مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (4) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (4) 2017

Building a cloud-based Internet of Things System using the CoAP protocol

Dr. Radwan Dandah*
Rasha Ghadeer**

(Received 22 / 6 / 2017. Accepted 9 / 8 / 2017)

□ ABSTRACT □

Internet of Things (IoT) and cloud computing are two very different technologies, Their massive adoption and usage is expected to increase further, making them important components of the future internet. Internet of Things consists of a set of things that are connected with each other using the internet in order to provide smart services to the end users. Generally, all of these things lack storage capacities and have limited processing power. Cloud computing on the other hand is a technology that has unlimited storage and processing capabilities, hence most of the problems plaguing the IoT can be solved, even partially, in the cloud.

In this paper we present a complete structure of an IoT system that can detect faces, recognize them, and predict age based on smart algorithms provided by the cloud using Constrained Application Protocol (CoAP) to communicate between nodes and cloud. This protocol is intended for real-time systems made up of small, low-cost IoT devices with limited resources. The performance of the server was tested by the CoAPBench tool.The results show that CoAP protocol reduces the burden on the IoT devices and improves the performance of cloud productivity by serving a large number of applications from connected devices.

Keywords: Internet of Things, Cloud Computing, Machine Learning, Constrained Application Protocol (CoAP) , Californium, Face Detection, Face Recognizer.

***Professor – Department of Computer Systems and Networks – Faculty of Information Engineering – Tishreen University – Lattakia – Syria.**
****** Postgraduate Student – Department of Computer Systems and Networks – Faculty of Information Engineering – Tishreen University – Lattakia – Syria.**

مجلة جامعة تشرين للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (39) العدد (4) 2017
Tishreen University Journal for Research and Scientific Studies - Engineering Sciences Series Vol. (39) No. (4) 2017

بناء نظام إنترنت الأشياء معتمد على السحابة باستخدام بروتوكول CoAP

الدكتور رضوان دنده *

رشا سمير غدير **

(تاريخ الإيداع 22 / 6 / 2017. قُبِل للنشر في 9/ 8 / 2017)

□ ملخّص □

إنترنت الأشياء (IoT) والحوسبة السحابية، تقنيتان مختلفتان تماماً، ولكنهما تعتبران جزءاً أساسياً من مستقبل الإنترنت. إنترنت الأشياء يعتمد على مجموعة من الأشياء تتصل مع بعضها البعض باستخدام الإنترنت لتقديم خدمات ذكية، ولكن كل هذه الأشياء تعاني من مشاكل التخزين وقدرات المعالجة المحدودة. من ناحية أخرى فإن الحوسبة السحابية تقنية أنضج بكثير من إنترنت الأشياء من حيث امتلاكها طاقات غير محدودة فيما يتعلق بالتخزين و قدرات المعالجة، و بالتالي فإن معظم المشاكل التي تعاني منها إنترنت الأشياء من الممكن حلها ولو بشكل جزئي في السحابة.

في هذا البحث نقدم بنية كاملة لنظام إنترنت الأشياء لتحديد وتمييز الوجوه ، وتوقع العمر بالاعتماد على الخوارزميات الذكية المقدمة من قبل السحابة. وباستخدام بروتوكول التطبيقات المقيدة (CoAP) للتواصل ما بين العقد والسحابة. هذا البروتوكول مخصص لأنظمة الزمن الحقيقي المتكونة من أجهزة IoT صغيرة الحجم ذات تكلفة قليلة و محدودة الموارد. تم اختبار انتاجية المخدم المنجز عن طريق الأداة CoAPBench حيث أظهرت النتائج أن بروتوكول CoAP يخفف العبء على عقد IOT كما أنه يحسن من أداء إنتاجية السحابة بتخديمها عدد كبير من الطلبات من الأجهزة المتصلة.

الكلمات المفتاحية: إنترنت الأشياء، الحوسبة السحابية، بروتوكول التطبيقات المقيدة، Californium، تعلم الآلة، التعرف على الوجه، تمييز الوجه.

* أستاذ – قسم النظم والشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة تشرين – اللاذقية – سورية.

**طالبة دراسات عليا (ماجستير)– قسم النظم والشبكات الحاسوبية– كلية الهندسة المعلوماتية– جامعة تشرين – اللاذقية – سورية.

مقدمة :

نتيجة انتشار الإنترنت يوماً بعد يوم و زيادة عدد الأجهزة الموصولة إلى هذه الشبكة، بالإضافة إلى طبيعة حياة الإنسان الحالية التي أصبحت مرتبطة بالإنترنت بشكل كبير إما من خلال عمله أو لاستخدامها في إدارة شؤونه الشخصية، ومع تطور الصناعات أصبحت جميع الأجهزة الالكترونية تمتلك أنظمة تشغيل بعد أن كانت تقتصر على المتحكمات، وأصبح بمقدورها جدولة مهامها والتواصل ليس مع الأجهزة الأخرى فقط بل مع الانسان لإدارتها أو إنذارها نتيجة حدوث حدث ما.

إن إنترنت الأشياء Internet Of Things أو اختصاراً IoT عبارة عن مجموعة من الأشياء التي تتواصل مع بعضها البعض لتقديم خدمات ذكية للإنسان. أما الأشياء في تعبير إنترنت الأشياء، هي وحدات فيزيائية تستطيع إعلام جهة معينة موصولة بالإنترنت عن هويتها وعن حالتها. يمكن اعتبار أي شيء يمكن ربط حساس معه (بقرة ترعى في حقل، أو حاوية في طائرة شحن، أو مكيف في مكتب، أو عمود إنارة في شارع) كعقدة في إنترنت الأشياء [1] .

الحساسات : هي المكونات التي تجمع وتنشر البيانات المختلفة، مثل: الموقع والارتفاع والسرعة والحرارة وشدة الإضاءة والحركة والطاقة والرطوبة وسكر الدم وجودة الهواء ورطوبة التربة، وغيرها الكثير. هذه العناصر ليست حواسيب، مع أنها قد تحوي نفس عناصر الحواسيب مثل المعالج والذاكرة والتخزين والإدخال والإخراج ونظام التشغيل والبرمجيات. ما يميزها أنها رخيصة الثمن، وتكلفتها تتناقص باستمرار، ومتوافرة بكثرة ، ويمكنها التخاطب إما مباشرة مع الإنترنت أو مع أجهزة متصلة بالإنترنت.

استهدفت العديد من الجهود البحثية تحديات البيئات المقيدة من حيث قدرات المعالجة والتخزين والبطارية مع موارد قليلة وروابط لاسلكية قابلة للضياع، حيث أن عقد IOT تقوم بتبادل رسائل تحكم ورسائل بيانات بين الحساسات في الوقت الحقيقي، وأغلب نوعية هذه الرسائل تكون ذو حجم صغير ولكنها رسائل كثيرة أو متعددة، إن طبيعة أجهزة IOT محدودة جداً وبالتالي هذه الأجهزة أو الحساسات غير قادرة على معالجة البيانات. من هنا كان السيناريو النموذجي هو استخدام خدمات السحابة (Cloud Services) لإدارة هذا العدد الكبير من الأجهزة، معالجة البيانات الخاصة بهم، وتنسيق النتائج [2].

من جهة أخرى العقد في إنترنت الاشياء بحاجة إلى بروتوكولات بعبء خفيف على العقد. إن البروتوكولات المتاحة من قبل فرقة العمل المعنية بهندسة الإنترنت Internet Engineering Task Force (IETF) غير مناسبة للعمل في بيئات مقيدة، فمثلاً استخدام بروتوكول HTTP مع TCP تملك العديد من المشاكل في بيئات مقيدة، مع وجود حجم اطار صغير وفقدان في الاتصالات اللاسلكية. بدلاً من معالجة المشاكل الموجودة مع http في البيئات السابقة، قامت IETF بتصميم بروتوكول ويب جديد يدعى بروتوكول التطبيقات المقيدة [3].

في هذا البحث قمنا ببناء سيناريو كامل لمجموعة من عقد إنترنت الأشياء تقوم بإرسال بياناتها باستخدام بروتوكول التطبيقات المقيدة (CoAP) و تجميع هذه البيانات في سحابة ومن ثم معالجتها بالاعتماد على الخدمات المقدمة من قبل السحابة، هذا سيوفر قابلية التعامل مع أعداد ضخمة من أجهزة إنترنت الأشياء المتصلة في نفس الوقت.

أهمية البحث وأهدافه :

نظراً لانخفاض تكاليف الوحدة ومعايير الإنترنت المفتوحة، يتوقع المحللون أكثر من 200 مليار جهاز إنترنت الأشياء بحلول نهاية عام 2020. وبالتالي نحن بحاجة إلى تقييم أداء البروتوكولات الخاصة بإنترنت الأشياء ، تأتي

أهمية هذا البحث في بناء نظام لإنترنت الأشياء معتمد على السحابة باستخدام بروتوكول CoAP عوضاً عن البروتوكولات الأخرى المستخدمة وتقييم أداءه. ويهدف البحث إلى إمكانية استبدال بروتوكولات الويب المستخدمة عادة ببروتوكول CoAP لاستخدامه في بيئات انترنت الأشياء المقيدة معتمدة على السحابة، وبالتالي تقليل العبء قدر الامكان على الأشياء وعلى السحابة.

طرائق البحث ومواده

يبدأ البحث بتعريف لبروتوكول التطبيقات المقيدة مع توضيح بنية عمله، كما أنه يقوم بتعريف العديد من المصطلحات التي تم استخدامها في البنية العملية كخدمات مايكروسوفت المعرفية أو الإدراكية Microsoft Cognitive Services و أيضاً الخدمة المقدمة من قبل شركة Google (Firebase Cloud Messaging) ، ومن ثم يتم شرح البنية العملية المنجزة .

1- بروتوكول التطبيقات المقيدة :

CoAP بروتوكول التطبيقات المقيدة اختصار لـ Constrained Application Protocol هو عبارة عن بروتوكول ويب تم تصميمه لأنظمة الزمن الحقيقي المتكونة من أجهزة IoT صغيرة الحجم – ذات تكلفة قليلة و محدودة الموارد ، يمكن أيضاً من تبادل رسائل بيانات الحساسات و رسائل التحكم كثيرة. يؤمن عمليات تبادل للرسائل غير المتزامنة باستخدام بروتوكول (User Datagram Protocol) UDP، CoAP يعتمد على نموذج مخدم/زبون مثل بروتوكول HTTP ويمثل عملية التفاعل بنفس الطريقة. يتم تعريف مجموعة من الموارد عن طريق تحديدها بمعرف الموارد المحدد (URIs) Uniform Resource Identifier باستخدام طرائق REST (Representational State Transfer) مثل GET , POST , PUT , DELETE أي (إحضار – إرسال – تعديل – حذف) مورد من مخدم [7]. كما أنه يتم الرد بإحدى أرقام الاستجابة والذي يعبر عن الاستجابة الصحيحة أو الخاطئة [3].

على عكس بروتوكول HTTP، بروتوكول CoAP يقوم بتبادل الرسائل بطريقة غير مباشرة عبر UDP . لتوفير آلية موثوقية خفيفة الوزن، يقوم باستخدام تقنية الحمل على الظهر و عمليات اعادة الارسال بالاعتماد على توقف – انتظر ، وأكثر من ذلك إنه يحقق الكشف عن الرسائل المتكررة باستخدام معرفات الرسائل الفريدة التي تم إنشاؤها عشوائياً [4].

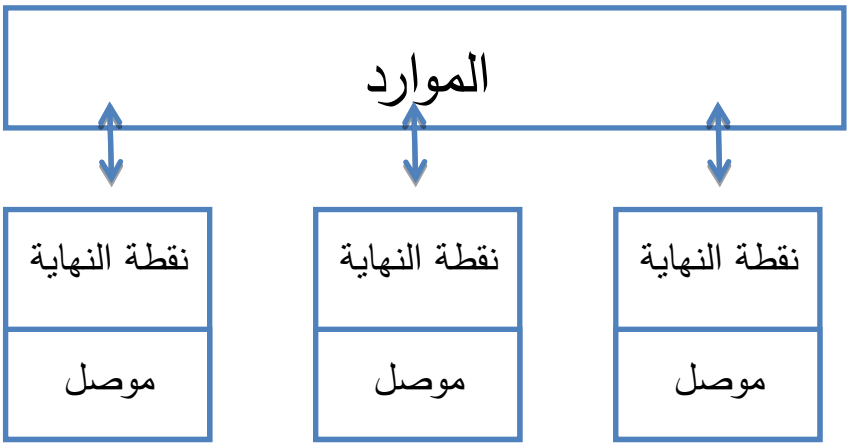
يوجد عدة تحقيقات لهذا البروتوكول منها Californium و nCoAP بلغة الجافا ، libcoap بلغة الـ C ، CoAP.NET بلغة #C، وأيضاً aiocoap ، CoAPthon بلغة Python وغيرها الكثير .

تم استخدام Californium في البنية العملية المنجزة في هذا البحث وتم اختياره لأن اللغة المستخدمة في التطبيق العملي هي الجافا كما أنه مخصص لبيئات إنترنت الأشياء مع الحوسبة السحابية، وبالتالي هو الأنسب للاستخدام [5] .

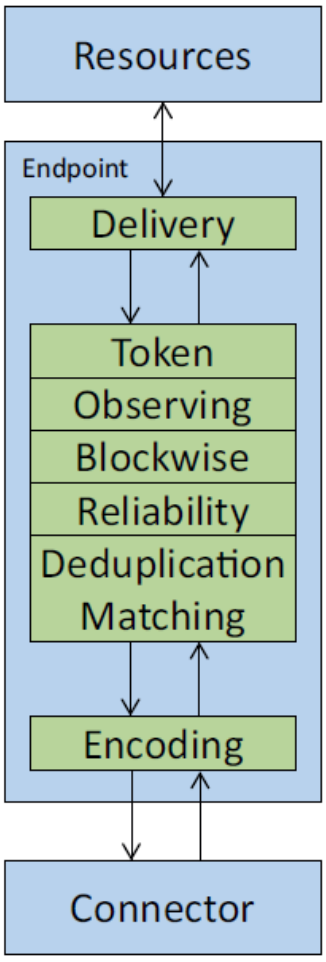
1-1- معمارية بروتوكول CoAP Architecture :

معمارية مخدم CoAP مقسمة إلى ثلاث مراحل : الموارد Resources ، نقاط النهاية EndPoints و موصلات Connectors. يظهر الشكل (1) كيف تعمل المراحل الثلاث السابقة مع بعضها البعض لتشكيل ما يدعى بمخدم Cf أو Californium Server ، يملك المخدم شجرة من الموارد، واحدة أو عدة نقاط نهاية، وكل نقطة نهاية

تملك موصل واحد فقط. أما بالنسبة للزبون فإنه سيعمل فقط مع نقاط النهاية و الموصلات . والآن سيتم شرح هذه المراحل الثلاثة بالتفصيل [6].



الشكل (1) : معمارية بروتوكول CoAP



الشكل (2) : مكدس نقطة النهاية

1-1-1- الموارد Resource :

الموارد هي البنية الأساسية التي سوف يستخدمها المطور لتصدير موارد التطبيق إلى العالم باستخدام بروتوكول CoAP. كل مورد CoAP يوفر واجهة RESTful إلى الزبائن، بحيث يكون هذا المورد قابل للوصول من قبلهم وقابل للتعديل عن طريق الاستجابة لأي من الأوامر الأربعة التالية (GET, POST, PUT, DELETE). كل مخدم يملك مجموعة من الموارد موزعة في بنية شجرية بحيث أن كل عقدة تمثل مورداً ، ويتم تمثيل كل عقدة عن طريق معرف URI مكون من الـ URI الخاص بالعقدة الأب بالإضافة إلى الاسم الخاص بالمورد. عندما يصل الطلب إلى المخدم يقوم بالبحث في شجرة الموارد عن المورد الذي يتطابق عنوانه مع URI الهدف المرفق مع الطلب. اذا وجد المخدم المورد يقوم بالاستجابة إلى الزبون مع رمز استجابة (Response Code). اذا لم يستطع المخدم إيجاد المورد الهدف فإنه يستجيب مع رمز الاستجابة المخصص للخطأ للدلالة على أن المورد المطلوب غير موجود [6] .

1-1-2- نقطة النهاية Endpoint :

يملك المخدم نقطة نهاية وحيدة أو أكثر من نقطة متصلة مع شجرة الموارد، تلتزم كل نقطة نهاية بتنفيذ بروتوكول CoAP أي أنه يحتوي على سلسلة من عمليات المعالجة والتي تقوم بمعالجة رسائل CoAP القادمة والصادرة. عندما يصل طلب CoAP جديد، تقوم نقطة النهاية بمعالجة الخطوات المطلوبة بالاعتماد على البروتوكول، على سبيل المثال فك تشفير الاطار إلى عدة كائنات ، الكشف عن

التكرار، معالجة الطلب. بنهاية عمليات المعالجة تقوم نقطة النهاية بتسليم الطلب إلى شجرة الموارد في المرحلة التالية، والتي تقوم بدورها بمعالجة الطلب والاستجابة على نفس نقطة النهاية. التطبيق يمكنه أيضاً استخدام نقطة النهاية لإرسال طلبات إلى مخدم CoAP آخر و استقبال الاستجابة فيما بعد [6].

في مرحلة نقطة النهاية يوجد العديد من المهام الواجب تحقيقها يتم توضحيها في الشكل (2) والذي يحتوي على عدة طبقات تشكل في مجموعها مكدس نقطة النهاية لبروتوكول CoAP وهي :

1- إدارة Token لتوليد ID من أجل كل طلب جديد .

2-مراقبة العلاقات Observe Relations من الزبائن وصولاً للموارد .

3-تحويل أو نقل Blockwise وذلك لتقسيم الرسالة الكبيرة إلى عدة رسائل صغيرة.

4-النقل بموثوقية Transfer Reliability من خلال إعادة ارسال الرسائل التي لم يتم الرد عليها.

5-مطابقة Matching الاستجابات الواردة و الإشعارات وتعيين كل الرسائل الصادرة السابقة ومقارنتها مع الجديدة لكشف التكرار .

6-تشفير الرسائل الصادرة إلى سلاسل من البايتات وفك تشفير الرسائل الواردة للحصول على تمثيل داخلي للرسالة.

1-1-3- الموصلات Connectors :

الموصل أو الرابط Connector مسؤول عن كيفية ارسال واستقبال المخدم للرسائل أي مسؤول عن بروتوكول طبقة النقل transport. تقوم نقطة النهاية بتمرير الرسائل المشفرة مع عنوان الهدف ورقم منفذ port إلى الموصل ليقوم بإرسالها عبر الشبكة. عندما يتلقى الموصل رسالة الاستجابة تكون أيضاً بشكل سلسلة من البتات تقوم بتمريرها إلى نقطة النهاية مع عنوان المصدر ومنفذه. بعض الموصلات تستخدم بروتوكول UDP وتقوم بإرسال واستقبال الحزم عبر المقابس socket . أما موصلات أخرى تقوم باستخدام بروتوكول طبقة النقل الآمن Datagram Transport Layer Security (DTLS).

الموصل يختلف عن المقبس socket. نقطة النهاية لا تقوم بالطلب من الموصل بتلقي رسائل جديدة، بدلاً من ذلك تقوم نقطة النهاية بإرسال الرسائل الجاهزة عبر الموصل، والعكس بالعكس يستدعي الموصل نقطة النهاية الخاصة به ويسلمها رسائل جديدة في حال ورودها إليه [6].

2- خدمات مايكروسوفت المعرفية أو الإدراكية Microsoft Cognitive Services :

هي عبارة عن مجموعة من واجهات برمجة التطبيقات APIs، وخدمات متاحة ومتوفرة للمبرمجين لجعل تطبيقاتهم أكثر ذكاءً، تندرج هذه الخدمات تحت جملة الخدمات التي تطورها مايكروسوفت فيما يخص تعلم الآلة (Machine Learning) وتمكن المطورين بسهولة من تضمين مميزات ذكية مثل التعرف على الوجه، العاطفة، وفهم الكلام واللغة إلى تطبيقاتهم. رؤية مايكروسوفت لبناء العديد من التجارب الحاسوبية وتحسين الخبرات وتعزيز الانتاجية وفعالية الأنظمة لنرى تطبيقات ترى ، تتكلم ، تفهم، وحتى تبدأ بالتفكير [8].

من أهم APIs المقدمة من قبل مايكروسوفت هو Face API : وهي عبارة عن خدمة معتمدة على السحابة تؤمن أهم الخدمات المتعلقة بالوجه : كشف الوجه (Face Detection) مع السمات وتمييز الوجوه (Face Recognition):

✓ كشف الوجه : تقوم بتحديد شخص أو أكثر في الصورة و إعادة احداثيات عن أماكن وجود هذه الوجوه في الصورة ، مع الكثير من خصائص الوجه المعتمدة على خوارزميات التعلم المتقدمة، من هذه الخصائص : العمر – الجنس – شكل الرأس – لون الشعر. وأكثر من 27 علامة مميزة في كل وجه في الصورة.

✓ تمييز الوجه: بعد رفع مجموعة من الصور لوجوه الأشخاص المطلوب التحقق من وجودهم في الصور ، يتم تدريب السحابة عليهم والحصول على صيغة معينة يتم المقارنة معها عند طلب الخدمة. تملك هذه الخدمة مجموعة من الخدمات الفرعية منها التحقق من الوجه (Face Verification) أي التحقق من احتمال وجود وجهان ينتميان إلى نفس الشخص بالاعتماد على قيمة ثقة confidence يتم حسابها بالاعتماد على خوارزميات ذكية مستخدمة في هذه التطبيقات المقدمة من شركة مايكروسوفت [8].

3- Firebase Cloud Messaging (FCM):

وهي عبارة عن خدمة مقدمة من شركة Google مطورة من الخدمة Google Cloud Messaging تتعامل مع إرسال الرسائل وتوجيهها وتخزينها بين تطبيقات المخدم وتطبيقات الزبون المتواجدة على الجهاز المحمول. إن FCM عبارة عن وسيط ما بين مرسلي الرسائل والزبائن. تطبيق الزبون هو تطبيق يعمل على جهاز يدعم FCM ، أما تطبيق المخدم هو عبارة عن مخدم يتواصل مع تطبيق الزبون عن طريق FCM .

باستخدام FCM يمكن لتطبيق المخدم أن يقوم بإرسال الرسائل أو الاشعارات إلى جهاز واحد، أو مجموعة من الأجهزة، أو إلى عدد من الأجهزة التي تشترك أو تهتم بموضوع محدد. أما تطبيق الزبون فإنه يقوم باستقبال الرسائل من تطبيق المخدم أي تلقي اشعارات عن بعد [9] .

النتائج والمناقشة :

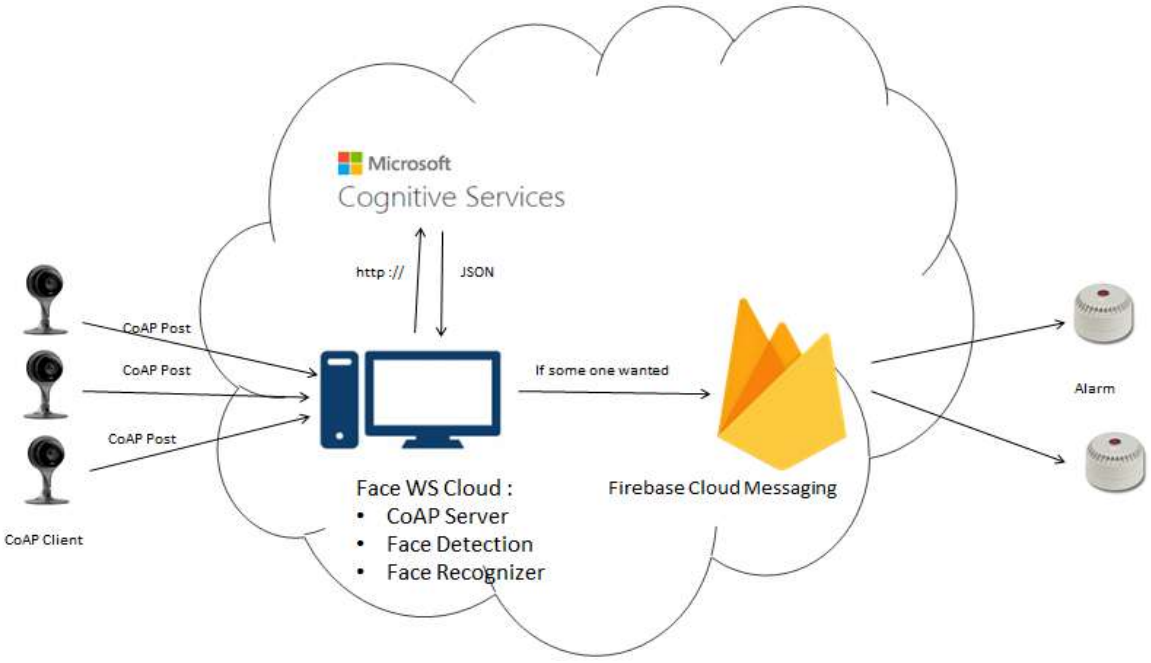
تم انجاز بنية عملية لإنترنت الأشياء مع الحوسبة السحابية موضحة في الشكل (3) .

في هذه البنية الأشياء : كاميرا – جهاز انذار.

أما السحابة تقدم الخدمتين التاليتين :

1- Detect : تحديد اذا وجد وجه في الصورة أو لم يوجد، و تحديد الجنس (ذكر- أنثى) و إعطاء رقم تخميني للعمر ، اذا كان الجنس ذكر وعمره بين قيمتين محددتين في التطبيق (في مثالنا محدد بالمجال 22- 40) يتم اصدار صافرة انذار .

2- Recognize : التعرف على الوجه و مقارنة الوجه مع مجموعة من الوجوه المدربة عليها السحابة مسبقاً ومن ثم اذا تم تمييز الشخص فإنه يتم اصدار صافرة انذار .



الشكل (3) : بنية IoT Cloud المنجزة

1- الزبائن CoAP Client :

بسبب عدم امتلاك كاميرا تم الاستفادة من الكاميرا المقدمة من قبل الجهاز المحمول دون الاستفادة من أي خدمات أخرى موجودة فيه ، وتم بناء تطبيق على نظام Android يوفر المطلوب.

تم بناء تطبيق موبايل سمي بـ CameraCoAP باستخدام برنامج Android Studio ، تم تضمين مكتبة بروتوكول CoAP المخصصة للغة الجافا والتي تدعى Californium [10,11]، يحتوي التطبيق على الواجهة كما في الشكل (4). يوجد في الواجهة زرين، زر التشغيل Start وزر الايقاف Stop ، ويوجد زر للإعدادات Setting.

الشكل (4) : تطبيق CameraCoAP

عند الضغط على زر Start يتم البدء بالتقاط الصور كل فترة زمنية محددة بقيمة المؤقت، بعد ذلك يقوم بتوليد نيسب (thread) مهمته توليد زبون CoAP Client يقوم بإرسال طلب Request باستخدام بروتوكول CoAP إلى مخدم CoAP Server له الشكل التالي :

POST URI Payload

- ✓ الطلب POST يعني أنه سيتم اضافة مورد محدد بالحمولة إلى المخدم .
- ✓ Uniform Resource Identifier أو معرف الموارد الموحد له الشكل: (coap://ServerIpAddress:ServerPortNumber/image) محددة عنوان المخدم و رقم المنفذ (الافتراضي هو 5683 في حال استخدام البروتوكول UDP و المنفذ 5684 في حال استخدام البروتوكول DTLS) .
- ✓ الحمولة Payload عبارة عن الصورة التي تم التقاطها بعد تحويلها إلى سلسلة من البايتات.
- ✓ يتم أيضاً ارسال نوع الخدمة المطلوبة (detect , recognize) وارسالها إلى السحابة .

2-السحابة FaceWS : تم انجاز السحابة باستخدام لغة java على بيئة التطوير Eclipse Neon . تتكون السحابة من المكونين التاليين:

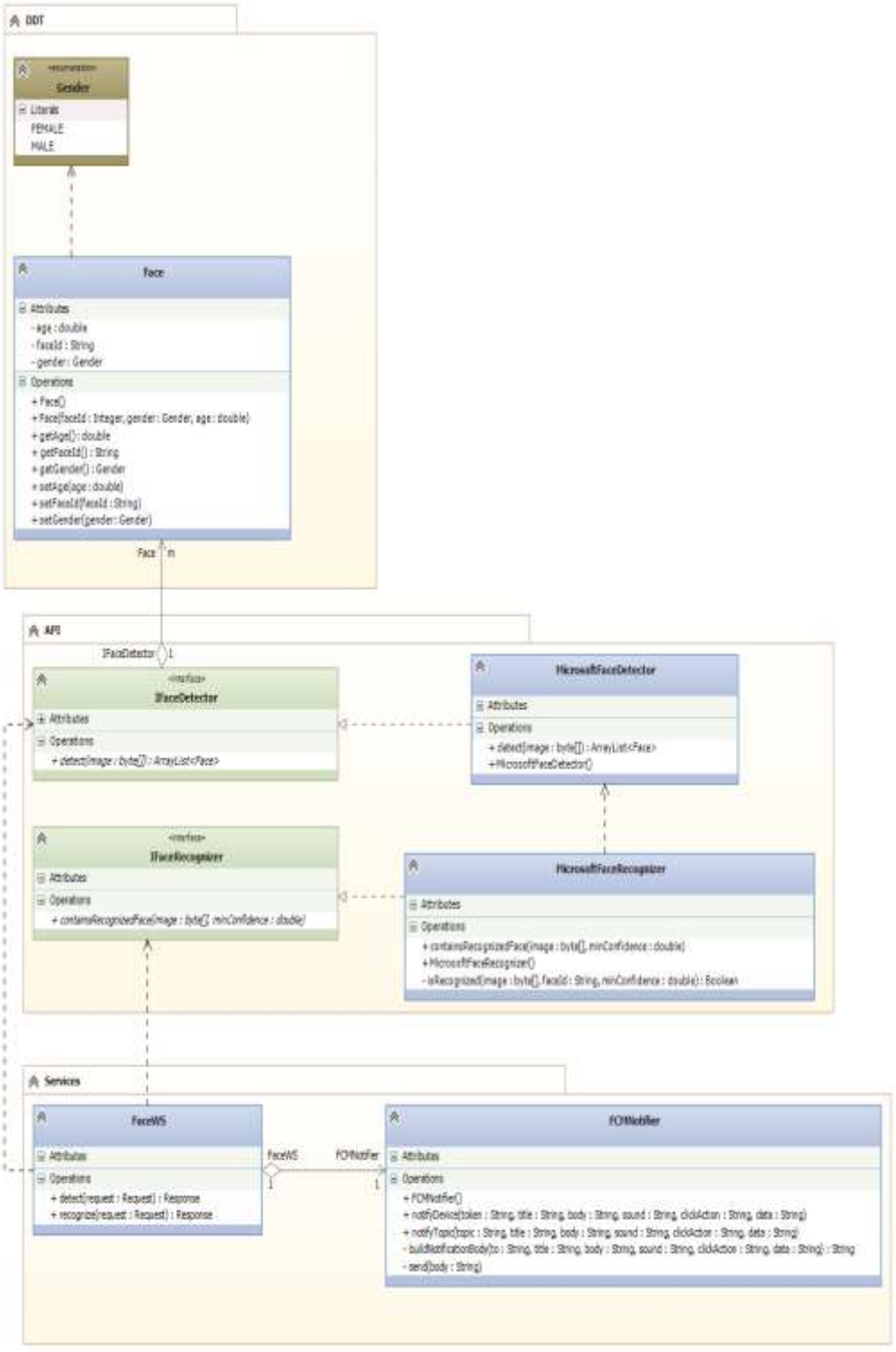
2-1- مخدم CoAP : بعد تضمين مكتبة بروتوكول CoAP المخصصة للغة الجافا والتي تدعى Californium [10,11] إلى التطبيق. تم انشاء مخدم CoAP وتهيئته بالقيم المناسبة ومن ثم اضافة مورد إلى السيرفر يدعىimage، هذا المورد يوفر الطرائق GET, POST, PUT,DELETE (إحضار مورد من المخدم ، إرسال مورد إلى المخدم ، حذف مورد من المخدم ، تعديل مورد في المخدم) للزبائن التي تقوم بإرسال الطلبات.

في تطبيق الأندرويد CameraCoAP السابق تم استخدام الطريقة POST مع المورد image والتي تقوم باستقبال سلسلة من البايتات كحمولة مع الطريقة POST والتي تعبر عن الصورة واستقبال نوع الخدمة (اذا كانت detect يتم تمرير الصورة إلى الخدمة Face Detector – اذا كانت recognize يتم تمرير الصورة إلى الخدمة Face Recognizer).

2-2- تحديد الوجوه وتمييز الوجوه :

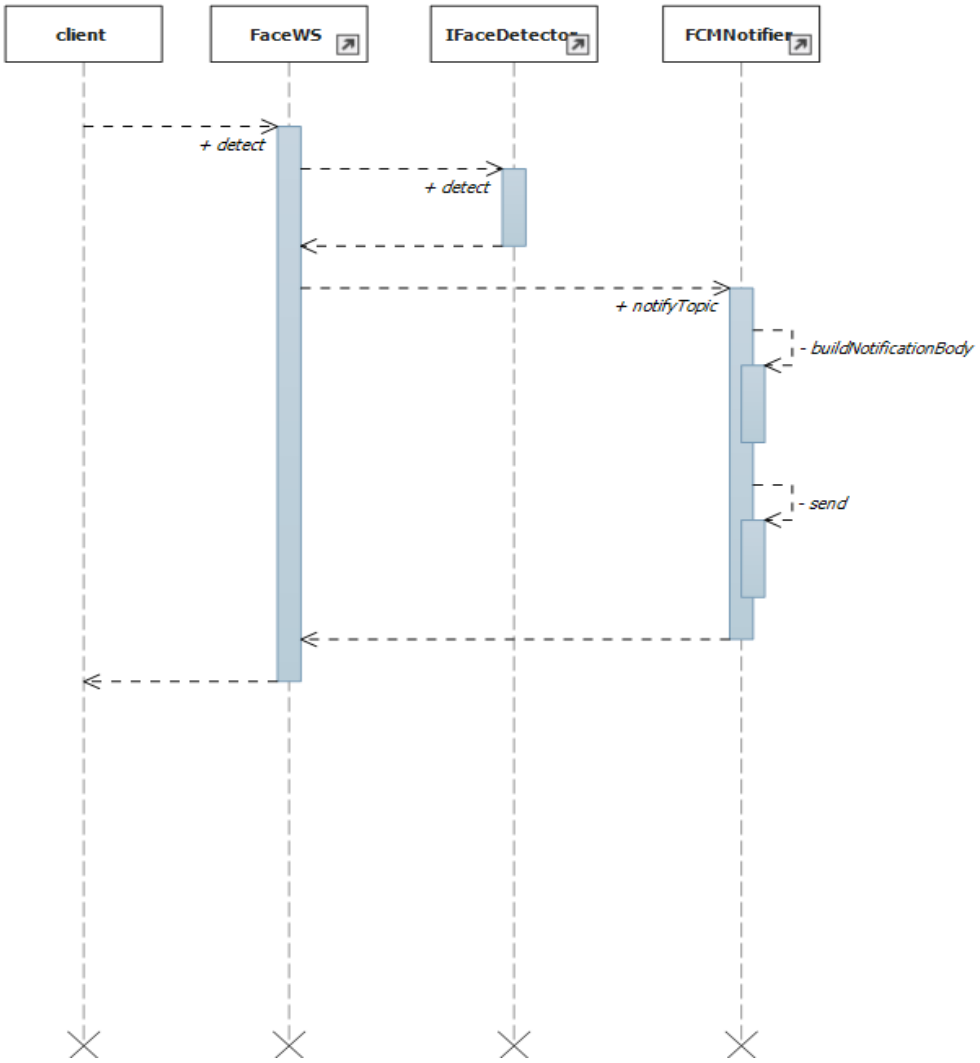
يمثل الشكل (5) مخطط الصفوف (Class Diagram) الخاص بخدمة تحديد و تمييز الوجوه في السحابة.

تم تحقيق كل من الواجهات IFaceDetector و IFaceRecognizer باستخدام ال API المتاحة من شركة مايكروسوفت (Microsoft Cognitive Programming) [8]، حالياً مع امكانية التعديل مستقبلاً لتوفير تحقيقات أكثر مرونة و سرعة في معالجة الصورة.



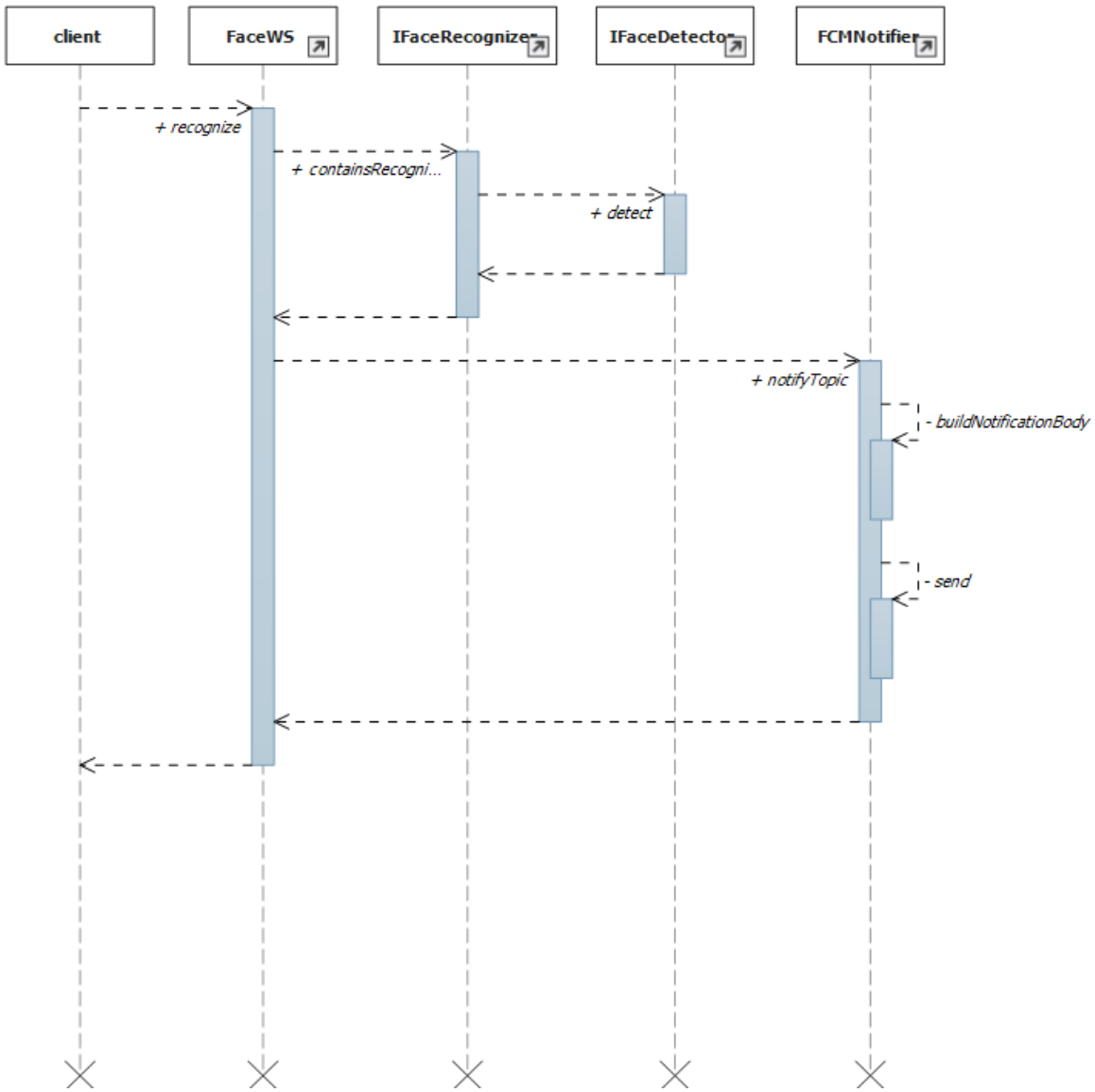
الشكل (5) : مخطط الصفوف

يمثل الشكل (6) المخطط التسلسلي (Sequence Diagram) لخدمة Face Detector التي توفرها السحابة.



الشكل (6) : المخطط التسلسلي لخدمة Face Detector

يمثل الشكل (7) المخطط التسلسلي (Sequence Diagram) لخدمة Face Recognize التي توفرها السحابة .



الشكل (7) : المخطط التسلسلي لخدمة
Face Recognize

2-3- جهاز الانذار :

بسبب عدم امتلاك جهاز انذار لاستخدامه في التطبيق العملي تم بناء تطبيق أندرويد يدعى بـIOTFaceWSAlarm تظهر واجهته في الشكل (8). عند تنصيب التطبيق على الجهاز المحمول يقوم بتسجيل نفسه في قاعدة بيانات تدعى Alarm منشأة باستخدام تقنية [9] Firebase Cloud Messaging. تقوم السحابة بعد معالجة الصورة بتحديد و تمييز الوجوه و من ثم ووفقاً للخدمة المطلوبة وعند توفر الشروط المناسبة ترسل الى FCM طلب لإرسال الاشعارات إلى كل الأجهزة المسجلة والتي تقوم بدورها بإصدار صوت صفارة إنذار .

الشكل (8) : واجهة تطبيق IOTFaceWSAlarm

3- تقييم الأداء :

لا توجد أداة مخصصة لبروتوكول CoAP لقياس انتاجية النظام السابق المنجز بالكامل، ومن أجل توقع جودة الانتاجية بالنظام المبني تم اجراء اختبار لإنتاجية مخدم CoAP فقط باستخدام الأداة CoAPBench [12]، تعبر الإنتاجية (throughput) عن معدل عدد الطلبات التي تم تخديمها بنجاح في الثانية .

CoAPBench هي أداة قياس أداء مخدمات CoAP. يتم اغراق المخدم بعدد من الطلبات من زبائن افتراضيين في الوقت نفسه لقياس قدرة المخدم على تخديم الطلبات. يمكن تنفيذ الـ CoAPBench على عدة أجهزة . يوجد عقدة زبون رئيسية Client master تقوم بالتحكم بعدد من عقد الزبائن الثانويين Client Slave بإرسال رسائل تحكم TCP مرتبطين بوصلات فيزيائية. تقوم العقدة الرئيسية في الوقت ذاته بإرسال اشارة بداية إلى كل العقد الثانوية ، متضمنة عدد الزبائن الافتراضيين (concurrent c) والتي من المفترض أن تقوم بإرسال طلبات متزامنة في الوقت نفسه. كل عقدة ثانوية تقوم بإرسال طلبات لمدة 60 ثانية وتقوم الأداة بحساب عدد الاستجابات من المخدم، وحساب عدد الطلبات التي تم تخديمها وتخزينها داخل قيمة received و عدد الطلبات التي لم يتم تخديمها وتخزينها داخل قيمة timeouts. مجموع كل الاستجابات لكل الزبائن الافتراضيين مقسمة على الفترة الزمنية (60 ثانية) تعطي معدل الانتاجية التي يستطيع المخدم تحقيقها ويتم تخزين قيمتها بداخل throughput. اذا تم فقدان رزم UDP و لم يحصل الزبون الافتراضي على الاستجابة بعد مرور 10 ثواني من عدم الاستجابة ، يقوم الزبون الافتراضي باعتبار الطلب أنه فاشل ، ويقوم بإرسال طلب جديد[12].

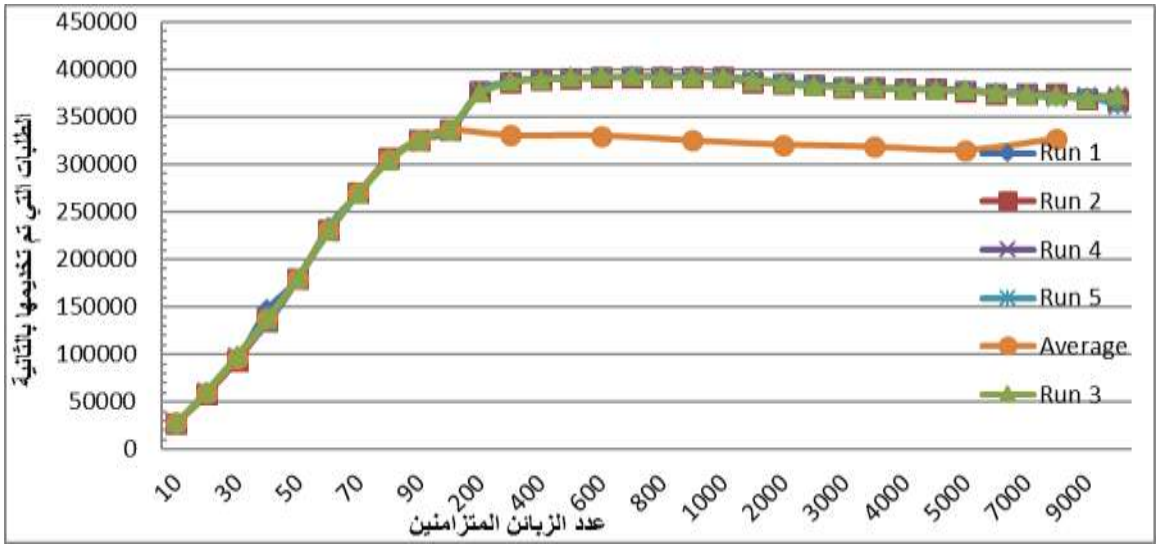
تتكون البيئة العملية من السحابة السابقة و مخدم CoAP موجود على حاسوب بالمواصفات (Windows 7 64-bits Intel Core i7 – 8GB RAM) ، يتم توليد زبون رئيسي يقوم بإرسال طلبات إلى ثلاث زبائن ثانويين، يتم تحديد عدد الزبائن المتزامنين و المدة الزمنية قبل البدء بالقياس. تم تقييم أداء سيناريوهين :

السيناريو الأول :

تم قياس أداء مخدم CoAP مع عدد مختلف من الزبائن الافتراضيين. تم البدء بإعطاء قيمة C لمستوى التزامن من 10 وزيادتها حتى الوصول إلى 10000، وبالتالي أخذت قيمة عدد الزبائن المتزامنين القيم التالية في التجربة :

C=10,20,30,40,50,60,70,80,90,100,200,300,400,500,600,700,800,900,1000,1500,
2000,3000,4000,5000,6000,7000,8000,9000,10000

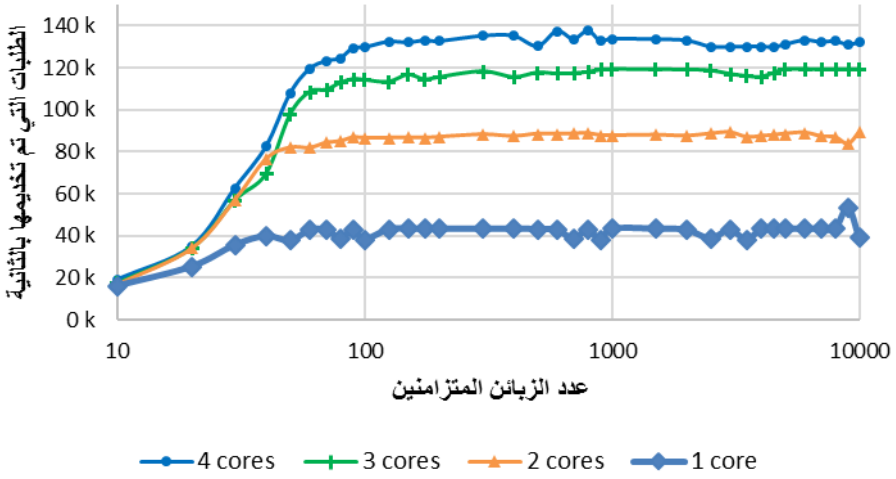
تم اجراء كل تجربة خمسة مرات لمدة 60 ثانية من أجل قيم C السابقة، ومن ثم تم حساب المعدل average من أجل كل قيمة. ومن ثم تم رسم التجارب الخمس مع المعدل كما في الشكل (9)، نلاحظ من المخطط قدرة مخدم CoAP على تخديم عدد كبير من الزبائن في نفس الوقت وبالتالي هذا يعطي مرونة كبيرة للسحابة و يخفف العبء بشكل كبير على عكس البروتوكولات التالية.



الشكل (9) : تقييم أداء مخدم CoAP

السيناريو الثاني :

تم اجراء التجارب من أجل مخدم بنوى متعددة (. 1 core و 2core و 3 core و 4 core) وتم حساب عدد الطلبات التي تم تخديمها ورسم النتائج كما في الشكل (10)، وبالتالي نرى أن مخدم CoAP يتم تحسن أدائه بشكل ملحوظ عند زيادة عدد cores لأن بروتوكول الـ CoAP منجز للقيام بالمهام على عدة نوى والاستفادة من معمارية المعالج وذلك بسبب خاصية التزامن المتواجدة فيه. نلاحظ في المخطط أن Californium حقق انتاجية عظمى لـ 140,000 طلب بالثانية. و أن انتاجيته بحلول 4 نوى أعلى بحوالي 3.5 مرة من نواة وحدة.



الشكل (10) : تقييم أداء مخدم CoAP بالنسبة لعدد النوى

الاستنتاجات والتوصيات:

في هذا البحث تم تجهيز بيئة عملية يمكن من خلالها بناء الكثير من تطبيقات الزمن الحقيقي ، مثلاً وضع كاميرا أمام المسافرين الذين سيصعدون إلى الطائرة و مراقبة وجوه الداخلين، في حال تطابق الوجه لأحد المارة مع أحد المطلوبين فإنه يقوم بإصدار صفارة انذار في مكان آخر. أيضاً يمكن تثبيت الكاميرا أمام حاجز للسيارات و في حال مرور ذكر وتخمين عمره، اذا كان بين حدين معينين يقوم بإصدار صفارة انذار في مكان آخر، وغيرها من السيناريوهات الاخرى المفيدة في الزمن الحقيقي.

كما تم بناء نظام إنترنت الاشياء و ارسال البيانات و تجميعها ومعالجتها بداخل السحابة ، و تم استخدام بروتوكول التطبيقات المقيدة CoAP عوضاً عن البروتوكولات المستخدمة عادة في تطبيقات الويب، وقد أظهرت النتائج السابقة قدرة مخدم CoAP على استقبال وتخديم عدد كبير من الطلبات خلال واحدة الزمن.

يمكن استخدام البنية المنجزة للمقارنة بين العديد من البروتوكولات المستخدمة في أنظمة إنترنت الأشياء مع السحابة مثل Hypertext Transfer Protocol (HTTP) ، Message Queue Telemetry Transport (MQTT) بالإضافة إلى بروتوكول التطبيقات المقيدة لاقتراح البروتوكول الأنسب مع بيئات إنترنت الأشياء المستقبلية.

المراجع :

[1] ROSE, K. ; ELDRIDGE, S. ; CHAPIN, L. *The internet of things: An overview.* The Internet Society (ISOC).Oct. 2015, (pp. 1-50).

[2] BOTTA, A. ; DE DONATO, W. ; PERSICO, V. ; PESCAPE, A. *On the integration of cloud computing and internet of things*. In Future Internet of Things and Cloud (FiCloud), 2014 International Conference on Barcelona, Spain, IEEE , Aug. 2014, (pp. 23-30).

[3] JOSHI, M. ; KAUR, B.P. , *CoAP Protocol for Constrained Networks.* published in IJ Wireless and Microwave Technologies, vol.5, no.6, NOV. 2015, (pp.1-10).

[4] SHELBY, Z. ; HARTKE, K. ; BORMANN, C. *The constrained application protocol (CoAP).* Internet Engineering Task Force (IETF), ISSN: 2070-1721, June. 2014, < https://tools.ietf.org/html/rfc7252>.

[5] PAULI, D. ; OBERSTEG, D. *Californium Documentation*. ETH Zurich, Spring. 2010, pp. 29, < https://people.inf.ethz.ch/mkovatsc/resources/californium/cf-thesis.pdf >.

[6] Lanter, M. *Scalability for IoT Cloud Services*, ETH Zurich, Oct. 2013, pp. 82.

[7] Representational state transfer , 2Mar. 2017. <https://en.wikipedia.org/wiki/Representational_state_transfer>.

[8] Microsoft Azure , 10Mar. 2017. < https://azure.microsoft.com/en-us/services/cognitive-services/face/ >.

[9] Google Firebase Cloud Messaging, 15Mar. 2017. <https://firebase.google.com/docs/cloud-messaging/>.

[10] The Eclipse Foundation, 23Mar. 2017. <https://eclipse.org/californium/>.

[11] Eclipse Californium, 20April. 2017. <https://github.com/eclipse/californium>.

[12] Californium Tools, Eclipse, github, 13May. 2017.
< https://github.com/eclipse/californium.tools>.