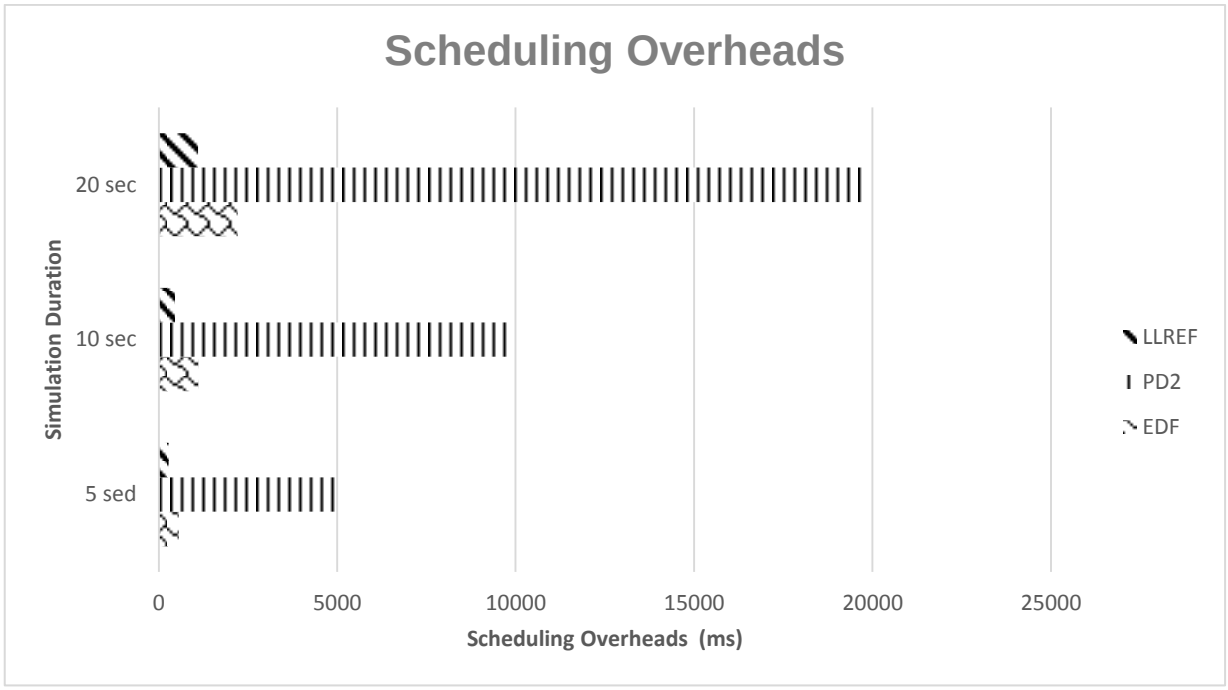
تم في هذا السيناريو محاكاة تطبيق خوارزميات الجدولة الثلاثة المذكورة سابقاً على مجموعة من المهام العشوائيةوتبيناالأشكال (4)، (5)، (6) المقارنة بين خوارزميات الجدولة الثلاثة من ناحية حمل المعالج، وأعباء عمليات تبديل السياق، وأعباء الجدولة، وذلك عند العمل على نواتين.



الشكل (4) المقارنة من حيث حمل المعالج.



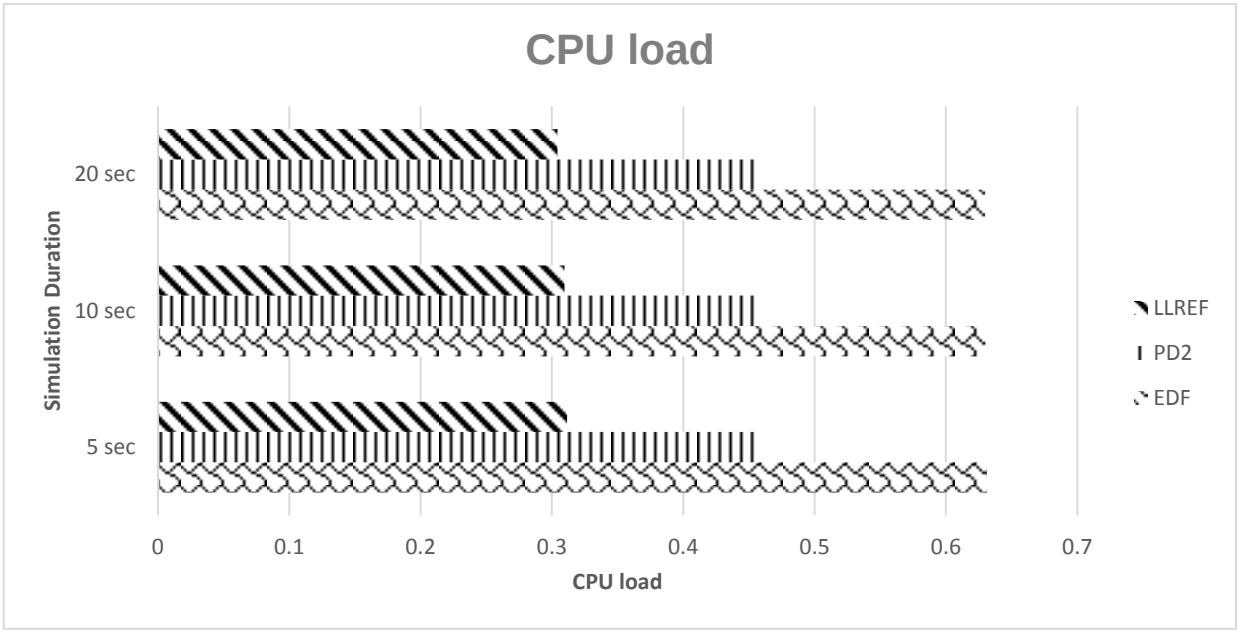
الشكل (5) يبين المقارنة من حيث أعباء تبديل السياق



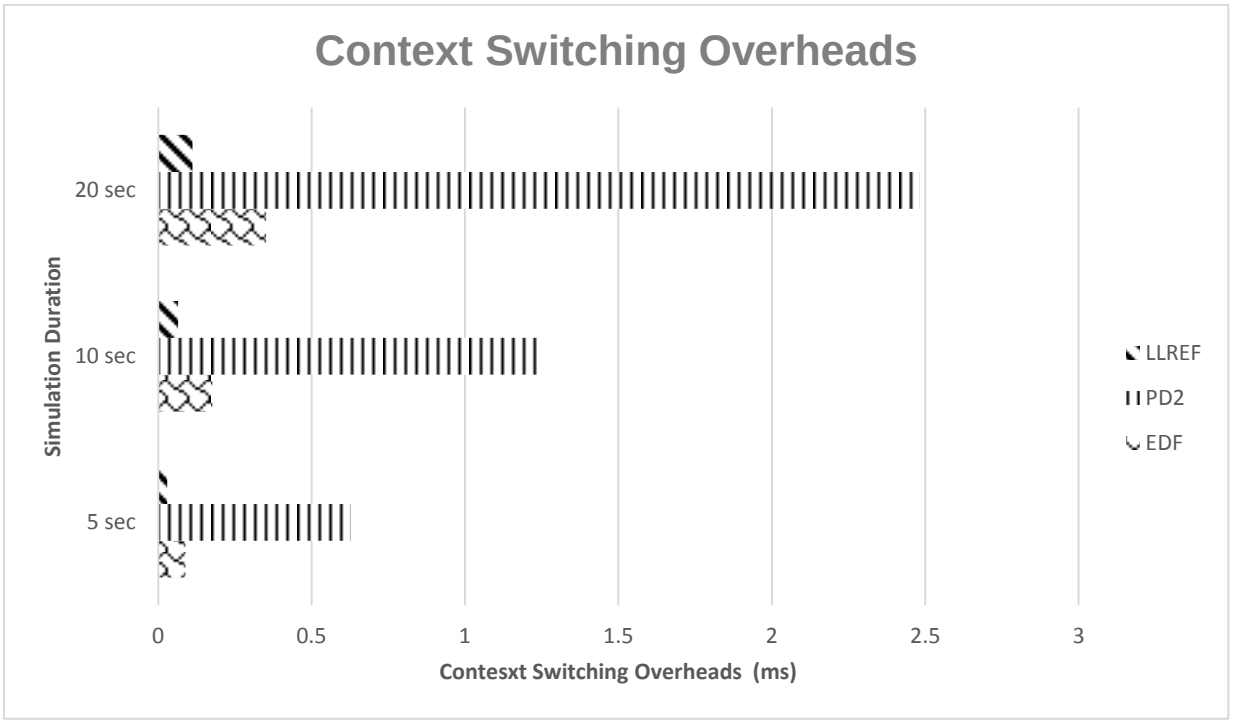
الشكل (6) يبين المقارنة بين أعباء الجدولة

2. السيناريو الثاني:

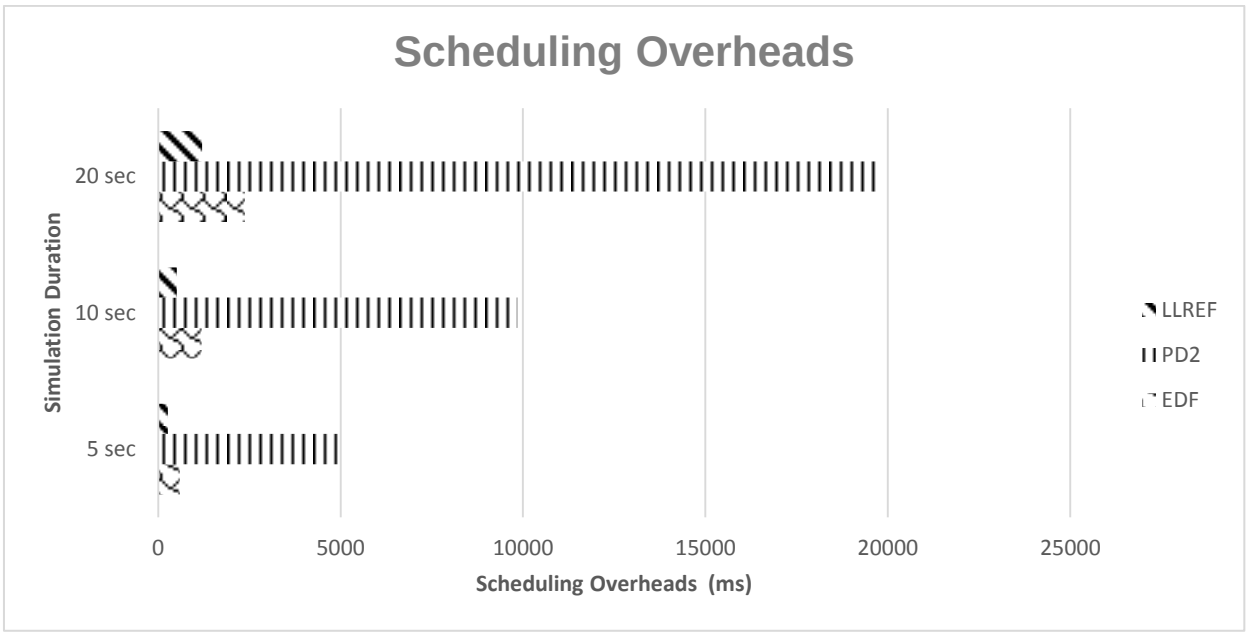
تم في السيناريو الثاني اتباع الخطوات نفسها التي تمت في السيناريو الأول لكن مع وجود أربع نوى، وتبين الأشكال (7)، (8)، (9) المقارنة بين خوارزميات الجدولة الثلاثة من ناحية حمل المعالج، وأعباء عمليات تبديل السياق، وأعباء الجدولة، وذلك عند العمل على أربع نوى.



الشكل (7) يبين المقارنة من حيث حمل المعالج



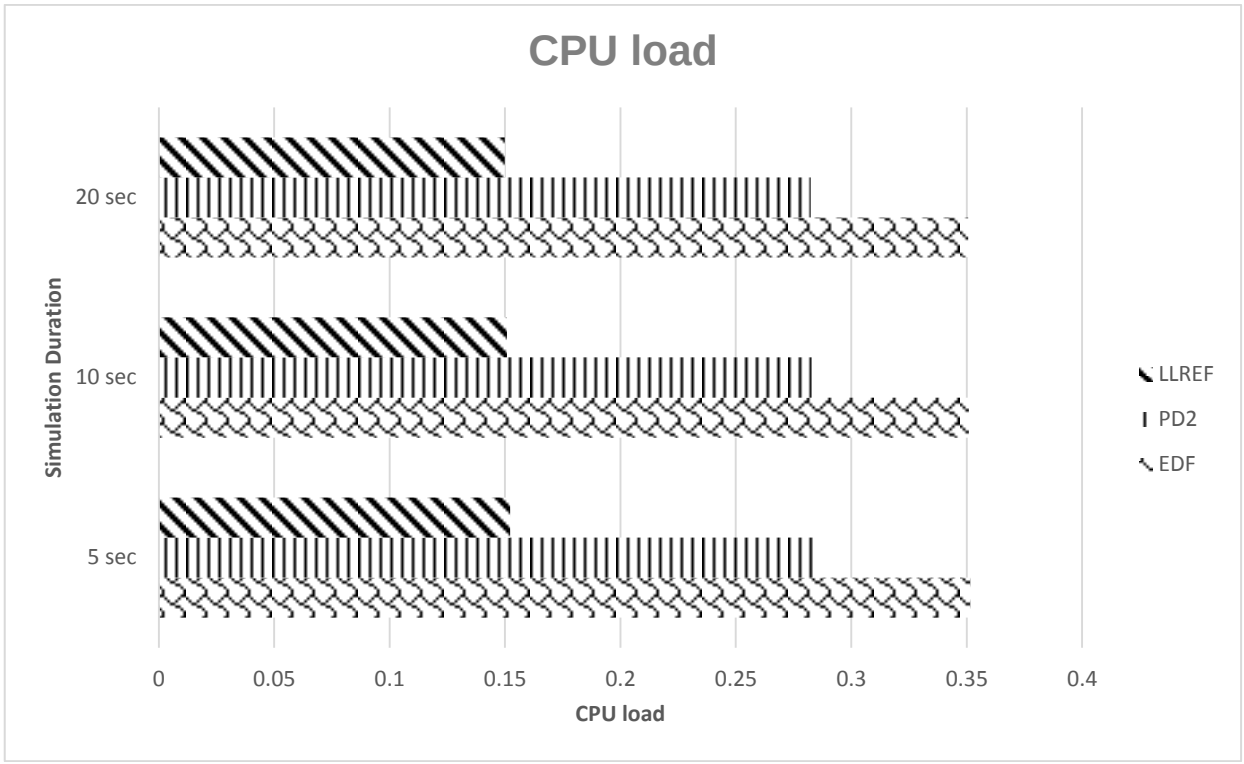
الشكل (8) يبين المقارنة من حيث أعباء تبديل السياق



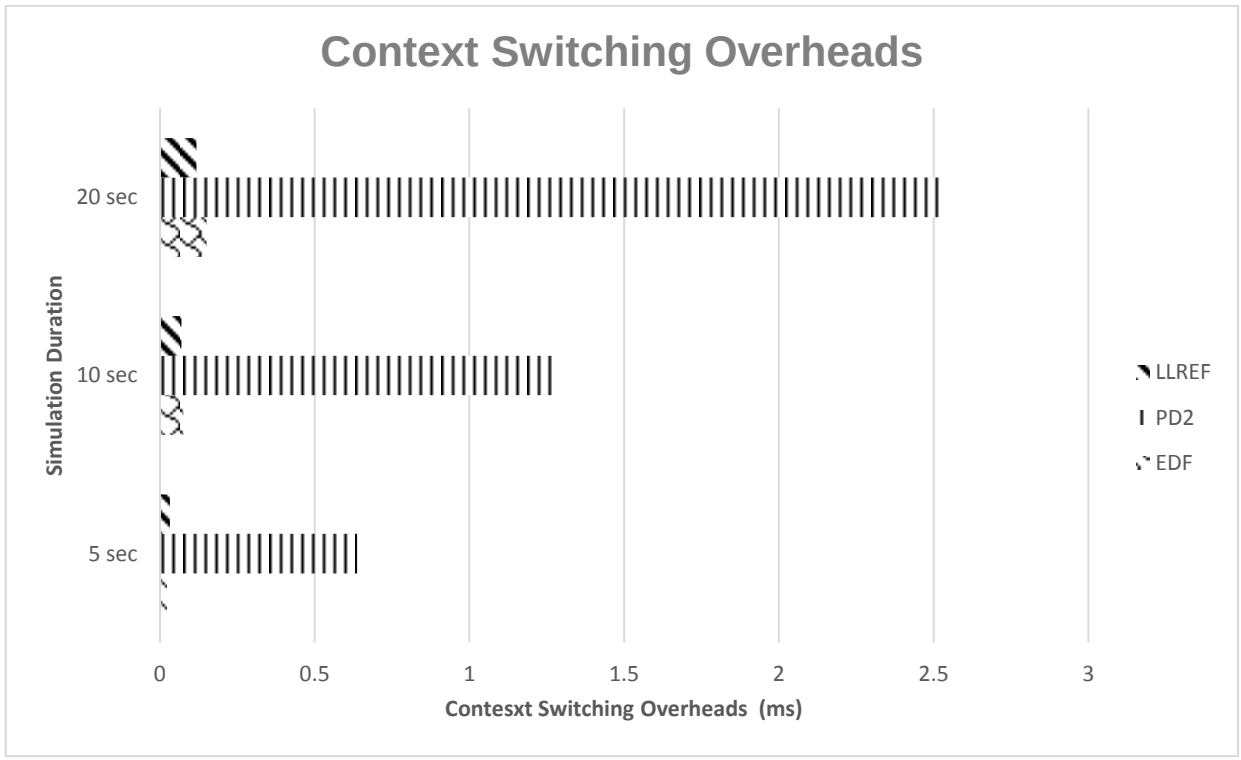
الشكل (9) يبين المقارنة من حيث أعباء الجدولة

3. السيناريو الثالث:

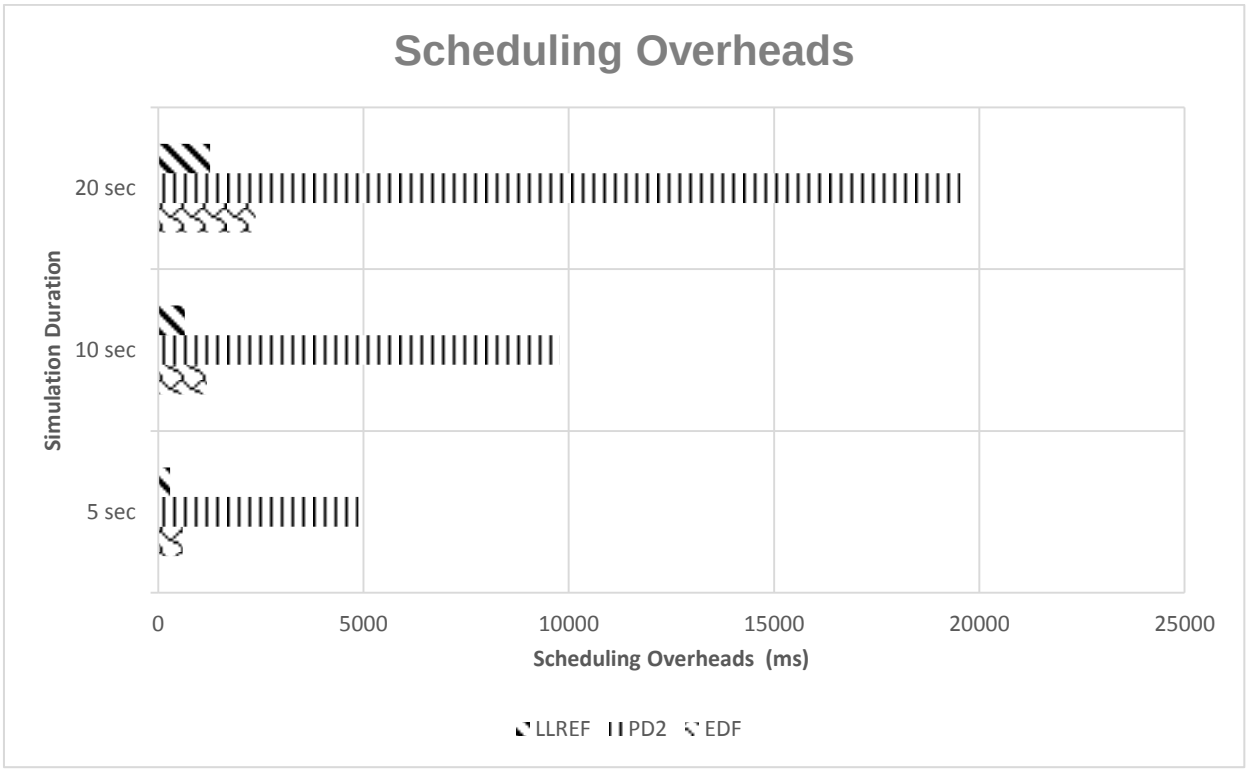
تم في السيناريو الثالث اتباع الخطوات نفسها التي تمت في السيناريو الأول لكن مع وجود ثمان نوى، وتبين الأشكال (10)، (11)، (12) المقارنة بين خوارزميات الجدولة الثلاثة من ناحية حمل المعالج، وأعباء عمليات تبديل السياق، وأعباء الجدولة، وذلك عند العمل على ثمان نوى.



الشكل (10) يبين المقارنة من حيث حمل المعالج



الشكل (11) يبين المقارنة من حيث أعباء الجدولة



الشكل (12) يبين المقارنة من حيث أعباء الجدولة

الاستنتاجات والتوصيات:

الاستنتاجات:

نستنتج من خلال النظر إلى النتائج في السيناريوهات الثلاث المدروسة أن:

1. خوارزمية LLREFتعطينا أداء أفضل من باقي الخوارزميات من ناحية حمل المعالج حيث أن معدل الحمل الوسطي لكل نواة ينقص مع زيادة عدد النوى بشكل كبير مقارنة مع النقصان الحاصل في باقي الخوارزميات.

2. أعباء العمليات الناتجة عن تبديل السياق تكون صغيرة في خوارزمية LLREF مقارنة مع الخوارزميات الأخرى بينما تكون هذه الأعباء أكبر في خوارزمية PD2 كون أن هذه الخوارزمية مقادة بالفترات الزمنية على خلاف الخوارزمية LLREF والتي هي مقادة بالأحداث.

3. الأعباء الناتجة عن الجدولة تزداد في خوارزمية LLREF بشكل طفيف مع زيادة زمن المحاكاة، بينما هذه الزيادة تكون ملحوظة أكثر في باقي الخوارزميات ويرجع ذلك إلى كثرة عدد مرات استدعاء المجدول لاتخاذ قرار الجدولة في الخوارزميات الأخرى.

4. تتمتع خوارزمية LLREF بخصائص أفضل من الخوارزميات الأخرى عند المقارنة مع بعين الاعتبار البارامترات الثلاثة المذكورة سابقاً ويرجع ذلك إلى الأداء الديناميكي لهذه الخوارزمية مع مختلف أنواع المهام فضلاً عن قلة الأحداث التي يتم إطلاقها عند جدولة مجموعة عشوائية مختارة من المهام قياساً مع غيرها من الخوارزميات.

التوصيات:

حسب نتائج البحث ووفقاًللسيناريوهات المدروسة يمكن التوسع في الدراسة من خلال زيادة عدد النوى المدروسة أو من خلال زيادة أزمنة المحاكاة، كما يمكن وضع مقارنة ودراسة الخوارزمية التي حققت أفضل النتائج في هذه الدراسة مع خوارزميات أخرى لم يتطرق لها البحث بغية التعرف على تفاصيل أكثر حول أداء هذه الخوارزمية والعمل على تلافي العيوب التي قد تنشأ من الدراسة التجريبية إن وجدت.

المراجع:

1. RADHAKRISHNA NAIK at el, "*Periodic and Aperiodic Real -Time Task Scheduling AlgorithmsSimulator",* International Journal of Pure and Applied Mathematics, Volume 118 No. 20 , 2018, 2681-2687 .
2. HYEONGBOO BAEK, "*Real-Time Scheduling for PreventingInformation Leakage with PreemptionOverheads* ", Advances in Electrical and Computer Engineering, Volume 17, Number 2, 2017.
3. ANKUR JAIN , **"** *Multishare Task Scheduling Algorithm For Real Time Micro-controller Based Application* **" ,** Mechatronics and Applications: An International Journal (MECHATROJ), Vol. 1, No.1, 2015.
4. SCHOEBERL, M., PEZZAROSSA, "*A multicore processor for time-critical applications*", Technical University of Denmark , 2018.
5. CHARU RANI, MRS. MANJUGODARA , **"** *Real Time System Scheduling Algorithms & Fault Tolerance* **"** , International Journal of Advanced Research in Computer and Communication Engineering , Vol. 4, Issue 7, July 2015.
6. SRI RAJ PRADHAN, SITAL SHARMA, DEBANJANKONAR, **"***A Comparative Study on Dynamic Scheduling of Real-Time Tasks in Multiprocessor System using Genetic Algorithms "* , International Journal of Computer Applications , Volume 120 – No.20, June 2015.
7. CHRISTINE ROCHANGE, **"***Parallel Real-Time Tasks, as Viewed by WCETAnalysis and Task Scheduling Approaches* **"** , University of Toulouse, Toulouse, France 2016.
8. FERNANDA F. PERONAGLIO AND ALEARDOMANACERO, "*Modeling Real-Time Schedulers For Use In Simulations Through A Graphical Interface***",** Dept of Computer Science and Statistics São Paulo State University – UNESP, 2017.
9. KARTHIK S. LAKSHMANAN, **"** *Scheduling and synchronization for Multicore Real Time Systems* **",** Carnegie Mellon University, Pittsburgh, PA, 2011.
10. MANAR QAMHEIH, **"***Scheduling of Parallel Real-time DAG Tasks on Multiprocessor Systems",* University of Paris-est 2015.
11. DAN MCNULTY, LENA OLSON, MARKUS PELOQUIN , "*A Comparison of Scheduling Algorithms for Multiprocessors"* , University of Maryland 2010.
12. GREG LEVIN , "*DP-FAIR: A Simple Model for Understanding Optimal Multiprocessor Scheduling"* International Journal of Computer Applications (0975 – 8887) Volume 110 – No. 6 , 2009.
13. KUSHALANJARIA, ARUN MISHRA, "*Thread scheduling using ant colony optimization: An intelligent scheduling approach towards minimal information leakage"*, Department of Computer Science & Engineering, DIAT, Pune, India, 2017.

14. GOKULSIDARTHTHIRUNAVUKKARASU*, RAGIL KRISHNA, "*Scheduling Algorithm for Real-Time Embedded Control Systems Using Arduino Board*", Deakin University, School of Engineering, Waurn ponds, Australia, 2017.

15. AMJAD MAHMOOD , SALMAN A. KHAN, "*Energy-Aware Real-Time Task Scheduling inMultiprocessor Systems Using a HybridGenetic Algorithm*", University of Bahrain, 2017.

16. LINLINTANG*A*, KAIQIANG MA, ZUOHUA LI, "*A New Scheduling Algorithm Based on Ant ColonyAlgorithm and Cloud Load Balancing*", Harbin Institute of Technology Shenzhen Graduate School Shenzhen, China, 2016.

17. JAYARAMMUDIGONDA, "*Fast Switching of Threads Between Cores*", ACM SIGOPS Operating Systems Review special issue, 2009.

18. SISU XI, MENG XU, CHENYANG LU,' *Real-Time Multi-Core Virtual Machine Scheduling in Xen*', Washington University, 2013.

19. MENG XU LINH T.X. PHANINSUP LEE OLEG SOKOLSKY, "*Cache-Aware Compositional Analysis of Real-Time MulticoreVirtualization Platforms*", *University of Pennsylvania*, 2013.

20. VIJAY SHREESHINDE, "*Comparison of Real Time Task Scheduling Algorithms"*, Terna Engineering College, Navi Mumbai, India, 2017.

21. GIRISH S. THAKARE, DR. PRASHANT. R. DESHMUKH, "*Performance Analysis of Real Time TaskScheduling Algorithm"*, Maharashtra, India 2017.

22. ANKUR JAIN, "*Multishape Task Scheduling Algorithm For Real Time Micro-Controller Based Application*", Department of Computer Engineering, ABV-IIITM, Gwalior 2017.

23. JALIL BOUKHOBZA, "*Cache Memory Aware Priority Assignment and Scheduling Simulation of Real-Time Embedded Systems*", UNIVERSITÉ DE BRETAGNE OCCIDENTALE, 2017.