# تحسين أداء خوارزمية التقطير في انترنت الأشياء

طالبة الدكتوراه: منال سعد الله العمر

كلية الهندسة المعلوماتية – جامعة البعث

اشراف الدكتور: أكرم المرعي + د. محسن عبود

**الملخص**

اكتسبت خوارزمية Trickle التي تم تقديمها وتوحيدها في RFC 6206 شعبية كبيرة حيث يمكنها أن تضمن حل تناقض البيانات في البيئات الموزعة بدقة وسرعة مع انخفاض تكلفة الصيانة ودعم جيد للتوسع وزيادة كثافة الشبكة. بالنسبة لهذه الميزات الجذابة تشكل Trickle أساس العديد من معايير الإنترنت ويتم نشرها في العديد من التطبيقات مثل البث الموثوق واكتشاف الخدمة والموارد الموزعة وكذلك بروتوكولات التوجيه ومنها بروتوكول RPL المستخدم في شبكات انترنت الأشياء. قمنا في هذا البحث بتقديم خوارزمية تقطير جديدة (New Sys-Trickle) قادرة على تقليل كل من زمن التقارب و عدد حزم التحكم المرسلة في الشبكة و مقدار استهلاك الطاقة ومقدار استهلاك المعالج في آن واحد, وهذا ما يميزها عن الخوارزميات السابقة التي كانت تحسن معيار على حساب معيار آخر. حيث قمنا بدمج ثلاث خوارزميات تقطير وهي Trickle-F وAdaptive-k و Trickle-plus مع إضافة تحسيناتنا على كل منها.

**الكلمات المفتاحية:** خوارزمية التقطير، انترنت الأشياء، بروتوكول RPL ، نظم موزعة.

# Improving the performance of IoT Trickle algorithm

**Abstract**

The Trickle algorithm that was introduced and standardized in RFC 6206 is gaining popularity as it can ensure fast and accurate resolution of data inconsistency in distributed environments with low maintenance cost, good support for scaling and increased network density. It is for these attractive features that Trickle forms the basis of many Internet standards and is deployed in many applications such as reliable broadcasting, service discovery, distributed resources, as well as routing protocols, including RPL, which is used in IoT networks. In this paper, we presented a new Trickle algorithm (New Sys-Trickle) capable of reducing both the convergence time, the number of control packets sent in the network, the amount of power consumption and the amount of processor operating time at the same time, and this is what distinguishes it from the previous algorithms that were improving a standard at the expense of a standard Else. We have combined three Trickle algorithms Trickle-F, Adaptive-k and Trickle-plus with our improvements to each of them.

**KEYWORDS**: Trickle algorithm, Internet of Things, RPL protocol, distributed system.

## 1. المقدمة:

تتطلب شبكاتLLN(Low-Power and Lossy Networks) بناء طوبولوجيا الشبكة بأسلوب يتسم بالكفاءة والسرعة، وبالتالي من الضروري وجود خطة فعالة لاختيار اتجاه البيانات والتوجيه بشكل سريع. يعتمد RPL ( IPv6 Routing Protocol for LLN ) وهو بروتوكول التوجيه في شبكاتLLN من أجل الحفاظ على الطاقة والحفاظ على طوبولوجيا الشبكة مع أقل تكلفة للتوجيه خوارزمية Trickle التي تضمن الانتشار السريع لمعلومات التوجيه وتكلفة صيانة منخفضة لطوبولوجيا الشبكة. تستخدم هذه الخوارزمية أسلوب يسمى بالثرثرة المهذبة" polite gossip" وتعني أن المشارك لن ينشر ثرثرة إذا قام شخص آخر بذلك بالفعل وبالتالي بواسطة خوارزمية Trickle يتم التنظيم الذاتي لتبادل الحزم مع العقد المجاورة[2] .

## 2. مشكلة البحث:

على الرغم من الجهود البحثية الموجودة لتحسين خوارزمية Trickle إلا أنه لا تواجد خوارزمية تقطير قادرة على تقليل كل من زمن التقارب وعدد حزم التحكم واستهلاك المعالج والطاقة معاً ,لذلك عملنا على إيجاد خوارزمية تقطير قادرة على تحسين هذه المعايير الأربعة معاً.

## 3. الهدف من البحث:

يهدف هذا البحث إلى تقديم خوارزمية تقطير جديدة ذات أداء أفضل من أداء خوارزميات التقطير في الدراسات السابقة وذلك باتباع الخطوات التالية :

- الهدف 1: الهدف الأول هو اكتساب معرفة متعمقة وإتقان أحدث ما توصلت إليه مفاهيم إنترنت الأشياء مع التركيز بشكل أكبر على بروتوكول التوجيه RPL وخوارزمية Trickle.

- الهدف2 :فهم الكود البرمجي الخاص ببروتوكول RPL داخل نظام التشغيل contiki وذلك بهدف التعديل عليه .
- الهدف 3: دراسة وتحقيق ومقارنة التحسينات على خوارزمية Trickleفي بروتوكول التوجيه RPL.
- الهدف 4: تصميم خوارزمية Trickle جديدة بهدف تحسين أداء RPL من حيث تخفيف استهلاك الطاقة وتخفيض وقت تقارب الشبكة وعدد حزم التحكم ومقدار استهلاك المعالج.حيث سيتم التعديل على الملفات التالية–rpl , rpl.h, rpl.c . dag.c , rpl–icmp6.c , rpl–timers.c

## 4. أهمية البحث:

نظرًا لحقيقة أن بروتوكول التوجيه هو أحد الأعمدة الرئيسية لهندسة الشبكات، ويتوقع بثقة ضرورة وجود شبكاتLLN ، أصبح RPL سريعًا بروتوكول التوجيه الفعلي لـ IoT (Internet Of Thing)، علاوة على ذلك لأن RPL هو بروتوكول التوجيه الموحد الوحيد لشبكات LLN حتى الآن ، فإن عدد الأعمال المنشورة حول RPL يزداد كل عام بشكل واضح ومن أهمية خوارزمية Trickle في ضبط انتشار بيانات توجيه بروتوكول RPL تنبع أهمية بحثنا هذا الذي يقدم دراسة شاملة لأهم التحسينات على هذه الخوارزمية وتقييمها ومقارنتها ومن ثم تقديم خوارزمية تقطير جديدة ومحسنة قادرة على تقديم أفضل أداء ممكن.

## 5. انترنت الأشياء:

تتمثل الرؤية الرئيسية لإنترنت الأشياء في خلق عالم ذكي يوفر مزيدًا من الذكاء للطاقة والصحة والنقل والمدن والصناعة والمباني والعديد من المجالات الأخرى، سيتم تحقيق ذلك من خلال جعل الكائنات التي نتعامل معها يوميًا مزودة بأجهزة استشعار ومحدد

هوية ومحدد مواقع ولها عنوان IP لتصبح كائنات ذكية وقادرة على التواصل ليس فقط مع الكائنات الذكية الأخرى ولكن أيضًا مع البشر [3].

### 6. بروتوكول توجيه IPv6 للشبكات منخفضة الطاقة والخسارة(RPL) [2]:

هو بروتوكول توجيه IPV6 لشبكات انترنت الأشياء حيث تم اقتراحه للشبكات ذات الوصلات المفقودة التي تكشف عن معدل عالي في أخطاء الحزم وبالإضافة إلى فصل وانقطاع الوصلات. يعتبر بروتوكول RPL من النوع Source Routing (path addressing) حيث يسمح لمُرسِل الحزمة بتحديد المسار الذي تتبعه الحزمة عبر الشبكة جزئيًا أو كلياً ويتيح للعقدة اكتشاف جميع الطرق الممكنة للهدف كما يوفر RPL دعم التوجيه لنقل البيانات إما multicast أو unicast. تنظم RPL طوبولوجيا الشبكة على شكل DAG (Directed Acyclic Graph) الذي يتم تقسيمه إلى واحد أو أكثر من DODAG (Destination Oriented DAGs) حيث يتم تحديد المسارات الافتراضية بين العقد في LLN بواسطة هذه البنية الشجرية. يمكن أن تنتمي العقدة الموجودة في DAG إلى أكثر من والد مقارنة بالشجرة التقليدية. ترتبط العقد في بنية DODAG ليس فقط بوالدها ولكن أيضًا مع العقد الأخوية[8]. يكون جذر هذه الشجرة في عقدة المصرف ويعمل هذا الجذر كنقطة عبور إلى شبكات IPv6 ويتميز بكونه عقدة لا تحتوي على نفس القيود مثل عقدة المستشعر ويسمى جهاز التوجيه الحدودي. علاوة على ذلك يتم التحكم في إرسال التحديثات إلى العقد الأخرى بواسطة جهاز التوجيه الحدودي.

#### 6-1. رسائل التحكم في RPL [8]:

a) **DODAG Information Solicitation (DIS)**: يتم إرسال هذه الرسالة من خلال عقد جديدة للانضمام إلى DODAG تلتمس من خلالها كائن المعلومات (DIO) من عقدة RPL.

b) **DODAG Information Object (DIO):** رسالة بث مرسلة بواسطة عقدة الجذر لإنشاء DODAG يحتوي DIO على معلومات عامة مثل RPLInstanceID أو الترتيب أو DODAGID ، ، إلخ.

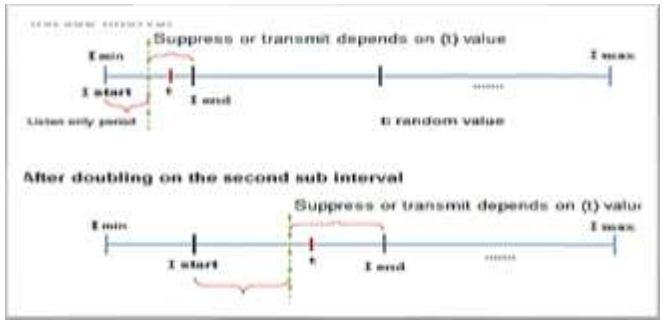
c) **Destination Advertisement Object (DAO):** إنها رسالة تُستخدم لإنشاء مسارات توجيه للأسفل من جذر DODAG إلى العقد الأخرى.

d) **Destination Advertisement Object (DAO-ACK):** ترسل هذه الرسالة من قبل مستلم رسالة DAO (جذر الـ DODAG ) كرد على رسالة DAO أحادية الإرسال المستلمة. تحتوي على البيانات المتعلقة بالحالة، تسلسلDAO ، ومعرف حالة الـ RPL .

### 6-2 . خوارزمية Trickle [9]:

يتكون RPL من مجموعة من الخوارزميات، لكل خوارزمية مهام محددة. الخوارزمية الرئيسية في RPL هي خوارزمية Trickle. يتمثل الهدف الرئيسي لخوارزمية Trickle في إدارة عملية الإرسال في الشبكة، حيث تقوم Trickleبتقليل نشر الرسائل التي لا تحتاج إلى إعادة الارسال مثل الرسائل المتكررة في الشبكة. يتم ذلك باستخدام آليتين الأولى تحدث عندما تظهر حالة عدم تناسق في الشبكة، تزيد الخوارزمية من معدل التحكم في الإشارة لتعود إلى الوضع التوافقي في الشبكة، والثانية تحدث عند تكرار نفس الرسالة في الشبكة والعقد لم تعد بحاجة إليها لأنها متصلة بجيرانها تعمل عندها الخوارزمية على كبح ارسال الرسالة وهذا يساعد على تقليل الرسائل المنتشرة على الشبكة وتوفير الطاقة. تقوم خوارزمية Trickleبتعيين الفاصل الزمني الرئيسي لكل عقدة في الشبكة، ويبدأ هذا الفاصل الرئيسي من Imin وينتهي عند Imax حيث كل من Imin وImax متغيرات. تقسم العقدة الفاصل الزمني الرئيسي الخاص بها إلى مجموعة من المناطق الفرعية، كل فاصل زمني يبدأ من Istart وينتهي في Iend وكل من Istart وIend هي متغيرات. يبدأ التنفيذ من الفترة الفرعية الأولى في العقدة، وعند

انتهاء الفترة الفرعية الأولى يبدأ الفترة الفرعية التالية، وما إلى ذلك حتى ينتهي من جميع المناطق الفرعية عندما يصل المؤقت إلى قيمة Imax .




**الشكل(1) خوارزمية Trickleالقياسية لكل عقدة[9].**

تحتوي خوارزمية Trickleالقياسية على ثلاثة بارامترات أساسية:

- Imax :الحد الأقصى لحجم الفاصل الزمني يوصف Imax على أنه عدد من مضاعفاتImin .
- Imin: الحد الأدنى لحجم الفاصل.
- K: القيمة الحدية وتسمى أيضاً بثابت التكرار k.

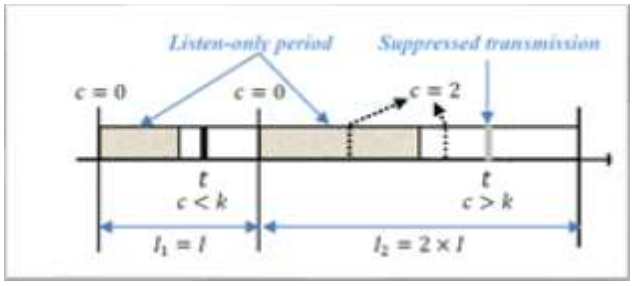
محلياً، تحافظ كل عقدة في الشبكة على مؤقت وثلاثة متغيرات:

- I: حجم الفاصل الحالي.
- C: عداد لعدد رسائل Trickle المستلمة أثناء الفاصل الحالي ويسمى عداد الاتساق.
- t: وقت الارسال وهو عشوائي محدد داخل الفاصل الحالي (I و I/2).

يمكن التعبير عن خوارزمية Trickle من خلال القواعد أو الخطوات الستة أدناه[1]:

1. **الخطوة 1:** عندما يبدأ Trickle التنفيذ فإنه يختار(I)بشكل عشوائي وموحد من [ Imin , Imin*2Imax] ويبدأ الفاصل الأول.
2. **الخطوة 2:** عند بدء فاصل زمني(I) تعيد Trickle إعادة تعيين c إلى 0 وتختار t بشكل عشوائي من النطاق [I/2,I].
3. **الخطوة 3:** كلما سمعت عقدة انتقال متناسق مع بياناتها تزيد Trickle عداد الاتساق c.

4. **الخطوة 4:** في الوقت t يرسل Trickle إذا كان (c <k) فقط وخلاف ذلك يتم قمع الإرسال.
5. **الخطوة 5:** عند انتهاء الفاصل الزمني I يضاعف Trickle طول الفاصل حتى الوقت المحدد بواسطة Imax ثم يبدأ فاصل زمني جديد كما في الخطوة 2.
6. **الخطوة 6:** إذا سمع Trickle انتقال غير متناسق بينما I أكبر من Imin فإنه يصفر المؤقت. للقيام بذلك يقوم Trickle بجعل I يساوي Imin ويبدأ فاصلًا جديدًا كما في الخطوة 2، وإلا مثلاً I تساوي Imin عند اكتشاف التناقض ولن يفعل Trickle شيئًا. يضمن اختيار t من النصف الثاني من الفاصل الزمني في الخطوة 2 فترة استماع تعادل فقط نصف الفاصل ويتم ذلك استجابة لمشكلة الاستماع القصير.




**الشكل(2) خوارزمية Trickle على فترتين مع $k = 1$ .الخط الأسود عبارة عن عملية ارسال ونقل، والرمادي هو ارسال تم منعه والخطوط المنقطة هي عمليات استقبال رسائل[1].**

7. **الدراسات السابقة:** سنشرح في هذا القسم أهم الدراسات التي حسنت خوارزمية التقطير لنقوم في الأقسام التالية بتحقيقها ومقارنتها ثم إضافة تحسيننا عليها.

- **Trickle–F** : اقترح الباحث في [11] نسخة معدلة من خوارزمية Trickle وهي Trickle–F والتي تهدف إلى حل مشكلة موازنة التحميل من خلال ضمان آلية عادلة لكبح البث على المدى القصير بين العقد في المنطقة لتسهيل الاكتشاف السريع لجميع المسارات المتاحة. يتمثل الأساس المنطقي وراء Trickle–F في تحديد أولوية كل عقدة اعتمادًا على عدد عمليات القمع المتتالية بمعنى أنه كلما طال الوقت الذي تقضيه العقدة دون الإرسال زادت أولويتها للإرسال في الجولة التالية.

- **I–Trickle**: الخوارزمية المقترحة في[12]والتي تدعى I–Trickle تضع قيمة عداد التكرار redundancy counter عند قيمة الصفر ليس في بداية الفاصل الزمني الجديد ولكن في وقت منع أو إرسال رسالة كائن معلومات DODAG في خوارزمية Trickle من أجل حل مشكلة موازنة الحمل وتقليل استهلاك الطاقة من خلال مراعاة الرسائل التي يتم سماعها من وقت t إلى نهاية الفاصل الزمني.
- **Optimized Trickle**: تم الاقتراح في [1] تحسين بسيط وقوي يقلل من التأخير دون تكبد أي نفقات عامة إضافية. في هذه الخوارزمية المحسّنة عندما يتم اكتشاف عدم تناسق، يعيد تعيين I إلى Imin الذي يتم أيضًا في خوارزمية Trickle الأصلية. لكنه يقول أنه في هذا الوقت سيختار الوقت العشوائي t في [0,Imin] بدلاً من [I/2,I] كما هو الحال في خوارزمية Trickle الأصلية.
- **E–Trickle** : في الورقة[14] تم اقتراح نسخة محسنة من Trickleوهي–E Trickle حيث تقدم هذه الخوارزمية ثلاث تعديلات لخوارزمية Trickle. أولاً بدلاً من تحديد الوقت العشوائي t من النطاق [ I / 2 ، I] يتم تحديد قيمة t من النطاق [0 ، I]. ثانياً بدلاً من تعيين عداد التكرار c على قيمة 0 في بداية كل فاصل زمني يتم تعيين c إلى قيمة 0 فقط في بداية الفاصل الزمني الأول Imin وأيضًا في الوقت الذي تم اختياره عشوائيًا t وذلك من أجل القضاء على التأثير التراكمي لمشكلة الاستماع القصير. كما قد يلاحظ المرء، فإن ضبط عداد التكرار c إلى 0 في الوقت المحدد عشوائياً سيؤدي إلى فواصل زمنية غير متساوية بين العقد من حيث الطول وبالتالي، فإن العقد ذات طول الفاصل الأقل سيكون لها فرص أكبر للإرسال. للتغلب على هذا الموقف تم اضافة خطوة ثالثة لتعديل قيمة عامل التكرار k حيث تصبح قيمته (newk= ((2*Inz–I)/I)*k) بشكل متكيف لتعكس التمدد الذي يحدث في حجم الفاصل .
- **Adaptive–k**: يقترح الباحث في [15] امتدادًا لخوارزمية Trickle التي تسمى adaptive–kوالتي تسمح للعقد بتعيين ثابت التكرار الخاص بها وفقًا للمعلومات المحلية حول كثافة الشبكة.حيث تعتمد كل عقدة على العداد c لتخمين عدد جيرانها وتسمح للعقد بتعيين قيمتها k بشكل مستقل .

- **Trickle–Plus**:يوجد في [10] نسخة موسعة من خوارزمية Trickle. في هذه النسخة بدلاً من زيادة حجم نافذة الإرسال أضعافا مضاعفة، يتم وضع حجم الفاصل الزمني الجديد بالقيمة (I*SF*2)حيث SF هو عامل الازاحة ويعني عدد مرات مضاعفة الفاصل الزمني التي يمكن تخطيها، وبالتالي تتقارب الشبكة بسرعة مع استهلاك أقل للطاقة.
- **RANDOMIZED DYNAMIC TRICKLE**: اقترح الباحث في [7] تحسينًا لخوارزمية Trickle القياسية مرتبطاً بقيمةt .في الخوارزمية المقترحة يتم تحديد وقت لفترة الاستماع وفترة الإرسال بناءً على قيمة t ضمن فترة زمنية فرعية. المتغيرt هو وقت عشوائي يستخدم كمحدد للاستماع والإرسال. لذلك يركز هذا العمل بشكل أساسي على اختيار قيمة t. حيث يتم اختيار قيمة t وفقًا لكثافة الشبكة، لتحقيق ذلك يتم اقتراح أربع حالات لاختيار قيمة t أفضل في الفترة الفرعية. في كل مرة يتم التحقق من عدد الجيران (C) لتحديد النطاق الذي يجب أن يتم اختيار فيه وقتًا عشوائيًا (t).
- **FL–Trickle**:قدم الباحثون في [6] تحسينًا جديدًا لخوارزمية Trickle القياسية وهي خوارزمية Flexible Trickle (FL–Trickle) تقلل FL–Trickle من التأخير في إرسال رسائل التحكم عن طريق تحديد وقت الإرسال T عند2 / I بدلاً من اختياره بشكل عشوائي في [ I/2، I ] .تقلل FL–Trickle من معدل النقل من خلال العمل بقيمة عالية من الحد الأدنى للفاصل الزمني (Imin) من أجل الحصول على عبء منخفض مع عدم وجود زيادة في استهلاك الطاقة.

## 8. القسم العملي :

### 8–1 . نظام التشغيل Contiki ومحاكي الشبكات Cooja [16]:

تم اختيار Contiki لأنه تم تصميمه خصيصًا لأجهزة IoT منخفضة الطاقة والخسارة ولديه تطبيق أساسي لخوارزمية Trickle في مكتبة ContikiRPL التي سيتم استخدامها كأساس لبحثنا . كما تم اختيار المحاكي Cooja الذي يعمل على نظام تشغيل

Contiki (الإصدار 3.0). يعتمد اختيار هذا المحاكي على اعتبارين رئيسيين: الأول هو محاكي مفتوح المصدر مصمم لتطبيقات إنترنت الأشياء. ثانيًا يجعل تعديل وتحسين خوارزمية Trickleسهلًا جدًا بسبب تنفيذه على RPL الأساسي. كما يوفر محاكي Coojaأدوات لإخراج البيانات من كل اختبار بسهولة شديدة بتنسيق قابل للقراءة. هذا ما يسهل علينا استخلاص البيانات منها وإنتاج الرسوم البيانية من المخرجات باستخدام نصوص Perl لتصفية البيانات الناتجة إلى مقاييس الأداء المرغوب فيها، بالإضافة إلى ذلك يوفر هذا المحاكي أداة تدعى collect view التي يمكن للمبرمج دمجها بالكود الخاص به وتعرض معلومات ومخططات مفيدة في عملية قياس الأداء.

### 8-2. المقاييس المستخدمة لتقييم الأداء ومقارنته:

- **The Convergence Time(setup time)**:وقت تقارب الشبكة مقدار الوقت الذي تحتاجه جميع العقد القابلة للوصول (من حيث الراديو) في الشبكة للانضمام إلى DAG.

$$Convergence\ Time = Last\ DIO\ joined\ DAG - First\ DIO\ sent$$

- **Energy Consumption**:مقدار استهلاك الطاقة بالميلي واط من قبل كل من المعالج والاستماع والارسال ونمط توفير الطاقة LPM.
- **Control Traffic Overhead**: وهذا يشمل رسائل DIO و DIS و DAO التي تم إنشاؤها بواسطة كل عقدة.

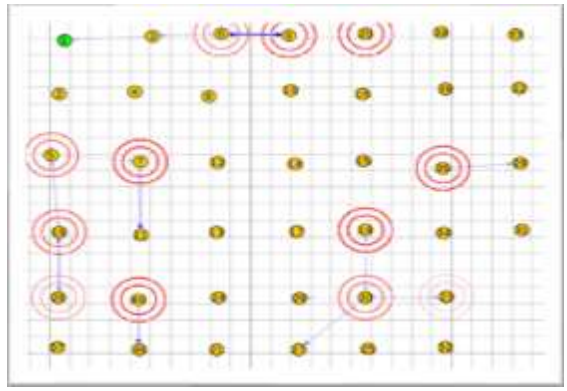
$$Control\ Traffic\ Overhead = \sum_{k=1}^{n} DIO(k) + \sum_{k=1}^{m} DIS(k) + \sum_{k=1}^{o} DAO(k)$$

- **Total CPU consumption (ticks)**:مقدار الوقت الذي تم تشغيل المعالج به وذلك لجميع العقد.

لم يتم استخدام معدل تسليم الرزم والتأخير من طرف لطرف في المقارنة وذلك لأنَّ خوارزمية التقطير تقلل عدد حزم التحكم وليس لها تأثير مهم على هذين المعاملين [4].

### 8-3. معاملات المحاكاة[5] :

نظرًا لأن الاختبارات التي تم إجراؤها لمدة 7 دقائق أسفرت عن نتائج مشابهة لتلك التي أجريت لمدة 20 دقيقة، فقد اخترنا 7 دقائق كوقت المحاكاة لجميع عمليات محاكاة Cooja.تحوي طوبولوجيا الشبكة على 20 ثم 40عقدة عميل في منطقة 100 م × 100 م. تمثل العقدة 1 في الزاوية اليسرى العليا جهاز التوجيه الحدودي. يتم وضع جهاز توجيه الحدود في الزاوية بحيث تكون هناك عُقد ذات مسافات إرسال متعددة القفزات من جهاز توجيه الحدود. يوضح الشكل التالي طوبولوجيا الشبكة المستخدمة في المحاكاة:



**الشكل (3) طوبولوجيا الشبكة المستخدمة في المحاكاة تحوي 40 عقدة بمصرف واحد.**

### 8-4. مستوى التطبيقApplication Level:

لتشغيل الاختبارات قمنا باستخدام نموذج تطبيق UDP Contiki يسمى "Hello World!".يرسل هذا التطبيق البسيط رسالة "مرحبًا" في فاصل زمني محدد. بحيث ترسل كل عقدة عميل رسالة "مرحبًا" إلى جهاز التوجيه الحدودي. يستخدم موجه الحدود الملف udpserver.c وجميع عُقد المستشعر تستخدم الملف udp-client.c .لقد قمنا بتعديل شيفرة المصدر لـ RPL في Contiki لكتابة خوارزمياتTrickle حيث تم التعديل على الملفrpl-timers.c .

سنستخدم ملحق Cooja المسمى Contiki Test Editor لقياس وقت المحاكاة وإيقاف المحاكاة بعد الوقت المحدد. ينشئ هذا البرنامج المساعد أيضًا ملف سجل

(COOJA.testlog)لجميع مخرجات المحاكاة التي سنقوم بتحليلها في نهاية المحاكاة باستخدام برنامج نصي مكتوب بلغة Perl.من أجل إدخال الضياع في الوسط اللاسلكي نستخدم Cooja Unit Disk Graph Medium، الذي يقدم الضياع فيما يتعلق بالمسافات النسبية للعقد في وسيط الراديو.كما هو موضح في الجدول التالي، فإن تأخير البدء start delay هو تأخير البدء الأولي للتطبيق لبدء إرسال الرسائل إلى عقدة المصرف. وقت البدء الأولي هذا هو الوقت التقريبي الكافي لتقارب الشبكة الأولي. وهذا يضمن أيضًا عدم فقدان الحزمة المرسلة إلى الخادم بسبب نقص اتصال الشبكة. لذلك يمكن إجراء تقييم صحيح لعدد الحزم المرسلة. يتم تعيين DIO Min و DIO Doublingsعلى القيم الافتراضية في ContikiRPL .تمثل نسبة الاستقبال (RX) مدى ضياع الوسيط الراديوي ويتم تعيينه بالنسب المئوية أثناء التكرار المتتالي للمحاكاة.يتم تعيين نسبة الإرسال (TX) على 100٪ (بدون خسارة) لأننا لا نهدف إلى إحداث خسائر عند طرف الإرسال ولكن فقط عند طرف الاستقبال .تم ضبط نطاق TX على 50 م ونطاق التداخل على 55 م .

**الجدول(1) اعدادات المحاكاة.**

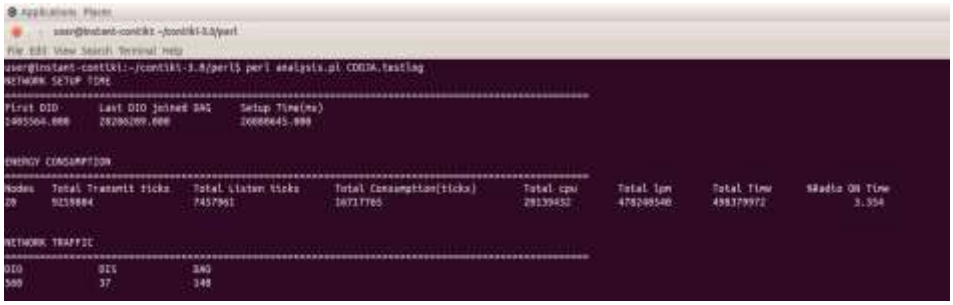
| Parameters | Value |
|---|---|
| Start Delay | 65 s |
| RPL MOP | NO_DOWNWARD_ROUTE |
| **OF** | ETX |
| DIO Min | 8 |
| DIO Doublings | 12 |
| RDC Chanel Check Rate | 16 |
| Send Interval | 4 s |
| **RX Ratio** | 30-100% |
| TX Ratio | 100% |
| TX Range | 50m |
| Interference Range | 55m |
| Simulation Time | 7 min |
| Client Nodes | 20,40 |

## 9. التجارب والنتائج:

قمنا ومن أجل كل خوارزمية بإعادة التنفيذ عشر مرات وأخذ متوسط القيم الناتجة عن التجارب وذلك بعد استثناء القيم الشاذة والمتطرفة لبعض التجارب بتطبيق قانون المدى الربيعي عن القيم العشر الناتجة.

- مثال لنتيجة تحليل الملف COOJA.testlogباستخدام ملف التحليل analysis.pl الذي قمنا بكتابته بلغة PERL وذلك لخوارزمية Trickle الأصلية:


**الشكل (4) نتيجة تحليل ملف الخرج COOJA.testlog حالة خوارزمية Trickle وشبكة تحتوي 20 عقدة.**




**الشكل (5) نتيجة تحليل ملف الخرج COOJA.testlog حالة خوارزمية Trickle وشبكة تحتوي 40 عقدة.**

### 9-1. ملخص النتائج قبل إضافة التحسين:

بناءً على ما سبق يمكننا رسم الجدول التالي الذي يحوي ملخص النتائج حيث ترمز الإشارة (+) إلى أن الخوارزمية المقابلة قدمت أداء أفضل من الخوارزمية الأصلية

بالنسبة للمعيار المقابل وترمز الإشارة(–) إلى أنَّ الخوارزمية المقابلة قد قدمت أداء أسوأ من الخوارزمية الأصلية بالنسبة للمعيار المقابل بينما ترمز الإشارة (*) إلى أن الخوارزمية المقابلة قدمت أفضل أداء من بين جميع الخوارزميات الباقية أما الفراغ يعني أن لها نفس أداء الخوارزمية الأصلية:

**الجدول (2) تلخيص النتائج قبل إضافة التحسين.**

| Total CPU | Energy Cosum 40 20 | Traffic Overhead | Setup Time | Metric / Algorithm |
|---|---|---|---|---|
| – | | | + | Trickle–F |
| – | – | – | * | I–trickle |
| – | – | – | + | Opt–trickle |
| – | – | – | + | E–trickle |
| + | * | + | | Adaptive–k |
| * | + | + | – | Trickle–plus |
| – | | – | + | RD–trickle |
| + | – | * | – | FL–trickle |

نلاحظ من الجدول السابق أن الخوارزمية التي تقدم أفضل (أقل) زمن تقارب هي خوارزميةI–trickle ولكن ذلك كان على حساب عدد حزم التحكم المرسلة في الشبكة وكمية استهلاك الطاقة اللذين ازدادا عما كانا عليه في Trickle الأصلية. تقدمTrickle–plus أقل عدد حزم تحكم مرسلة في الشبكة لكنها بالمقابل تزيد من زمن التقارب وكذلك FL–trickle التي تقلل من عدد حزم التحكم على حساب زيادة زمن التقارب. تقدم Adaptive–k أقل كمية استهلاك طاقة في الشبكة .

### 9–2. مناقشة النتائج:

- تحسن Trickle–F من زمن التقارب نعلل ذلك بسبب جعل فترة الاستماع فقط للعقد المحرومة من الارسال لفترات أطول من غيرها أصغر ما يمكن بحيث تتناسب طرداً مع عدد فترات الحرمان من الارسال مما يعطي فرصة أكبر للعقد العائمة

الغير منضمة إلى DODAG من تلقي رسالة DIO أسرع ما يمكن . أما بالنسبة لعدد حزم التحكم الذي بقي بنفس القيمة تقريبا كما في خوارزمية التقطير الأصلية فيعود ذلك إلى كون إبقاء الخوارزمية أطوال الفواصل الزمنية كما هي في الخوارزمية الأصلية وعند دعمها لعقد للإرسال كانت بالمقابل تحرم عقد أخرى من الارسال لذلك لم تشكل زيادة في عدد حزم التحكم وتقديم الخوارزمية فترة استماع فقط كافية بالنسبة لوضع كل عقدة كما نعلم أن مقدار استهلاك الطاقة يعود إلى أربع أشياء تستهلك الطاقة وهيlow power mode –cpu listen power– transmit power –power بمعنى أن استهلاك الطاقة يتأثر بعدد حزم التحكم المرسلة والمستقبلة وبما أنها بقيت كما في الخوارزمية الأصلية لذلك فإن استهلاك الطاقة بقي كما في الخوارزمية الأصلية .أما بالنسبة لعدد أجزاء الوقت التي تم استخدام المعالج فيها فقد ازداد بشكل بسيط وذلك بسبب عملية حساب طول فترة الاستماع فقط .

- تحسن trickle–امن زمن التقارب وذلك لأنها تتبع نفس أسلوب trickle–F ولكنها تجعل فترة اختيار t من 0 إلى (I/2^s) أي أن فترة الاستماع فقط قد تكون معدومة في بعض الفواصل وبالتالي جعلت فترة الاستماع فقط أقل من سابقتها مما يتيح المجال إلى ارسال الرسائل التحكم بشكل أسرع وانضمام أسرع للبيان الموجه الخاص بالشبكة. تظهر جلياً في هذه الخوارزمية مشكلة الاستماع القصير وعدم التزامن حيث بدون التزامن يمكن أن تعاني Trickle من مشكلة الاستماع القصير.أي أن بعض المجموعات الفرعية من العقد تستمع وقت قصير من بداية الفاصل الزمني وقبل قدرة أي عقدة أخرى لديها فرصة للإرسال على الارسال. إذا تمت مزامنة جميع الفواصل الزمنية عندها فستمنع عملية ارسال العقدة الاولى أي عقدة أخرى من الارسال، ومع ذلك إذا لم تتم مزامنتها فربما تبدأ العقدة الفاصل الزمني بعد البث مباشرةً وقد تختار أيضًا فترة استماع قصيرة مما ينتج عنه ارسالات متكررة. إن الاستماع القصير أدى الى زيادة عدد حزم التحكم واستهلاك الطاقة واستخدام المعالج زاد بسبب الحسابات.

- بالنسبة لخوارزمية optimal trickle و E-trickle فإن اختيارt ضمن المجال[0,Imin] يقلل من زمن تقارب الشبكة ولكن ممكن أن يؤدي إلى ظهور فترة استماع فقط شبه معدومة وتظهر مشكلة الاستماع القصير التي تعني ان العقدة قد تستمع لوقت قصير وتكرر ارسالها لمعلومات مكررة وبالتالي زيادة حزم التحكم في الشبكة وتحسن زمن التقارب في الوقت ذاته .

- خوارزمية Adaptive-k تعتمد على فكرة ضبط قيمة k للفاصل القادم اعتمادا على قيمة c السابق إن هذه العملية والحصول على قيمة kتمثل الجوار يقضي على الارسال الزائد عن الحاجة الذي قد يظهر عندما تكون قيمة k منخفضة والشبكة كثيفة ويقضي على عملية قمع العقدة من الارسال عند ضبط قيمة k على قيمة عالية والشبكة كانت منخفضة الكثافة . لا تحسن هذه الخوارزمية زمن التقارب الذي يبقى كما هو في الخوارزمية الأصلية بينما بسبب تلافي الارسالات المتكررة بشكل فعال تؤدي الى تقليل عدد حزم التحكم واستهلاك الطاقة والمعالج.

- تقوم فكرة خوارزمية trickle plus على عملية تلافي أطوال زمنية وتجاوزها والانتقال إلى الاطوال الأعلى بشكل أسرع مما يجعل عدد رسائل التحكم أقل من حالة trickleالأصلية .تعتمد trickle plus على فترة استماع فقط تساوي نصف الفاصل الزمني كما وأنه من أجل بداية عملية الازاحة بالقرب من Imin أدى ذلك إلى ازدياد زمن تقارب الشبكة لأن الخوارزمية تبدأ بمضاعفة الفاصل الزمني قبل انضمام جميع العقد للشبكة وتلقيها رسائل DIO.إن تخفيض عدد رسائل التحكم يخفض بدوره مقدار استهلاك الطاقة اللازمة لإرسال الرسائل كما أن تجاوز اطوال زمنية وانتقال الى اطوال اعلى يجعل عدد عمليات زيادة c ومقارنتها ب k أقل مما يجعل هذه الخوارزمية الأقل استخداماً للمعالج.

- أما خوارزمية RD-trickleنلاحظ انخفاض زمن التقارب بسبب جعل t وفترة الاستماع فقط تتناسب مع عدد الجيران وبالتالي في العقد الحدودية تصبح هذه الفترة

شبه معدومة مما يولد رسائل تحكم زائدة عن الحاجة مما يؤدي الى زيادة استهلاك الطاقة والمعالج.

- تتشابه FL-trickle مع trickle plus حيث حسنت عدد حزم التحكم ولكن ساء زمن التقارب بالمقابل .

## 10. خوارزمية التقطير الجديدة المتكيفة والمتزامنة New sys-trickle:

انطلاقاً من كون وجود ثلاث خوارزميات Trickle كل منها قادرة على تحسين معامل أداء واحد قررنا في هذا البحث دمج الخوارزميات الثلاثة مع بعضها البعض مع إضافة تحسيننا على كل منها .

الخوارزمية الأولى هي خوارزمية I-trickle التي هي بالأصل تحسين على Trickle-F وانطلاقاً من أن مزامنة الفواصل الزمنية أمر هام جداً وذلك لأنه عند عدم وجود تزامن قد تتزامن فترة الاستماع فقط مع فترة الارسال لعقدة بشكل مستمر مما يؤدي إلى تكرار الرسائل من العقدة الأولى وقمع الرسائل من العقدة الثانية لذلك قدمنا تعديل هام على خوارزمية Trickle-F يجعل من فواصلها متزامنة كما في الشكل التالي :

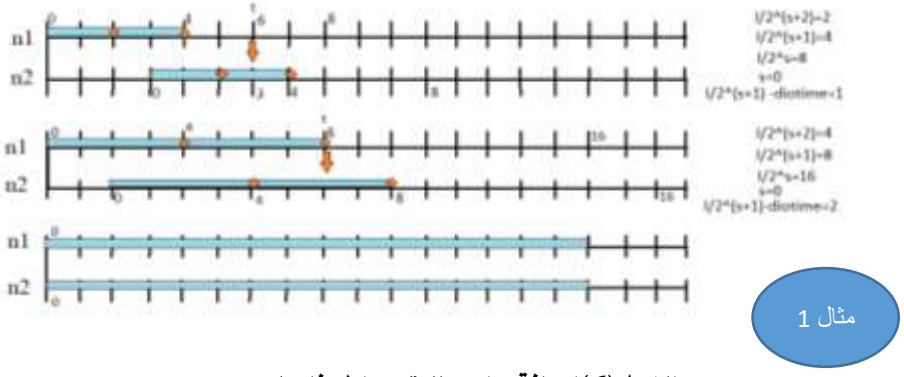
في Trickle-F هناك فترة استماع فقط تنتمي إلى المجال [0, I/2^S+1] بالتالي فإن الفترة المخصصة لاختيار t ضمنها هي [1/2^s+1,1/2^s] نلاحظ أن العقدة تكون غير متزامنة عندما تستلم في النصف الثاني من فترة الاستماع فقط رسائل DIO لذلك نقوم بحساب الاختلاف الزمني بين العقدتين من خلال طرح زمن وصول أول رسالة DIO ضمن الفاصل الزمني [I/2^s+1, I/2^s+2] من I/2^s+1 .

عند طرح قيمة اللحظة التي استلمت فيها العقدة رسالة DIO (diotime) من I/2^s+1 سنحصل هنا على زمن tsys الذي سنقوم ببدء الفاصل الزمني التالي على أساسه حيث أن:I=I-tsys أي سنجعل الفاصل الزمني التالي يبدأ بشكل أبكر بمقدار يساوي tsys .

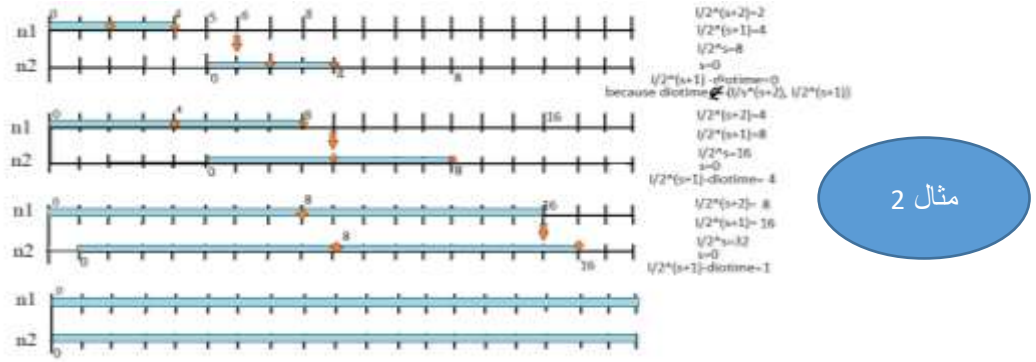
إن جعل الفاصل الزمني للعقد يبدأ بشكل أبكر يؤدي إلى جعل الشبكة تتقارب بشكل أسرع كما يؤدي إلى تلافي منع وقمع عقد قد تمتلك معلومات توجيه جديدة بشكل متكرر

من الارسال وتلافي منح فرصة الارسال بشكل متكرر لعقد أخرى الذي يؤدي إلى نقل معلومات مكررة ضمن الشبكة ,وبالتالي تحقيق موازنة جيدة للحمل.

سنقوم بهذه الخطوة فقط في بداية المحاكاة وأثناء عملية الانضمام إلى الشبكة أي في أول أربع فواصل زمنية وذلك لأنه في هذه الفترة تكون قيمة s التي تمثل عدد الفواصل التي تمنع فيها العقدة من الارسال صفر , وبالتالي تكون فترات الاستماع للأب والابن متساوية لأنه عندما تكون هذه الفترات غير متساوية سوف تتلقى العقدة من جيرانها رسائل DIO في الجزء الثاني من فترة الاستماع فقط على الرغم من تزامن الفواصل الزمنية للعقد الذي قمنا بتحقيقه مسبقاً.




**الشكل(6)إضافة تزامن للعقد خلال فاصلين زمنين.**

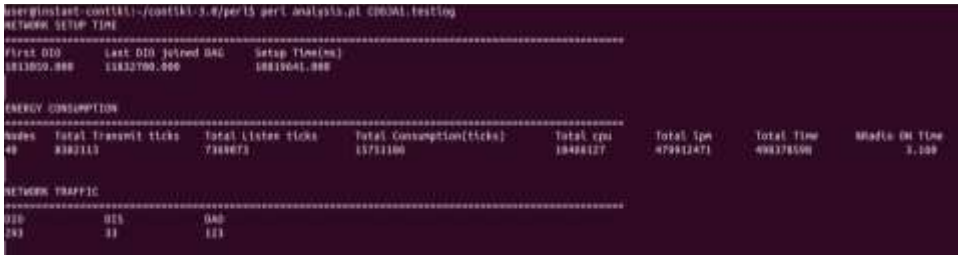



**الشكل(7) إضافة تزامن للعقد خلال ثلاث فواصل زمنية.**

الفكرة الثانية هي في خوارزمية FL–trickle التي هي عبارة عن تحسين على خوارزمية trickle –plus تفرض هذه الخوارزمية أن الازاحة ستكون عشوائية وفي بعض التجارب قد تبدأ الازاحة قبل انضمام جميع العقد وبالتالي زيادة زمن التقارب. التحسين هو أنه سنعمل على جميع الفواصل الزمنية ونجعل iss تساوي Imin و ise ستكون Imax ثم سنعرف متحول يدعى is1 الذي يعبر عن النقطة الفاصلة بين مضاعفة الفاصل بشكل معتاد كما في الخوارزمية الأصلية و الانتقال مباشرة إلى طول الفاصل الزمني الاعظمي , أي أنه عند المرور بمجموعة من الفواصل الزمنية التي تعتبر كافية لتقارب الشبكة سننتقل مباشرة إلى طول الفاصل الأعظمي Imax وبالتالي سنكون ضمنا أنَّ الإزاحة وتلافي الفواصل لن يؤثر على زمن التقارب كما أن الانتقال مباشرة إلى الطول الأعظمي يقلل من عدد رسائل التحكم بشكل واضح.

الفكرة الثالثة هي في خوارزمية adaptive–k تقوم هذه الخوارزمية بضبط قيمة k حسب عدد الجيران وبالتالي مع كل فاصل زمني جديد قد تظهر قيمة جديدة لـ k ولكن يمكن في بعض الفواصل استقبال رسائل من نصف عدد الجيران أو ربعهم أو من لا احد وبالتالي تغيير قيمة k غير فعال .لذلك اقترحنا إضافة متحول نضع فيه قيمة k من الفاصل السابق إذا كانت قيمة k الجديدة أكبر من القديمة عندها لن نغير قيمة المتحول وستبقى في k القيمة الأكبر التي حصلنا عليها من الفواصل السابقة.


**الشكل (8) نتيجة تحليل ملف الخرج COOJA.testlog حالة خوارزمية New Sys– Trickle وشبكة تحتوي 20 عقدة.**




**الشكل (9) نتيجة تحليل ملف الخرج COOJA.testlog حالة خوارزمية New Sys- Trickle وشبكة تحتوي 40 عقدة.**

```
Algorithm  new sys-Trickle
  function INITIALIZATION()
        I← I_min   sys= 0   kmid=0  Iss=Imin
        s ← 0       diotime= 0   Is1=5
  function INTERVALBEGINS()
        c ← 0
        sys = 0
        t ← random(I/2^s+1, I/2^s)
  function CONSISTENTTRANSMISSIONRECEIVED()
        c ← c + 1
      diotime = clock _time ()
        if diotime > I /2^s+2 && diotime < I/2^s+1
            sys= I/2^s+1 - diotime

  function TIMEREXPIRES()

      if k ≥ c then
            Transmit DIO
            s ← 0
      else
            s ← s + 1
      end if    if sys>0 && I <5  I=I-sys
   function INTERVALENDS()
          c ← 0
        newk=0.5*c
      if newk<kmin      k=kmin
     else if newk>kmax       k=kmax
   else if newk >kmin && newk<kmax   k = floor(newk)
     if kmid > k   k=kmid
     else kmid =k
    if  I> Iss && I< Is1
     I = I * 2
  else
    I=Imax

  if InconsistentTransmissionReceived then
        I ← I_min
        s ← 0
```

**الشكل(10) كود الخوارزمية.**

## 11. النتائج النهائية:

❖ **زمن التقارب(The Convergence Time (us :**

- الحالة الأولى شبكة تحتوي 20 عقدة موزعة بشكل منتظم:

**الجدول(3) متوسط زمن التقارب لجميع الخوارزميات حالة شبكة تحوي 20 عقدة.**

Setup Time (us)

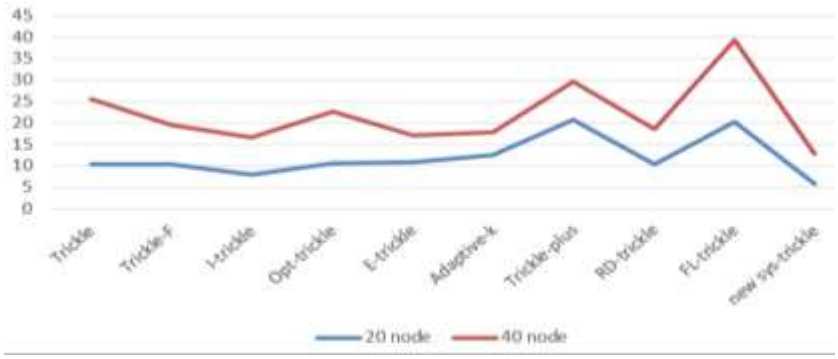
| Trickle | Trickle-F | I-trickle | Opt-trickle | E-trickle | Adaptive-k | Trickle-plus | RD-trickle | FL-TRICKLE | new syn-trickle |
|---|---|---|---|---|---|---|---|---|---|
| 10466754 | 17268865 | 7274411 | 9035959 | 7351746 | 8788046 | 53730100 | 11014344 | 61726613 | 3642894 |
| 16389861 | 18955903 | 5442737 | | 9194282 | 7008196 | 13079747 | 9841102 | 13679746 | 7521462 |
| 9533161 | 10660428 | 6708513 | 13258877 | 13916589 | 6542192 | 28662655 | 9917260 | 28662654 | 7797528 |
| 8782110 | 12763870 | 5948167 | 10208975 | 11039528 | 15819565 | 3197414 | 8983927 | 3197413 | 7783419 |
| 8490822 | 9681074 | 9733845 | 12492798 | 9883283 | 22856277 | 17299255 | 7733250 | 4733140 | 5163264 |
| 8502929 | 8128507 | 11899611 | 8749572 | 13923068 | 15943680 | 15872640 | 13076755 | 13072639 | 3884739 |
| 9468831 | 6314797 | 5348359 | 9562640 | 15018232 | 18042928 | | 7046437 | 61543651 | 4241321 |
| 7375552 | 7251512 | 8516019 | 9816738 | 9874613 | 11140887 | 14364353 | 10837103 | 14364352 | 4489129 |
| 14168758 | 13080801 | 11312998 | 15626168 | 11800852 | 11252943 | 4773977 | 11991922 | 4773976 | 6332014 |
| 9159267 | 5989226 | 5543932 | 6883483 | 6818987 | 9283698 | 4986785 | 12758695 | 4986784 | 7955311 |
| 10253526.3 | 10253234.1 | 7980769.4 | 10128266 | 10761838 | 12466264 | 20787427.33 | 10240885.5 | 20334828.8 | 5903186.1 |
| 10.2535 | 10.2532 | 7.980769 | 10.128 | 10.76183 | 12.466 | 20.787 | 10.240 | 20.334 | 5.90 |

- الحالة الثانية شبكة تحتوي 40 عقدة موزعة بشكل منتظم:

**الجدول(4) متوسط زمن التقارب لجميع الخوارزميات حالة شبكة تحوي 40 عقدة.**

Setup Time (us)

| Trickle | Trickle-F | I-trickle | Opt-trickle | E-trickle | Adaptive-k | Trickle-plus | RD-trickle | FL-TRICKLE | new syn-trickle |
|---|---|---|---|---|---|---|---|---|---|
| 48855148 | 17416753 | 14648888 | 18698322 | 15697859 | 12159532 | 32735529 | 19773911 | 52238323 | 18819641 |
| 17349523 | 22843382 | 28383758 | 28176269 | 28042879 | 14385461 | 19246483 | 15118974 | 21378268 | 17691435 |
| 28357626 | 24867818 | 14287715 | 28031817 | | 23283283 | 21856557 | 16940838 | 61855813 | 14972835 |
| 19416853 | 29780928 | 21170419 | 11486959 | 15649781 | 18898253 | 61974434 | 13575898 | 39973721 | 14288358 |
| 17475568 | 16936437 | 15724871 | 22290939 | 17986915 | 18892852 | 21185596 | 22680196 | 31624718 | 8823179 |
| 26880645 | | 11190896 | 15718811 | 19386832 | 11734759 | 13953223 | 22931517 | 14365945 | 15811296 |
| 11891187 | 16460855 | 15857995 | 38599584 | 18833311 | 14613986 | 18371649 | 24178340 | 18371648 | 18545684 |
| 14534196 | 14365158 | 9638225 | 28842332 | 24283226 | 14866878 | 25333581 | 22413630 | 25885222 | 18498384 |
| 10111137 | 15623516 | 27398113 | 15698698 | 28818759 | 11466285 | 28888885 | 13377894 | 73888878 | 13972883 |
| 21510616 | 17989378 | 16870917 | 18985362 | 15469556 | 17815004 | 61200884 | 15523752 | 71885237 | 8493642 |
| 25438228.9 | 19588602.6 | 16699515.7 | 22651018.5 | 17121812.8 | 17732812.5 | 29679866.1 | 18624595 | 39177817.3 | 12766557.4 |
| 25.438 | 19.58 | 16.69 | 22.65 | 17.12 | 17.73 | 29.67 | 18.624 | 39.177 | 12.766 |




**الشكل(11)المخطط الخطي لزمن التقارب.**

❖ **Traffic Overhead(عدد حزم التحكم):**

- الحالة الأولى شبكة تحتوي 20 عقدة موزعة بشكل منتظم:

**الجدول(4) متوسط عدد حزم التحكم لجميع الخوارزميات حالة شبكة تحوي 20 عقدة.**

Traffic Overhead

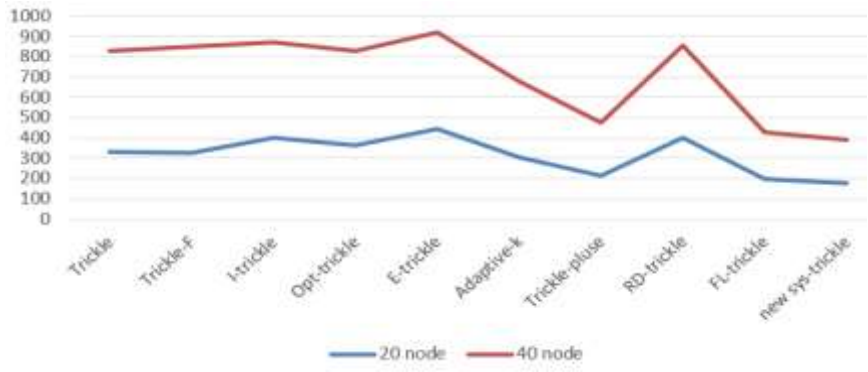
| Trickle | Trickle-F | I-trickle | Opt-trickle | E-trickle | Adaptive-k | Trickle-plus | RD-trickle | FL-TRICKLE | new syn-trickle |
|---|---|---|---|---|---|---|---|---|---|
| 306 | 305 | 364 | 359 | 388 | 283 | 212 | 364 | 199 | 178 |
| 332 | 324 | 478 | | 456 | 311 | 211 | 374 | 177 | 182 |
| 291 | 366 | 350 | 336 | 505 | 285 | 254 | 448 | 264 | 169 |
| 312 | 314 | 391 | 374 | 421 | 324 | 222 | 354 | 183 | 161 |
| 298 | 358 | 430 | 410 | 457 | 299 | 190 | 392 | 196 | 182 |
| 352 | 344 | 372 | 354 | 420 | 329 | 218 | 379 | 186 | 184 |
| 321 | 295 | 375 | 352 | 429 | 286 | | 521 | 203 | 187 |
| 302 | 317 | 375 | 364 | 430 | 323 | 202 | 389 | 181 | 165 |
| 342 | 306 | 410 | 366 | 436 | 323 | 207 | 358 | 192 | 194 |
| 383 | 333 | 438 | 337 | 512 | 288 | 213 | 441 | 177 | 179 |
| 330.9 | 325.7 | 399.9 | 361.3 | 445.4 | 305.1 | 215.2 | 402 | 195.8 | 178.1 |

- الحالة الثانية شبكة تحتوي 40 عقدة موزعة بشكل منتظم:

**الجدول(5) متوسط عدد حزم التحكم لجميع الخوارزميات حالة شبكة تحوي 40 عقدة.**

Traffic Overhead

| Trickle | Trickle-F | I-trickle | Opt-trickle | E-trickle | Adaptive-k | Trickle-plus | RD-trickle | FL-TRICKLE | new syn-trickle |
|---|---|---|---|---|---|---|---|---|---|
| 933 | 778 | 916 | 767 | 1186 | 642 | 451 | 763 | 376 | 372 |
| 755 | 777 | 871 | 766 | | 796 | 436 | 946 | 389 | 412 |
| 971 | 877 | 915 | 886 | 1086 | 635 | 543 | 831 | 458 | 395 |
| 728 | 819 | 829 | 832 | 873 | 666 | 537 | 918 | 447 | 466 |
| 737 | | 787 | 855 | 856 | 697 | 448 | 853 | 415 | 329 |
| 931 | 932 | 921 | 1858 | 828 | 627 | 449 | 785 | 486 | 395 |
| 795 | 1035 | 815 | 842 | 843 | 781 | 487 | 823 | 458 | 338 |
| 886 | 728 | 886 | 878 | 874 | 662 | 493 | 816 | 487 | 378 |
| 788 | 829 | 883 | 730 | 918 | 624 | 456 | 959 | 466 | 366 |
| 831.4 | 847.8 | 872 | 830.2 | 917.8 | 679.1 | 478.5 | 857.1 | 427.8 | 389.2 |




**الشكل(12)المخطط الخطي لعدد حزم التحكم.**

❖ **Total CPU (ticks):**

- الحالة الأولى شبكة تحتوي 20 عقدة موزعة بشكل منتظم:

**الجدول(6) متوسط زمن تشغيل المعالج لجميع الخوارزميات حالة شبكة تحوي 20عقدة.**

Total cpu (ticks)

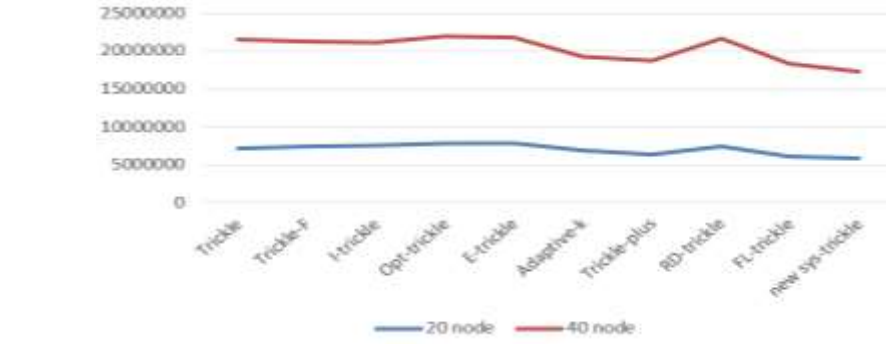
| Trickle | Trickle-F | I-trickle | Opt-trickle | E-trickle | Adaptive-k | Trickle-plus | RD-trickle | FL-TRICKLE | new syn-trickle |
|---|---|---|---|---|---|---|---|---|---|
| 7345945 | 6952532 | 7025586 | 7239631 | 7429804 | 6958365 | 5504659 | 7838214 | 5466091 | 5598756 |
| 7971606 | 7713851 | 8713367 | | 8292419 | 6366148 | 6041336 | 7602217 | 5740159 | 5152746 |
| 6988951 | 7868668 | 6521416 | 6712185 | 7747148 | 6918734 | 6818889 | 6821897 | 7531047 | 5163849 |
| 6915738 | 7875322 | 8153334 | 8878936 | 8483613 | 7524329 | 6951486 | 6731938 | 6657425 | 6077117 |
| 6877934 | 7869271 | 6872373 | 7881737 | 8047714 | 6479707 | 6102338 | 7277489 | 5988478 | 6083133 |
| 7542751 | 7579624 | 7588865 | 7897485 | 7518727 | 7000999 | 5830413 | 6827438 | 5798388 | 6574861 |
| 7526724 | 6962072 | 6416846 | 8653428 | 7875683 | 6871429 | | 7882106 | 6866996 | 5771887 |
| 7188442 | 7191783 | 7727335 | 7862395 | 8054008 | 6756744 | 5896255 | 7071861 | 5622298 | 5747467 |
| 7848568 | 7191168 | 8138371 | 7277621 | 8340414 | 6889558 | 6429553 | 6993798 | 6216423 | 6198614 |
| 7890345 | 7486845 | 7589467 | 6387198 | 8041788 | 6737116 | 6026696 | 7364755 | 5818777 | 5696281 |
| 7160180.4 | 7379104.8 | 7585887.2 | 7853312.6 | 7903139.8 | 6828311.3 | 6311282.7 | 7361010.5 | 6089864.8 | 5877511.1 |

- الحالة الثانية شبكة تحتوي 40 عقدة موزعة بشكل منتظم:

**الجدول(7) متوسط زمن تشغيل المعالج لجميع الخوارزميات حالة شبكة تحوي 40عقدة.**

Total cpu (ticks)

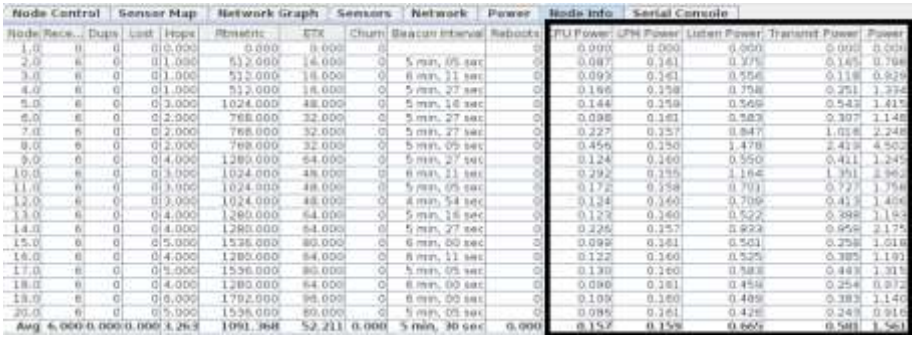
| Trickle | Trickle-F | I-trickle | Opt-trickle | E-trickle | Adaptive-k | Trickle-plus | RD-trickle | FL-TRICKLE | new syn-trickle |
|---|---|---|---|---|---|---|---|---|---|
| 23517403 | 19576845 | 22875640 | 20883128 | 22553150 | 20359388 | 17933422 | 22272881 | 17172979 | 18460127 |
| 23049982 | 21892411 | 21560461 | 19363458 | 22522990 | 17825284 | 17252774 | 20417211 | 16499228 | 16524545 |
| 21888834 | 19909682 | 22649196 | 21072393 | | 18625986 | 17797988 | 21960321 | 17644046 | 17639429 |
| 23874484 | 21841546 | 21883526 | 20529676 | 22888541 | 18486481 | 20116742 | 21617886 | 21228578 | 15559486 |
| 19719138 | 20699162 | 20499830 | 20963665 | 20903410 | 18276290 | 19833584 | 22725412 | 17544774 | 18789141 |
| 20139432 | | 19697381 | 22360024 | 21776705 | 23339714 | 19455967 | 20908387 | 19421735 | 16505887 |
| 23530947 | 24764771 | 19643342 | 27424898 | 19799886 | 18693492 | 17725867 | 20354101 | 17863671 | 16739733 |
| 21465372 | 23188667 | 21776572 | 23147446 | 22684888 | 19859319 | 19788474 | 21628866 | 20271831 | 15857085 |
| 19418695 | 18963855 | 20394449 | 23192598 | 22849893 | 18237213 | 19682172 | 20917117 | 19312415 | 20054444 |
| 19663391 | 21642255 | 20279529 | 18900918 | 21146964 | 19389834 | 18379439 | 22132806 | 17626753 | 16278869 |
| 21458878 | 21197621.5 | 21125999.2 | 21895819 | 21884481 | 19381492.5 | 18788622.9 | 21693338.8 | 18378128.2 | 17245386 |




**الشكل(13) المخطط الخطي لزمن تشغيل المعالج.**

❖ **Energy Consumption:**

لحساب استهلاك الطاقة نستخدم أداة Powertraceالمتوفرة في Contiki التي تقوم بتحديد ملامح الطاقة على مستوى الشبكة للشبكات اللاسلكية منخفضة الطاقة، حيث تقدر استهلاك الطاقة من قبل وحدة المعالجة المركزية، ونقل الحزم والاستماع . يتم ارسال هذه المعلومات إلى الأداة collect view التي تعرضه على الشكل التالي:




**الشكل(14) استهلاك الطاقة بالميلي واط لجميع العقد في الشبكة التي تحتوي 20 عقدة حالة خوارزمية Trickle الأصلية.**

كما في الشكل السابق فإنه ومن أجل كل خوارزمية ومن أجل شبكة بعدد عقد 20 وشبكة بعدد عقد40يوجد قيمة للطاقة Power سنرتبها في الجدول التالي :

**الجدول(10) متوسط استهلاك الطاقة لجميع الخوارزميات حالة شبكة تحوي 20 عقدة.**

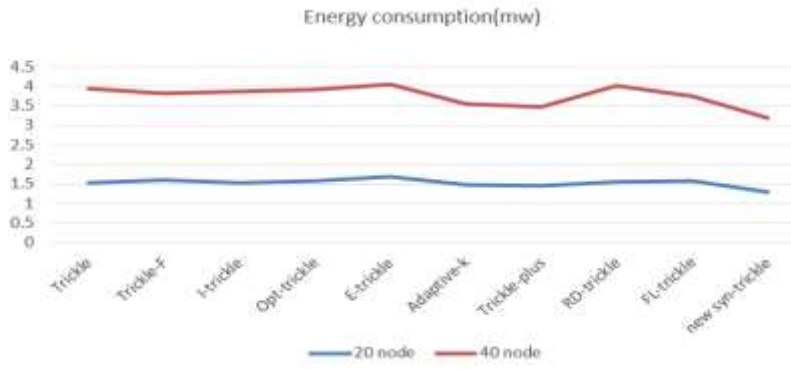
power consumption (mw)

| Trickle | Trickle-F | I-trickle | Opt-trickle | E-trickle | Adaptive-k | Trickle-plus | HD-trickle | FL-TRICKLE | new sys-trickle |
|---|---|---|---|---|---|---|---|---|---|
| 1.561 | 1.476 | 1.506 | 1.486 | 1.583 | 1.569 | 1.337 | 1.686 | 1.625 | 1.366 |
| 1.439 | 1.658 | 1.811 | 1.532 | 1.679 | 1.342 | 1.336 | 1.605 | 1.485 | 1.165 |
| 1.661 | 1.628 | 1.346 | 1.413 | 1.666 | 1.464 | 1.804 | 1.654 | 1.425 | 1.249 |
| 1.462 | 1.693 | 1.627 | 1.675 | 1.786 | 1.587 | 1.335 | 1.385 | 1.832 | 1.318 |
| 1.492 | | 1.506 | 1.743 | 1.773 | 1.431 | 1.487 | 1.451 | 1.372 | 1.383 |
| | | | | | | | | 1.747 | 1.351 |
| 1.523 | 1.61 | 1.527 | 1.569 | 1.697 | 1.478 | 1.443 | 1.556 | 1.58 | 1.292 |

**الجدول(11) متوسط استهلاك الطاقة لجميع الخوارزميات حالة شبكة تحوي 40 عقدة.**

power consumption(mw)

| Trickle | Trickle-F | I-trickle | Opt-trickle | E-trickle | Adaptive-k | Trickle-plus | RD-trickle | FL-TRICKLE | new sys-trickle |
|---|---|---|---|---|---|---|---|---|---|
| 2.546 | 2.106 | 2.398 | 2.388 | 2.202 | 2.091 | 2.019 | 2.610 | 2.456 | 1.926 |
| 2.474 | 2.296 | 2.248 | 2.248 | 2.549 | 2.129 | 1.916 | 2.443 | 1.888 | 1.961 |
| 2.295 | 2.311 | 2.298 | 2.196 | 2.402 | 1.979 | 2.328 | 2.258 | 1.992 | 1.775 |
| 2.261 | 2.163 | 2.487 | 2.529 | 2.489 | 1.913 | 1.969 | 2.591 | 2.458 | 1.856 |
| 2.474 | 2.278 | 2.284 | 2.462 | 2.168 | 2.219 | 1.968 | 2.357 | 1.997 | 1.969 |
| 2.41 | 2.22 | 2.337 | 2.347 | 2.344 | 2.066 | 2.04 | 2.451 | 2.158 | 1.897 |




**الشكل(15)المخطط الخطي لاستهلاك الطاقة.**

## 12. التوصيات:

قدمنا في هذا البحث خوارزمية تقطير جديدة متزامنة قادرة على إعطاء أقل زمن تقارب ممكن وذلك بسبب تزامن الفواصل الزمنية للعقد واختيارها فترة استماع فقط منطقية متغيرة مع حالة القمع لكل عقدة على حدى. بالإضافة على قدرة هذه الخوارزمية تخفيض العبء اللازم للتحكم بالتوجيه ببروتوكول RPL من خلال تلافي فواصل زمنية قد لا يكون للمرور فيها داعي أثناء التنفيذ والانتقال مباشرة إلى الطول الأعظمي للفاصل الزمني بعد ضمان تقارب الشبكة . كما أثبتت هذه الخوارزمية قدرتها تخفيض استهلاك الطاقة مقارنة بنظرائها من الخوارزميات لاستخدامها قيمة k تتناسب مع الجوار وبالتالي تقليل تكرار رسائل التحكم في الشبكة وتعتبر هذه الخوارزمية غير مستهلكة للمعالج بما يتناسب مع طبيعة العقد في شبكات الطاقة المنخفضة والخسارة التي تتميز بمواردها المحدودة .

## 13. المراجع:

[1] Djamaa, B., & Richardson, M. (2015). Optimizing the Trickle Algorithm, *13*(9), 10–13.

[2] Zhao, M., Kumar, A., Han, P., Chong, J., & Lu, R. (2016). A comprehensive study of RPL and P2P-RPL routing protocols : Implementation , challenges and opportunities. Peer-to-Peer Networking and Applications, (October 2017).

[3] Bagula, A., & Pietrosemoli, E. (n.d.). Internet of Things IN 5 DAYS.

[4] Goyal, S., Chand, T., & Member, S. (2018). Improved Trickle Algorithm for Routing Protocol for Low Power and Lossy Networks, 18(5), 2178–2183.

[5] Benson, D. J. (2016). A Performance Study of RPL with Trickle Algorithm Variants.

[6] Lamaazi, H., Benamar, N., Kahili, N. E. L., & Taleb, T. (2019). FL-Trickle : New Enhancement of Trickle Algorithm for Low Power and Lossy Networks.

[7] Yassein, M. B.,Alnadi, A., & Bataineh,A. (2018). RANDOMIZED DYNAMIC TRICKLE TIMER ALGORITHM FOR INTERNET OF THINGS, 187–197.

[8] Idrees, A. K., & Witwit, A. J. H. (2018). A Comprehensive Review for {RPL} Routing Protocol in Low Power and Lossy Networks.

[9] Yassein, M. B., Hmeidi, I., Shehadeh, H., & Yaseen, W. B. (2015). Performance Evaluation of " Dynamic Double Trickle Timer Algorithm " in RPL for Internet of Things ( IoT ).

[10] Ghaleb, B., Al-dubai, A., Ekonomou, E., Paechter, B., & Qasem, M. (2016). Trickle-Plus : Elastic Trickle Algorithm for Low- Power Networks and Internet of Things, (MEIoT).

[11] Vallati, C., & Mingozzi, E. (2013). Trickle-F : fair broadcast suppression to improve energy-efficient route formation with the RPL routing protocol.

[12] Yassein, M. B., Mohammad, R., Masadeh, T., World, T., & Science, I. (2017). A New Dynamic Trickle Algorithm for Low Power and Lossy Networks, (September 2016).

[13] Ph . D , Performance Evaluation and Improvement of the RPl Protocol. (n.d.).

[14] Ghaleb, B., Al-dubai, A., & Ekonomou, E. (2015). E-Trickle: Enhanced Trickle Algorithm for Low-Power and Lossy Networks,

[15] Meyfroyt, T. M. M., Stolikj, M., & Lukkien, J. J. (2015). Adaptive Broadcast Suppression for Trickle-Based Protocols, 0–8.

[16] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-level sensor network simulation with cooja," in Local computer networks, proceedings 2006 31st IEEE conference on, 2006, pp. 641–648.