الجمهورية العربية السورية

جامعة البعث

كلية الهندسة المعلوماتية

قسم هندسة النظم والشبكات الحاسوبية

تصميم نظام معتمد على انترنت الأشياء وتحقيقه وتقييمه لمراقبة بعض المؤشرات الحيوية للمرضى عن بعد

دراسة أعدت لنيل درجة الماجستير في هندسة النظم والشبكات الحاسوبية

إعداد:

المهندس اسامه علي إبراهيم

إشراف:

الدكتورة زينب خلَّوف

أستاذ مساعد في قسم هندسة النظم والشبكات الحاسوبية

جامعة البعث

الدكتور المهندس سهيل الحمود

مدرس في قسم هندسة البرمجيات ونظم المعلومات

جامعة البعث

1440هـ -2019 م

كلمــة شـــكر وتقــدير

لطــالما كانت ســـــطور الشـــكر صــعبة الصــياغة.....

ربما لأنها تشعرنا دوماً، بقصورها وعدم إيفائها حق من نشكره ...

تتسابق كلماتي وتتزاحم عباراتي لتنتظم في عقد شكرٍ لكلّ من علّــمني، ومسـح غيمة جـهلٍ مررت بهـا برياح العـلم الطـيّبة...

لمن كان له السبق في مساعدتي ولم يدّخر جهداً في سبيل نجاحي، لمن قَوم أخطائي وصـحّح عثراتي دكـاتـرتي الأفاضــل في كليّتي العزيزة أفضل ما أقدمه لكم هو دعاء الله لكم بالتوفيق والصحة والسعادة وأن يبارك فيكم إنه ولي ذلك والقادر عليه.

<u>وأخصّ بالشكر والتقدير</u>

الدكتورة زينب خلَّوف و الدكتور سهيل الحمود

اللذين تفضلا بالإشراف على إنجاز هذه الرسالة ولم يبخلا عليّ بإرشاداتهم وتوجيهاتهم الحكيمة شـــكراً من القـــلب.

م. اسامه إبراهيم

إهدائي الخاص

لسيد الوطن .. . القائد المفدى بشار الأسد الأب و القدوة و المثل .. .

للسيدة الأولى .. . سيدة الياسمين و الإنسانية التي تدعم الجرحى و تركز بشكل خاص على الجانب العلمي .. .

إلى سندي في أصعب و أقسى أيام حياتي .. . ملهمتي و سر نجاحي في مرحلة ما بعد الإصابة .. تلك التي تشعرني بالشجاعة و استمد منها اقدامي .. . أختي آلاء

وأوجه أسمى آيات الشكر والعرفان إلى الكادر التدريسي والإداري في كلية الهندسة المعلوماتية وأخص بالذكر:

الدكتور أكرم مرعي رئيس قسم هندسة النظم والشبكات الحاسوبية

أيضاً إلى الدكتور ماهر عباس الرائع

إلى الأستاذة منال زعرور التي كانت مثالاً للأخت المميزة، فساعدتني و يسرت أموري في كل الجوانب الإدارية، فشكراً أستاذة منال. ..

إلى الأمانة السورية للتنمية – برنامج جريح الوطن الذي قدم الدعم اللوجستي لكل نشاطاتي من مؤتمرات ومعارض علمية وأخص بالذكر الأخت والصديقة بشرى الحسين التي كانت مثالاً للجد و المتابعة الدائمة .. .

و إلى كل شخص حاربني أو استخف بقدراتي خلال مسيرتي، فزاد إصراري للوصول، وكان حافزاً إضافياً لنجاحي. ..

و أخيراً إلى كل القيود و السلاسل التي علمتني أن أقاتل بصبر .. .

م. اسامه إبراهيم

المشاركات و الجوائز و الإنتاج العلمي

- حاصل على قبول نشر من مجلة جامعة البعث بتاريخ 26/9/2019 في المجلد 41 عن بحث بعنوان:
  - **تصميم نظام معتمد على انترنت الأشياء وتحقيقه وتقييمه لمراقبة بعض المؤشرات الحيوية للمرضى عن بعد**
- حاصل على المركز الثاني في معرض الباسل للإبداع و الاختراع لعام 2018 عن عمله بعنوان:
  - **تصميم وتحقيق نظام معتمد على انترنت الأشياء لمراقبة بعض المؤشرات الحيوية للمرضى عن بعد**
- حاصل على الميدالية الذهبية في معرض الباسل للإبداع و الاختراع لعام 2019 عن عمله بعنوان:
  - **تصميم نظام المتابعة المستمر لمرضى الحالات المزمنة عن بعد عبر تطبيق أندرويد على الهواتف الذكية وتحقيقه وتقييمه**

(و هذا العمل هو النسخة المطورة عن النظام و مزودة بتجهيزات حقيقية بسيطة)

- شارك في الندوة العلمية المقامة في كلية الهندسة المعلوماتية – جامعة البعث لعام 2018 بعنوان:
  - **آفاق البحث العلمي في المعلوماتية**
- شارك في مؤتمر الباحثين السوريين المغتربين الأول لعام 2019 بعنوان :
  - **نحو اقتصاد المعرفة لسورية ما بعد الحرب: دور الباحثين السوريين في الوطن والمغترب**

الملخص

يولّد قطاع الرعاية الصحية كميّة ضخمة من البيانات الطبيّة المتعلقّة بالمرضى، ولذلك ركزّت التوجهات الحديثة على جمع هذه البيانات، لتتم الاستفادة منها في دعم اتخاذ القرارات السريرية، والتحليلات التنبؤية لسلامة المريض؛ بهدف تحسين أنماط تقديم الرعاية الصحية في عالم بات يحث الخطى باتجاه "مدن ذكيّة" تُقيّم بمدى جودة الخدمات والرعايّة المقدمّة لمواطنيها.

يقدّم هذا البحث تصميم وتحقيق بنيّة لنظام مراقبة عن بعد يستند إلى انترنت الأشياء لتجميع ومعالجة البيانات الضخمة الناتجة عن مراقبة بعض المؤشرات الحيوية للمرضى في الزمن الحقيقي، كما يعرضُ تقييماً لبروتوكولي التراسل MQTT و AMQP المسؤولين عن إيصال البيانات من أجهزة المراقبة إلى البنية، و أيضاً يقدّم مقترحاً لحل مسألة الثغرة الأمنية التي تعاني منها تقنية انترنت الأشياء باعتبار شروط البنية المحققة، و تحليلاً بسيطاً للمسألة بعد تحقيق ذلك المقترح، و أخيراً يقدّم مناقشة نظرية للجدوى من استخدام Kafka Apache في مسألة تحسين أداء النظام.

الكلمات المفتاحية: رعاية صحيّة، انترنت الأشياء، معالجة البيانات الضخمة، برتوكولات التراسل، RabbitMQ، MQTT، AMQP، Apache Storm .

Abstract

The health care sector generates a huge amount of patients-related medical data, so recent trends have focused on collecting this data to be used to support clinical decision-making and predictive patient safety analyzes to improve health care delivery patterns in a world that is becoming pushes the pace towards "smart cities" this cities are evaluated by the quality of services and care provided to its citizens.

This research presents designing and implementing an Internet of Things-based remote Monitoring system to collect and process Big data generated by monitoring some vital signs of the Patients in real time. It also presents an evaluating the MQTT and AMQP protocols responsible for the delivery of data from monitoring devices to the structure. It also presents a proposal to resolve the vulnerability of the IoT technology considering the conditions of the structure achieved, a simple analysis of the issue after Implementing this proposal, and finally provides a theoretical discussion of the feasibility of using Apache Kafka in the issue of improving the performance of the system.

KEYWORDS: Health Care, Internet of Things, Big Data Processing, Messaging protocols, RabbitMQ, MQTT, AMQP, Apache Storm.

الفهرس

الملخص..... I

Abstract.....III

الفهرس.....V

قائمة الأشكال والرسومات البيانية.....X

قائمة الجداول..... XIV

قائمة المختصرات..... XVI

الفصل الأول (المقدمة وأهداف البحث) 1

1-1 مقدمة 1

1-2 مشكلة البحث.....1

1-3 أهمية البحث.....1

1-4 أهداف البحث..... 2

1- 5 الجهات التي ستستفيد من هذا البحث.....2

1- 6الصحة الإلكترونية.....2

1-7 أهم تطبيقات الصحة الإلكترونية.....2

1-8 انترنت الأشياء.....3

1-9 البيانات الضخمة.....3

1- 10خصائص البيانات الضخمة.....3

1-11عتاد البيانات الضخمة.....4

الفصل الثاني (الدراسات المرجعية) 5

2 – 1 مقدمة 5
2–2 الهدف من الدراسة المرجعية..... 5
2–3 الدراسات التي قدمت مقترحاً لنظام صحة عن بعد..... 5
2–4 الدراسات التي قدمت نُهُجاً وتجارباً سابقة و أفكاراً نظرية..... 6
2–5 الدراسات التي قدمت تقييماً لبروتوكولات التراسل وفقاً لشروط وبيئات معينة..... 7
2–6 الجديد الذي يقدمه هذا البحث 8
الفصل الثالث (نظرة عامة عن مكونات البنية المقترحة) 10
3–1 مقدمة..... 10
3–2 بروتوكولات التراسل..... 10
3–2–1 البروتوكول MQTT..... 10
3–2–2 مبدأ عمل MQTT..... 11
3–2–3 تعليل إضافة البروتوكول MQTT للبنية..... 11
3–2–4– البروتوكول AMQP..... 11
3–2 –5 مبدأ عملAMQP 11
3–2–6 تعليل إضافة البروتوكول AMQP للبنية..... 12
3–3 وسيط الرسائل RabbitMQ..... 12
3–4 معالج البيانات الضخمة Apache Storm..... 13
3–4– 1أهم ميزات Apache Storm..... 13
3–4–2 بنية العنقود في Apache Storm..... 13
3–4–3 بنية الطوبولوجيا في Apache Storm..... 14
3–5 نظام التخزين Apache Cassandra 16
3–6 نظام التخزين Redis..... 17
3–7– الأداة Node–Red..... 18

الفصل الرابع (تصميم وتحقيق نظام لمراقبة المؤشرات الحيوية للمرضى عن بعد معتمد على انترنت الأشياء)... 19
4-1 مقدمة..... 19
4-2 البنية المقترحة لمراقبة المؤشرات الحيوية..... 19
4-2-1 المنتج (Publisher) producer..... 19
4-2-2 بروتوكول التراسلMessaging Protocol..... 19
4-2-3 وسيط الرسائلMessage Broker..... 20
4-2-4 معالج البيانات الضخمة Apache Storm..... 20
4-2-5 نظام التخزين المؤقت للبيانات Redis..... 20
4-2-6 نظام تخزين البيانات التاريخية Apache Cassandra..... 20
4-2-7 أداة عرض البيانات في الزمن الحقيقي Node-Red..... 20
4-3 منطق المعالجة ضمن Storm Topology..... 21
4-3-1 عملية تصفية البيانات..... 21
4-3-2 عملية التمييز بين المؤشرات من حيث نوعها 21
4-3-3 عملية إضافة الوسمة الزمنية أو الطابع الزمني Time Stamp للمؤشر الحيوي 21
4-3-4 عملية تخزين بيانات المريض..... 21
4-4 التمثيل التقني للطوبولوجيا Storm Topology..... 22
4-5 التحقيق الفعلي للبنية المقترحة 23
4-5-1 تحقيق المنتج..... 23
4-5-2 تنصيب و إعداد وسيط الرسائل 24
4-5-3 تنصيب و إعداد معالج البيانات الضخمة Storm Apache..... 26
4-5-4 تحقيق الطوبولوجيا في Storm ومكوناتها..... 28
4-5-5 تحقيق تخزين البيانات الموقتة في Redis..... 36
4-5-6 تحقيق تخزين البيانات التاريخية في Apache Cassandra..... 36
4-5-7 تحقيق الإظهار على Node-Red..... 37
4-5-8 إضافة البروتوكول AMQP للبنية..... 38

4-5-9 إضافة البروتوكول MQTT للبنية..... 40
الفصل الخامس (تقييم أداء بروتوكولات التراسل في البنية المقترحة) 44
5-1 الدراسة التجريبية وتقييم الأداء..... 44
5-2 اختيار معامل تقييم الأداء..... 44
5-3 مواد وطرق البحث..... 44
5-4 نتائج التقييم 45
5-5 مناقشة النتائج.....46
الفصل السادس (تحسين البنية المقترحة وتداعياتها) 47
6-1 تحقيق النسخة الآمنة..... 47
6-2 إضافة وسيط الرسائلApache Kafka..... 49
6-2-1 نهج دمج Apache kafka مع MQTT 49
6-2-2 مناقشة جدوى دمج Apache kafka مع MQTT.....50
6-2-3 نهج دمج Apache kafka معAMQP.....51
الفصل السابع (التوصيات وآفاق المستقبل) 52
7-1 النتائج والتوصيات..... 52
7-2 آفاق المستقبل..... 52
المراجع.....54

قائمة الأشكال والرسومات البيانية:

الرقم	الشكل	الصفحة
2-3	مبدأ عمل بروتوكول التراسل MQTT	11
3-3	مبدأ عمل بروتوكول التراسل AMQP	12
4-3	بنية العنقود في Apache Storm	14
5-3	مفهوم الStream	14
6-3	مفهوم الSpout	15
7-3	مثال عن مبدأ عمل الطوبولوجيا في Apache Storm	15
1-4	المخطط الكتلي للبنية المقترحة لمراقبة المؤشرات الحيوية	21
2-4	المخطط الكتلي لمنطق المعالجة في Apache Storm	22
3-4	التمثيل التقني للطوبولوجيا في Apache Storm	23
4-4	تصنيف مجالات قيم المؤشرات الحيوية	23
5-4	خرج منتج البيانات	24
6-4	واجهة وسيط الرسائل RabbitMQ	25
7-4	تبويب الأرتال في RabbitMQ	25
8-4	أهم محتويات الملف Storm.yaml	26
9-4	الDaemons المتزامنة لتشغيل Storm UI	27
10-4	Storm UI	27
11-4	المخطط الكتلي لارتباط الSpouts في Stormمع الأرتال في RabbitMQ	28
12-4	خوارزمية تصفية البيانات	29
13-4	خرج معالج تصفية البيانات بحالة استقبال قيمة غير منطقية لدرجة الحرارة	29

14-4	خرج معالج تصفية البيانات بحالة استقبال قيمة غير منطقية لمعدل نبضات القلب	30
15-4	خرج معالج تصفية البيانات بحالة استقبال قيمة لدرجة الحرارة أعلى من معدلاتها الطبيعية	30
16-4	خرج معالج تصفية البيانات بحالة استقبال قيمة لمعدل نبضات القلب أعلى من معدلاتها الطبيعية	31
17-4	خرج معالج تصفية البيانات بحالة استقبال قيم لمؤشرات حيوية ضمن المعدلات الطبيعية	31
18-4	المخطط التدفقي لمعالج تجميع درجة الحرارة	32
19-4	خرج معالج تجميع درجة الحرارة	33
20-4	البنية المعدلة لمصفوفة التجميع والتي تراعي ديناميكية عدد المرضى	33
21-4	خرج معالج إضافة الوسمة الزمنية	34
22-4	المخطط الكتلي لارتباط القنوات في Redis مع الأداة Node-Red	34
23-4	مقطع الكود الذي يبين قنوات Redis	35
24-4	خرج معالج تخزين البيانات المؤقتة Redis	35
25-4	خرج معالج تخزين البيانات التاريخية cassandra	35
26-4	خرج البيانات المخزنة في Redis (صيغة التخزين النهائية)	36
27-4	خرج البيانات المخزنة في cassandra (صيغة التخزين النهائية)	37
28-4	الكود الرسومي في Node-Red	37
29-4	الخرج النهائي للبيانات على Node-Red	38
30-4	التقاط رزم AMQP باستخدام Wire Shark	39
31-4	محتوى رسالة درجة الحرارة التي نشرت عبر AMQP	39
32-4	محتوى رسالة معدل نبضات القلب التي نشرت عبر AMQP	40
33-4	المخطط الكتلي لإضافة MQTT إلى RabbiMQ	41
34-4	المخطط الكتلي لتوجيه رسائل AMQP إلى MQTT	41
35-4	التقاط رزم MQTT باستخدام Wire Shark	42
36-4	محتوى رسالة درجة الحرارة التي نشرت عبر MQTT	43
2-5	تغيير معدل الاستقبال عبر AMQP مع تغيير معدل الارسال	45
3-5	تغيير معدل الاستقبال عبر MQTT مع تغيير معدل الارسال	46
1-6	خرج المنتج ذو النسخة الآمنة	47

2-6	التمثيل التقني للطوبولوجيا في Apache Storm بنسخته الآمنة	48
3-6	التقاط رزم AMQP المشفرة باستخدام Wire Shark	48
4-6	المخطط الكتلي لربط وسيطي الرسائل مع بروتوكولي التراسل	49

قائمة الجداول:

الرقم	الجدول	الصفحة
1-3	مقارنة لأهم بروتوكولات التراسل	10
1-5	مقارنة معدل استقبال الرسائل عبر بروتوكولي التراسل	45

قائمة الاختصارات:

HPC	High Performance Computing
MPP	Massively parallel processing
DAG	Directed Acyclic Graph
IoT	Internet of Things
MQTT	Message Queue Telemetry Transport
AMQP	Advanced Message Queuing Protocol
EHR	Electronic health record

الفصل الأول

المقدمة وأهداف البحث

1-1 مقدمة:

يعرّف المجتمع الذكي بمفهومه الحديث، بأنه المجتمع الصحي الذي يستفيد من بياناته الكبيرة، حيث بدأت تحليلات البيانات الضخمة في مجال الرعاية الصحية تفرض وجودها، فأصبحت بمثابة منصة هامة لتصميم الرؤية الجديدة لمفهوم المجتمع الذكي، وقد بدأ المجتمع يطمح لصياغة نظام رعاية صحية بتكاليف أقل، ونتائج أفضل، كتجربة نوعية للمشتركين؛ و لذلك كان لابد من التوجه نحو تقنيات البيانات الضخمة، و التي تتميز بكونها تقدّم حلولاً منخفضة التكلفة، وقابلة للتطوير بدرجة عالية، وموثوقة، و قادرة على تخزين كميات لا نهائية من البيانات، و ماذا لو كانت هذه البيانات مولدة من الحساسات و أجهزة القياس الطبية التي تنقل بياناتها عبر تقنية انترنت الأشياء Internet of Things .

2-1 مشكلة البحث:

تولد إنترنت الأشياء (IoT) كمية كبيرة من البيانات، حيث تزودنا بالقيم المقاسة التي جمعتها الحساسات، ليتم عرضها في الزمن الحقيقي، أو ليتم تخزينها كملف أرشيف (ملف CSV مثلاً) يمكن العودة إليه لاحقاً؛ بهدف سبر الحالات التي مرت بها المؤشرات الحيوية في فترة سابقة[3]، مما يساعد الشخص الذي يراجعها (طبيب مثلاً) في إيجاد حل لمشكلة ما؛ لذلك كان لابد من إيجاد حلول جديدة لمعالجة هذا الكم الهائل من البيانات و الذي تولده الحساسات و أجهزة المراقبة في بيئات انترنت الأشياء المختلفة، حيث يركز هذا البحث على معالجة البيانات الضخمة Big Data في نظام يراقب المؤشرات الحيوية للمرضى عن بعد معتمداً على انترنت الأشياء.

3-1 أهمية البحث:

لإنترنت الأشياء العديد من التطبيقات المهمة ومنها تقديم خدمات الصحة الإلكترونية، وخاصة بالنسبة للدول ذات الدخل المنخفض والمتوسط، حيثما تحول تحديات النظام الصحي دون تقديم الخدمات الصحية بفعالية إلى المحتاجين وخاصة حالات الرعاية الحرجة والحالات الطارئة، ومن أهم هذه التحديات التي تواجه قطاع الصحة[1]:

1- نقص عدد الكوادر الصحية.
2- الزيادة المستمرة في عدد المرضى.
3- البنى التحتية المتقادمة.
4- الحواجز الجغرافية والاقتصادية؛ وخاصة عند تقديم خدمات الرعاية الصحية في الأرياف والمناطق النائية.

1-4 أهداف البحث:

يهدف البحث إلى بناء نظام مراقبة للمؤشرات الحيوية للمرضى عن بعد، وتحديداً نبضات القلب ودرجة حرارة الجسم، أي أنه يندرج ضمن تطبيقات التطبيب عن بعد. سنعرض في هذا البحث تصميم نظام معتمد على انترنت الأشياء، وتحقيقه، وتقييمه لمراقبة بعض المؤشرات الحيوية للمرضى عن بعد، حيث يستند للتقنيات التالية:

1- انترنت الأشياء IoT.

2- نظم وسائط الرسائل متمثلة بالأداة RabbitMQ، و بروتوكولي التراسل MQTT و AMQP.

3- نظام المعالجة الموزع Apache Storm.

4- نظم إدارة و تخزين البيانات Apache Cassandra , Redis.

5- أداة الإظهار Node-Red.

1-5 الجهات التي ستستفيد من هذا البحث:

1- قطاع الرعاية الصحية، من حيث تحسين مستوى جودة الخدمة، وتخفيف ضغط العمل الملقى على عاتق الكادر الطبي.

2- المرضى، من حيث تقليل نسب الوفيات المفاجئة وخاصة بالنسبة لمرضى الحالات المزمنة، والوقوف الآني على الحالة الصحية للمرضى من قبل أطبائهم، وأيضاً كسر الحاجز الجغرافي لتقديم خدمات الصحة.

1-6 الصحة الإلكترونية (e-Health or EHealth):

هي مصطلح حديث نسبياً ويقصد به استخدام تقنيات الشبكات والاتصالات لتقديم خدمات الرعاية الصحية [1] healthcare، وللصحة الإلكترونية تطبيقات عديدة.

1-7 أهم تطبيقات الصحة الإلكترونية[1]:

1- **خدمة السجل الطبي الإلكتروني الموحد(EHRs)**: و يتضمن البيانات الطبية الخاصة بكل مريض مثل التاريخ الطبي للمريض، والتشخيص، والأدوية، وخطط العلاج، وصور الأشعة، والنتائج المخبرية، وإتاحة نقل هذه المعلومات الكترونياً ولحظياً عن طريق شبكات البيانات بين المرافق الطبية المختلفة للدولة.

2- **التطبيب عن بعد Telemedicine**: ويقصد به استخدام الاتصالات وتكنولوجيا المعلومات لتوفير الرعاية الصحية السريرية من مسافة بعيدة، وقد استخدم هذا المنهج للتغلب على حاجز المسافة، وتحسين إمكانية الحصول على الخدمات الطبية، و التي غالبا ما لا تكون متوافرة باستمرار في المجتمعات الريفية البعيدة، كما أنها تستخدم لإنقاذ الأرواح في حالات الرعاية الحرجة، والحالات الطارئة.

3- **نظم دعم القرارات السريرية Clinical decision support systems**: وتمثّل مجموعة النظم التي تقدّم المعلومات والمعايير إلكترونياً لأخصائي الرعاية الصحية، وذلك بهدف استخدامها في تشخيص الأمراض، وعلاج المرضى.

1-8 انترنت الأشياء:

وتعرّف بأنها شبكة من الأغراض التي تتضمن الالكترونيات والبرمجيات والحساسات والمشغلات والمركبات، وحتى الأجهزة المنزلية المتصلة من خلال شبكة لتبادل البيانات[2].

يعتبر مفهوم انترنت الأشياء الجيل الجديد المتطور و المتنامي في شبكة الانترنت، و الذي يزيد من قدرة الأشياء المادية (الأدوات و الأجهزة المختلفة التي تتميز بعنوان IP مخصص لها)على الاتصال بشبكة الانترنت و تنظيم عملية التفاهم بين الأشياء المادية المترابطة مع بعضها عبر بروتوكولات الانترنت، و أحياناً قد تسمح بعملية الاتصال و التفاهم حتى دون وجود وسيط بشري، و لعل أهم ما يميز هذه التقنية عن الحوسبة التقليدية أن البيانات قد تكون صغيرة الحجم و متواترة في النقل، إضافة إلى أن عدد الأجهزة أو العقد nodes المتصلة بالشبكة عادة ما تكون أكبر منها في الحوسبة التقليدية[4].

وحتى الآن لا يزال موضوع اختيار بروتوكول التراسل الأفضل بالنسبة لتطبيقات انترنت الأشياء موضوعاً قيد الدراسة، وذلك بسبب طبيعة البيانات المتبادلة، وطبيعة الأجهزة المستخدمة وعددها (الأجهزة المقيدة مثلاً)، وطبيعة الشبكات (الشبكات كثيرة التأخير أو قليلة عرض النطاق الترددي أو غير الموثوقة مثلاً)، وكذلك اختلاف البيئات التي أصبحت تعتمد على هذه التقنية، فطبيعة البيئة التي نطبق فيها هذه التقنية قد تفرض الاهتمام بوسائط أكثر من غيرها، أو قد تفرض طبيعة تصميمة معينة.

1-9 البيانات الضخمة:

لقد أصبحت البيانات الضخمة واحدة من المجالات الرئيسية لبحوث مقدمي الخدمات السحابية نظرًا لكمية البيانات الكبيرة المنتجة كل يوم وعدم كفاءة الخوارزميات والتكنولوجيا التقليدية للتعامل مع هذه الكميات الكبيرة من البيانات[5]، و يُعرِّف مؤتمر أوريلي البياناتِ الضخمة بأنها: "البيانات التي تصبح كبيرةً بشكلٍ كافٍ لعدم القدرة على معالجتها باستخدام الوسائل المتاحة"، وللتوضيح أكثر سنستعرض تعريفَ دراسة ماكينزي: "تُشير البيانات الكبيرة إلى مجموعات البيانات التي يكون حجمهُا أكبرَ من إمكانية أدوات برمجيات قواعد البيانات لجمعِها وتخزينها وإدارتها وتحليلها"، و على الرغم من أن العديد من الناس يعرِّفونها بحجمها ولكنْ حتى نفهم البيانات الكبيرة ونفرّقَ بينها وبين البيانات العادية، لا بد من فهم أربعة عوامل هي الحجم، التنوع، السرعة والصحة[6] .

1-10 خصائص البيانات الضخمة[5]:

1- **الحجم (Volume)**: تشير خاصية الحجم إلى حجم البيانات المستخرجة من مصدر ما. إنَّ البيانات ذات الحجم الضخم لا يُمكن تخزينها ولا التعامل معها بالطرق التقليدية كالأقراص أو الحواسيب المركزية، ولذلك فإن الحوسبة الموزعة والسحابية تأخذ دورها بفعالية، إذ تُجزّأ البيانات إلى حزمٍ تُخزَّنُ في مواقعَ مختلفة، تتم إدارةُ هذه البيانات وجلبُها جميعاً باستخدام برنامجٍ شامل مثل هادوب، الذي يقوم باستخدامِ القليلِ من قدرةِ كلّ حاسوب من تلك التي خُزِّنتْ عليها حزمُ البيانات لمعالجة البيانات ذات الأحجام الكبيرة والتي لا يُمكن معالجتُها باستخدام حاسوبٍ واحدٍ مهما كان خارقاً.

2- **التنوع (Variety)**: يشير التنوع إلى أنماط البيانات المختلفة التي من الممكن أن نتعامل معها في وقتنا الحاضر والتي تصنف إلى بياناتٍ مُنظَّمة Structured Data وغيرِ منظَّمة Unstructured Data. كان التركيز في الماضي على

البيانات المنظَّمة كالبيانات المالية مثلاً التي تلائم تماماً الجداول أو قواعد البيانات، ولكن في الواقع، فإنّ أكثرَ البيانات الحالية هي بياناتٌ غيرُ منظّمة (الصور وتسجيلات الصوت على سبيل المثال).

3- **السرعة (Velocity):** يُقصد بها السرعة في توليد بياناتٍ جديدة، وسرعةُ تحرّكِ هذه البيانات والقدرةُ على معالجتها دون تخزينها، كالسرعة في توليد البيانات في سوق الأوراق المالية؛ إذ يجب على المجموعات التجارية أن تحلل وتتخذ القرار بشراء الأسهم أو عدم الشراء خلال ثوان، ويتم عرض البيانات هنا في الزمن الحقيقي.

4- **الموثوقية والصحة (Veracity):** يشير هذا العامل إلى الثقة في البيانات، فنحن عادةً نحلّل البيانات المنظمة والموثوقة، لكننا اليوم مضطرون للتأقلم مع البيانات غير المنظمة وغير الصحيحة والتعامل معها، فهناك دراسات تقدر أنَّ حجم ضرر البيانات الغير الجيدة على الاقتصاد الأمريكي يقدر ب 3.1 ترليون دولار سنوياً.

1-11 عتاد البيانات الضخمة[5]:

تتوفر عدةُ تقنيات حوسبة، إلا أنَّ منصات الحوسبة التفرعية والموزعة هي المنصات الوحيدة الملائمة للتعامل مع سرعة وحجم البيانات المنتجة، منها ثلاثة خيارات متاحة اليوم:

1- **العناقيد والشبكات Clusters & Grids:** يتم فيها ربط الخوادم مع بعضها لتشكِّلَ شبكةً توزع الأعمال فيما بينها. يُمكنُ للعناقيد أو الشبكات أن تكونَ مكونة من أجهزة سلعيّةً إما متجانسةً Homogeneous أو غيرَ متجانسة Heterogeneous.

2- **المعالجة التفرعية أو المتوازية واسعة النطاق (MPP):** هي معالجة لبرنامج معين تتم بالتنسيق بين العديد من المعالجات التي تعمل على أجزاء مختلفة من البرنامج وكل معالج يستخدم نظام التشغيل والذاكرة الخاص به. بشكل عام، تتصل معالجات MPP باستخدام واجهات مراسلة. في بعض التطبيقات، يمكن أن يصل عدد المعالجات العاملة على نفس التطبيق إلى 200 معالج أو أكثر ويسمح ترتيب "الاتصال البيني" بإرسال الرسائل بين المعالجات، ومن الأمثلة منصات المعالجة التفرعية واسعة النطاق EMC Greenplum وParAccel.

3- **الحوسبة ذات الأداء العالي (HPC):** الحوسبة عالية الأداء تجمع بين أجهزة الكمبيوتر والبرامج والخبرات لحل المشكلات التي يصعب حلها بفعالية من خلال وسائل أخرى. يعمل HPC عادة عن طريق تجزئة المشاكل إلى قطع، والعمل على هذه القطع في نفس الوقت. تُنجَز حوسبة الأداء العالي HPC معظم الحسابات في الذاكرة، الأمرُ الذي يجعل الأداء الحسابي أسرعَ ما يمكن. تُعد IBM Blue Gene من الأمثلة على بيئات HPC، إذ تُستخدَم هذه البيئات من قِبَل المنظّمات البحثية ووحداتِ الأعمال التي تتطلّبُ تَدَرُّجيَّةً عاليةً جداً وأداءً حسابياً [7].

الفصل الثاني

الدراسات المرجعية

2-1 المقدمة:

تشكل مسألة المعالجة الموسعة للبيانات الكبيرة تحدياً كبيراً في العصر الحالي، وخاصة تلك البيانات التي تولدها الحساسات وأجهزة المراقبة المرتبطة بتقنية انترنت الأشياء لتقديم خدمة معينة. ينطوي هذا الفصل على دراسة مرجعية شاملة للدراسات السابقة التي عملت على تصميم وتحقيق نظم لمعالجة البيانات الضخمة التي تولدها الحساسات التي تراقب المؤشرات الحيوية للمرضى، والدراسات التي قدمت نُهُجاً لبناء منصة لإنترنت الأشياء باستخدام مكونات مفتوحة المصدر، وكذلك الدراسات التي قدمت تقييماً لبعض بروتوكولات التراسل ضمن بيئات وشروط وحالات استخدام معينة.

2-2 الهدف من الدراسة المرجعية:

إنَّ الهدف من الدراسة المرجعية هو عرض الأعمال البحثية التي قام بها الباحثون السابقون في مجال تصميم نُظم مراقبة المؤشرات الحيوية عن بعد، وتوضيح التقنيات التي اعتمدوا عليها لتقديم هذه الخدمة، وأيضاً الأدوات البرمجية المستخدمة لمعالجة تلك البيانات وتخزينها وعرضها، وتحديد نُهُج التصميم وما تنطوي عليه من مقايضات، وبالتالي إظهار ما يميز هذا البحث عن سابقيه والإضافة التي يقدمها ضمن هذا المجال.

2-3 الدراسات التي قدمت مقترحاً لنظام صحة عن بعد:

قام برايدهارشيني وآخرون [8] باقتراح نظام ذكي لمراقبة الأوضاع الصحية للمرضى، وذلك بهدف الكشف المبكر للأمراض، وتشخيص الأمراض الحرجة، بحيث تتم مراقبة المؤشرات الحيوية للمرضى باستمرار اعتماداً على السحابة، وترتبط الحساسات التي تجمع البيانات مع السحابة عبر Zigbee، وترسل البيانات إلى المستقبل (المخدم) في الزمن الحقيقي بالاعتماد على بروتوكول التراسل MQTT.

اقترح جانجر و آخرون [9] نظاماً مخصصاً لمراقبة الحالة الصحية للمرضى، من خلال استخدام حساسات معتمدة على تقنية البلوتوث منخفضة الطاقة، ومعالجة البيانات في الزمن الحقيقي باستخدام خوارزميات التعلم الآلي لمساعدة مرضى السكري على إدارة حالتهم المزمنة بشكل أفضل، حيث تم استخدام BLEs لجمع بيانات المؤشرات الحيوية للمرضى مثل ضغط الدم، ومعدل ضربات القلب، والوزن، ونسبة الجلوكوز في الدم (BG) من الحساس إلى الهواتف الذكية ، في حين تمّ

استخدام Apache Kafka لمعالجة البيانات في الزمن الحقيقي، و إدارة البيانات الضخمة للحساسات، كما تمّ استخدام MongoDB بهدف تخزين بيانات المرضى.

واقترح سوداكار وآخرون في البحث[10] نظام رعاية صحية عن بعد، قائم على السحابة لجمع البيانات الصحية المطلوبة للتحليل، وتمّ التأكيد على أهمية الميزات التي تقدمها الحوسبة السحابية، لبناء بنية تحتية مجهزة جيداً للمراقبة والتحليل عن بُعد، وقد استخدمت Apache Storm، لإجراء عمليات المعالجة في الزمن الحقيقي.

قدّمت نوال العبودي [11] اقتراح لبنية بيانات كبيرة قابلة للتوسعة، تدعم مفهومي المعالجة المتدفقة و المعالجة الدفعية، بهدف تعزيز موثوقية أنظمة الرعاية الصحية، من خلال توليد الإنذارات في الزمن الحقيقي، وإجراء تنبؤات دقيقة عن الحالة الصحية للمريض، وبالتالي تمّ تصميم نموذج أولي prototype لأنظمة الرعاية الصحية معتمد على أدوات مفتوحة المصدر و أهمها Apache Spark و MongoDB.

أما في البحث [12] فقد قدّم أحمد الخطيب وآخرون مقترحاً لبنية معتمدة على انترنت الأشياء، تقوم بجمع البيانات ونقلها إلى السحابة، حيث تتم معالجتها وتحليلها، ثم تمّ بناء نموذج أولي prototype لهذه البنية لإظهار إيجابيات أدائها، والتي تهدف لتقديم خدمة رعاية صحية ذكية وموثوقة وفعالة للمسنين.

وأما في الدراسة [13] و التي قدّمتها شركة StreamSets، فقد تمّ تحقيق بنية تقوم بتجميع بيانات الحساسات، ثم تقوم بنقلها إلىHadoop، و ذلك باستخدامStreamSets Data Collector، و يتكون السيناريو الذي يعالج البيانات في الزمن الحقيقي من ثلاثة عناصر :

1- "first mile": وظيفته قبول البيانات المستقبلة من الأجهزة، و عادة ما يكون MQTT message broker أو pub/sub mechanism .
2- "StreamSets Data Collector pipeline" : يقوم بتوجيه البيانات، و تصفية البيانات المعطوبة منها.
3- "Hadoop cluster": يتيح معالجة البيانات و تحليلها.

2-4 الدراسات التي قدمت نُهُجاً و تجارب سابقة و أفكاراً نظرية:

قدّم سيدهارت [14] نهجاً لبناء منصة لإنترنت الأشياء باستخدام مكونات مفتوحة المصدر، وأكد على أن هنالك العديد من الطرق لبناء تلك البنى، ونهجاً معيناً يعتمد على حالة الاستخدام، وعادة ما ينطوي على مقايضات بين السرعة والحجم والتكلفة والميزات والدعم وغيرها، كما بين الهدف من استخدام كل مكون ضمن النظام وحالة الاستخدام الأنسب لكل منها، و لذلك فقد تمّ تطوير إعدادات تناسب سيناريوهات لنظم الزمن الحقيقي وغير الحقيقي بالنسبة لعمليات التحليل و المعالجة و تخزين البيانات الخام الواردة، أو البيانات التي تم معالجتها مسبقاً، أو حتى نتائج التحليلات والتقارير.

وأشار إلى الأثر الإيجابي لإضافة بعض الأدوات، والتي تسبب تحسناً واضحاً في أداء النظام بشكل عام، إضافة لسهولة توسيع النظام.

وفي البحث [15] قدّم الدكتور عدمان مريزق نظرة عامة وشاملة عن دور الذكاء الاصطناعي في الطب، وأكد على إمكانية استخدام برامج الحاسب لمساعدة الأطباء وغيرهم من مقدمي الرعاية الصحية في أداء أدوارهم السريرية في التشخيص والعلاج، ثم عرض العديد من التجارب الحقيقة لتطبيقات الطب عن بعد، وركّز على موضوع مراقبة المؤشرات الحيوية عن بعد، فعرض عدّة نظم تم تحقيقها في عدة دول، بهدف مراقبة المؤشرات الحيوية عن بعد في الزمن الحقيقي، مثل معدل نبضات القلب، ومعدل التنفس، وضغط الدم، ودرجة حرارة جسم الإنسان، وغيرها.

أما في البحث[16] فقد أكد مايكل ماركس وآخرون على دور أنظمة المراقبة الصحية عن بعد، ودورها في إحداث ثورة في تقديم الخدمات الطبية، وذلك من خلال توفير بيئة طبية مساعدة، وتسهيل العيش المستقل للمرضى، وإتاحة المراقبة المستمرة للحالات النفسية والصحية للمرضى، وأيضاً تمّ التطرق بشكل موسع إلى خمس قضايا، وهي:

1- مسألة جمع البيانات وخصوصيتها.
2- النماذج والتقنيات والحلول لمعالجة البيانات الطبية وتحليلها.
3- تحليلات البيانات الطبية الكبيرة لرصد الصحة عن بعد.
4- التحديات والفرص البحثية في تحليلات البيانات الطبية.
5- بعض الأمثلة لدراسات الحالة والحلول العملية.

وفي البحث [17] أكد ليدونغ وانغ على أهمية بيانات الصحة، كونها مصادر لا تقدر بثمن، وخاصة عندما نطبق عليها خوارزميات التعلم الآلي أو التنقيب في البيانات، لإيجاد توصيات العلاج الضرورية، واتخاذ الإجراءات المناسبة للمريض، والقرارات الفعّالة التي قد تغير خيارات نمط الحياة بالنسبة للمريض، والتشخيص المبكر، والذي يعد أمراً حيوياً لتحسين جودة الرعاية الصحية بشكل كبير، حيث استخدم تقنيات البيانات الضخمة مثل hadoop و Spark.

2-5 الدراسات التي قدمت تقييماً لبروتوكولات التراسل وفقاً لشروط وبيئات معينة:

أجرى نيتين نايك [18] تقييماً تفصيلياً لبروتوكولات التراسل الأربعة MQTT و CoAP و AMQP و HTTP، بهدف عرض خصائصها، و تحليلها بشكل متعمق، و تمّ ذلك استناداً إلى بعض المعايير المترابطة، بهدف إعطاء نظرة عميقة على نقاط القوة والقيود المتعلقة بكل بروتوكول، و فهم إيجابيات وسلبيات هذه البروتوكولات بالنسبة لأنظمة إنترنت الأشياء، و تحديد سيناريوهات العمل الأفضل لكل منها.

و في البحث [19] أجرى هاري و زميله مقارنة فعلية بين بروتوكولات التراسل CoAP وMQTT وXMPP و WebSocket و ذلك من خلال تنفيذ تطبيق يحاكي موقف سيارات ذكي باستخدام برمجيات مفتوحة المصدر لتلك البروتوكولات، و قام بقياس زمن الاستجابة عبر تغيير حجم ال(different loads) Traffic، فوجد أنه بالنسبة لاستخدام منخفض للمخدم كان CoAP هو الأفضل بين البروتوكولات القائمة على أرتال الرسائل، و لكن عندما يدعم التطبيق multi-threading يصبح XMPP هو الأفضل من حيث الاستخدام المنخفض للمخدم، و عندما قام بزيادة استخدام المخدم فإن متوسط زمن الاستجابة للبروتوكولات يزيد باستثناء WebSocket الذي يسلك سلوكاً معاكساً، و من جهة أخرى عندما قام بزيادة استخدام المخدم تبين أن WebSocket هو الأفضل بين البروتوكولات الثلاثة .

أجرى توماس و زميله يانغ [20] مقارنة بين أهم بروتوكولات التراسل في انترنت الأشياء وهي CoAP وMQTT و DDS وcustom UDP-based protocol in a medical setting، حيث تم تقييم أداء البروتوكولات باستخدام محاكي شبكة عتادي network emulator، و الذي أتاح التقييم بالاعتماد على بعض المعايير مثل استهلاك عرض الحزمة الترددية، التأخير، فقدان الرزم و غيرها من المعايير، و من الجدير بالذكر بأن عملية التقييم الفعلية لأداء البروتوكولات أجريت ضمن قيود شبكة لاسلكية و ضمن سيناريو طبي مستخدماً أدوات محاكاة الشبكة و هي TBF و التي تستخدم لتعيين عرض النطاق الترددي على كل من المخدم والعميل، و كذلك الأداة NetEM و التي تستخدم لتعيين فقدان الرزم و التأخير على الرزم الصادرة، و اعتمد على Wireshark لقياس عرض النطاق الترددي المستهلك، و أيضاً قام بتطوير تطبيقي مخدم وعميل من أجل كل بروتوكول لإرسال و استقبال المعطيات اعتماداً على تحقيقات موجودة سابقاً .

أما الدراسة الأخيرة [21] و التي لا تنتمي لأي تصنيف، حيث وصّفت هذه الورقة الخصائص الخاصة لإنترنت الأشياء التي يمكن أن تؤدي دوراً أساسياً في تمكين قدرات الصحة العامة، و ذلك من خلال توصيف بنية نظام معلومات الصحة العامة القائم على الحفاظ على خصوصية إنترنت الأشياء؛ كما تم تقييم الخصوصية لهذا النظام، فتمت الإشارة إلى أننا نستطيع إنشاء نظم معلومات الصحة العامة التي تستخدم قدرات إنترنت الأشياء لضمان الخصوصية والأمن وعدم الكشف عن الهوية على وجه التحديد، و تمت الإشارة لنقطة أخرى و هي أن قدرات إنترنت الأشياء لا تتعارض مع الحفاظ على الخصوصية، وذلك بسبب تركيز الصحة العامة على البيانات الإجمالية وليس البيانات الفردية .

2-6 الجديد الذي يقدمه هذا البحث:

1- تصميم وتحقيق بنية نظام معتمد على برمجيات مفتوحة المصدر تمثل نواة لنظام قابل للتطوير، وذلك لتهيئته مستقبلاً لبناء جيل مطوّر وحديث لنظم داعمة لاتخاذ القرارات السريرية، وقادرة على التنبؤ بالحالة الصحية للمريض.

2- تقييم بعض بروتوكولات التراسل وفقاً لحالات استخدام معينة، وباعتبار الشروط والقيود التي تفرضها هذه البيئة.

3- تحديد ماهية الثغرة الأمنية لتقنية انترنت الأشياء باعتبار شروط البنية المقترحة، و تحديد المشكلة البحثية بشكل دقيق.

4- تحديد وسيط الرسائل الأفضل بالنسبة لهذه البنية و مثيلاتها ممن تدمج المحورين (IoT & Big Data) لأداء دورها تبعاً للمعطيات المتاحة .

الفصل الثالث

نظرة عامة عن مكونات البنية المقترحة

3-1 المقدمة:

إن التمثيل الأقرب لبنية معتمدة على انترنت الأشياء، وتستند للرسائل هو أن تعتبرها جهازاً عصبياً يدير عملية تبادل المعلومات بين مختلف المكونات، حيث يمكن أن نعتبر محرك تحليل البيانات analytics engine هو الدماغ الذي يقرر ماذا سيصنع بالبيانات وبقواعد البيانات التي تخزن تلك البيانات[14]، سنقدم في هذا الفصل دراسة وتحليل لمكونات البنية المقترحة لمراقبة المؤشرات الحيوية، وتعريفاً موجزاً لكل مكون، وتعليلاً منطقياً لسبب اختياره وفقاً لحالة الاستخدام وما تفرضه من قيود.

3-2 بروتوكولات التراسل:

3-2-1 البروتوكول MQTT :

هو بروتوكول يعمل على مبدأ نشر وتشارك publish/subscribe الرسائل البسيطة للغاية، حيث صُمم من أجل العمل مع الأجهزة محدودة الموارد، والشبكات كثيرة التأخير، أو قليلة عرض النطاق الترددي، أو غير الموثوقة، وهذا ما جعل منه بروتوكولاً مثالياً لاتصالات M2M وتطبيقات IoT [22].

يعتبر هذا البروتوكول فعالاً جداً so lightweight، فهو مدعوم من قبل أصغر أجهزة المراقبة والقياس [23]، ومن الجدير بالذكر أن أهم الميزات التي تشجع على استخدامه في تطبيقات IoT الحجم المضغوط للترويسات، حيث يبلغ الحجم الأصغري للترويسة 2 Byte [24]، ويمتاز أيضاً بانخفاض النفقات العامة overhead و استهلاك الطاقة، و يبين الشكل (3-1) أهم نقاط التمايز التي يتمتع بها هذا البروتوكول مقارنة مع بروتوكولات التراسل الأخرى [25]:

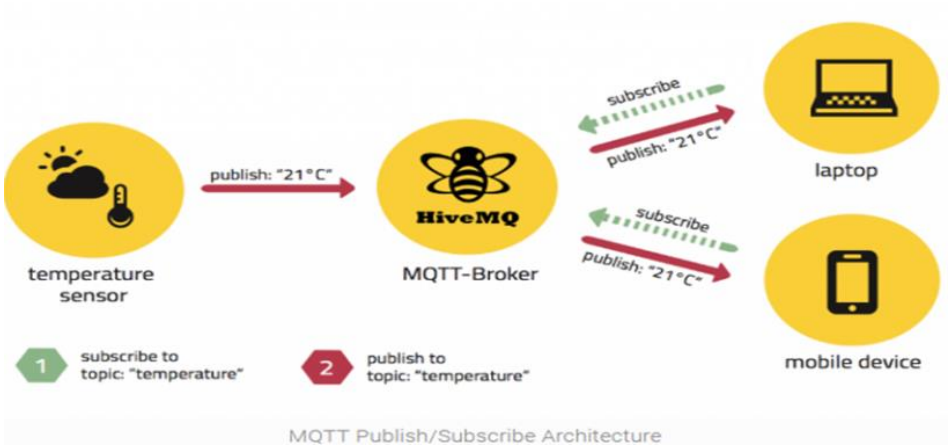
Table 1: IoT Transport Layer Standards Comparison

Protocols	UDP/TCP	Architecture	Security and QoS	Header Size (bytes)	Max Length(bytes)
MQTT	TCP	Pub/Sub	Both	2	5
AMQP	TCP	Pub/Sub	Both	8	-
CoAP	UDP	Req/Res	Both	4	20 (typical)
XMPP	TCP	Both	Security	-	-
DDS	TCP/UDP	Pub/Sub	QoS	-	-

الشكل (3-1): مقارنة بين أهم بروتوكولات التراسل [25]

3-2-2 مبدأ عمل MQTT:

يقوم نموذج publish/subscribe على مفهوم يسمى الموضوع Topic، والذي يتشارك عليه العملاء Clients، ففي مثالنا الموضح في الشكل (3-2)، نجد أن درجة الحرارة تمثل ال Topic، ونلاحظ وجود مكون أساسي في النموذج يتمثل بوسيط الرسائل MQTT-broker، والذي يمثل نقطة الاتصالات المركزية، فهي المسؤولة عن إيفاد الرسائل بين المرسلين والمستقبلين الشرعيين، وأهم ما يميز هذا النموذج أنه يقدم حلولاً قابلة للتوسيع لحد كبير من دون تبعيات بين جامعي البيانات وطالبيها [26].



الشكل (3-2): مثال يوضح مبدأ عمل البروتوكول MQTT [26]

3-2-3 تعليل إضافة البروتوكول MQTT للبنية:

تُعزى إضافة البروتوكول MQTT إلى البنية التي تم تحقيقها إلى اعتبارات متعلقة بدعم الدارات وأجهزة القياس المختصة بجمع بيانات المؤشرات الحيوية، بمعنى أن بعض دارات جمع البيانات مثل Arduino لا تدعم البروتوكول AMQP برمجياً، أي لا توجد مكتبة يمكن تضمينها في كود Arduino لدعم البروتوكول AMQP، بعكس Raspberry Pi التي تتيح تنصيب نسخة نظام تشغيل مثل Linux و دعم AMQP في عملة التراسل [27].

3-2-4 البروتوكول AMQP [28]:

هو بروتوكول قياسي مفتوح المصدر يقدم ميزات إضافية عن بقية بروتوكولات التراسل، وأهمها تقنية الأرتال، والتوجيه، والأمن، والموثوقية، ويعمل على مبدأ النشر والتشارك publish/subscribe، وأيضاً مبدأ نقطة لنقطة point-to-point.

3-2-5 مبدأ عمل AMQP [28]:

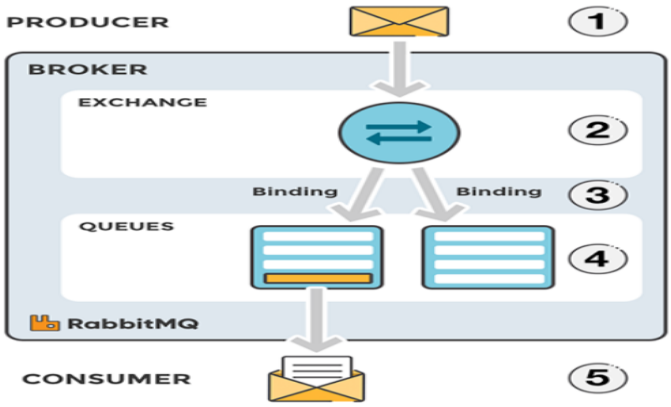
يعتمد AMQP في عمله على تقنية الأرتال، والتي تتيح إمكانية تخزين الرسائل المتعددة، وهذا ما يميزه بشكل أساسي عن MQTT والذي تتيح أرتاله الرسالة الأخيرة فقط، كما يتيح AMQP نشر الرسالة مع الخصائص إضافية مثل: الطابع

الزمني، ونوع الرسالة وغيرها، ويعتمد في أبسط حالاته على اسم الرتل لتوجيه الرسائل وتسليمها لوسيط الرسائل، ويعتمد في عمله على ثلاثة عناصر أساسية وهي:

1- المبدلات exchanges: هي النقاط النهائية التي تستقبل الرسائل.

2- الأرتال queues: وظيفتها تخزين الرسائل التي توجه لها من المبدلات، و تستخدم من قبل المشتركين لتشارك الرسائل.

3- الارتباطات bindings: هي القواعد التي تربط المبدلات والأرتال، وهذا موضح في الشكل (3-3).



[28] الشكل (3-3): مبدأ عمل البروتوكول AMQP

3-2-6 تعليل إضافة البروتوكول AMQP للبنية:

تُعزى إضافة البروتوكول AMQP إلى البنية التي تم تحقيقها إلى اعتبارات متعلقة بالأداء، والسبب يرتبط بشكل أساسي باستخدام وسيط الرسائل RabbitMQ، والذي يمثل تحقيقاً برمجياً للبروتوكول AMQP وبالتالي يحققان معاً الأداء الأفضل(معدل نشر الرسائل خلال واحدة الزمن).

3-3 وسيط الرسائل RabbitMQ :

هو التحقيق الفعلي للبروتوكول AMQP و الذي يوفر للتطبيقات البرمجية منصة مشتركة لإرسال واستقبال الرسائل، ومكاناً آمناً للتخزين حتى يتم استلامها، حيث يدعم مجموعة متنوعة من بروتوكولات التراسل بشكل مباشر أو من خلال إضافات plugins، حيث طور أساساً لدعم البروتوكول AMQP، ولاحقاً منحته الإضافات plugins القدرة على دعم بروتوكولات التراسل أخرى مثل STOMP و MQTT و HTTP ، أما النسخة RabbitMQ-3.7.9 و التي صدرت في تشرين الثاني من عام 2018 ، و قد تميزت عن سابقاتها من النسخ بدعمها للبروتوكول HTTPS.

و قد صُمم كأداة مفتوحة المصدر تؤدي دور وسيط أو وكيل رسائل للأغراض العامة، و يقدم الدعم للعديد من أنماط الاستخدام مثل point to point، و request/reply، pub-sub، و يركز على التسليم المتسق للرسائل إلى المستهلكين

و يتتبع الوسيط حالة المستهلك، و من أهم ما يميزه هو أنه مدعوم بشكل جيد (مكتبات العملاء Java، NET.، node.js، Ruby، PHP والعديد من اللغات الأخرى)، ولديه العشرات من الإضافات المتاحةplugins التي تساعده على دعم حالات استخدام أكثر وسيناريوهات تكامل متنوعة[29].

ومن المهم التأكيد على أهمية استخدام مثل هذه الأدوات عندما نعلم بأن النظام قد يحتاج لتوسيع في المستقبل، وبالتالي الحاجة لآليات التخزين المؤقت buffering mechanism [14]، و لكن حقيقة اختيار RabbitMQ تعود إلى دعمه المستقر لبروتوكولات التراسل، تلك النقطة التي قد تجعل النظام يعاني من بعض المحدوديات، و يفقد بعض الميزات عندما نستخدم منافساً متفوقاً عليه بالأداء مثل Apache Kafka,

3-4 معالج البيانات الضخمة Apache Storm [30]:

هو نظام حوسبة موزع مفتوح المصدر، صُمم لمعالجة البيانات الضخمة في الزمن الحقيقي بطريقة متوازية، ويعتبر النظام الأفضل والأشهر لمعالجة وتحليل البيانات الضخمة في الزمن الحقيقي؛ لذلك تستخدمه معظم الشركات كجزء أساسي في نظمها الخدمية.

3-4-1 أهم ميزات Apache Storm [31]:

1- يمتلك هذا النظام العديد من حالات الاستخدام مثل: تحليل المعطيات في الزمن الحقيقي real time analytics، والتعلم المباشر للآلة online machine learning ، والحوسبة المستمرة continuous computation، حيث يقوم بتسهيل عمليات نقل البيانات غير المحدودة، ومعالجة تدفق البيانات بشكل فوري في الزمن الحقيقي، واستدعاء الإجرائيات الموزعة عن بعد distributed RPC، واستخراج وتحويل وتحميل قواعد البيانات ووضعها في قاعدة بيانات أخرىETL .

2- يعتبر نظاماً بسيطاً، ويمكن استخدامه مع أي لغة برمجة.

3- السرعة العالية، حيث تصل سرعته في معالجة الأحداث الآتية من المصادر الخارجية إلى أكثر من مليون حدث في الثانية (أي أكثر من مليون رسالة معطيات) على كل عقدة.

4- يتمتع بقابلية التوسيع، والقدرة على التكيف مع الأعطال، ويقدم الضمانات حول معالجة البيانات، ويسهل إعدادها وتشغيلها. القدرة على التكامل مع تقنيات قوائم الانتظار وقواعد البيانات.

وتعتبر ميزة السرعة العالية هي السبب الأساسي لاختياره لمعالجة بيانات المؤشرات الحيوية في الزمن الحقيقي، ويعتبر مناسباً جداً لأنظمة انترنت الأشياء.

3-4-2 بنية العنقود في Apache Storm [32]:

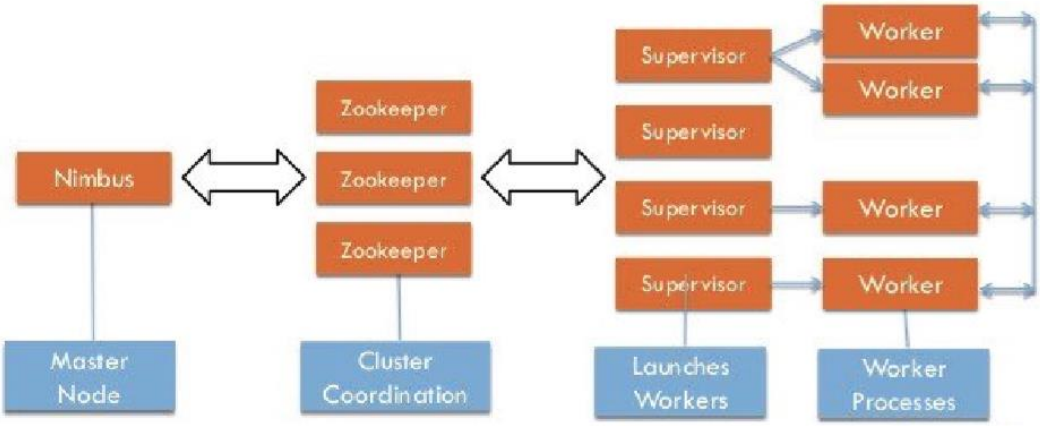
لدينا نوعان من العقد في عنقود Storm:

1- العقدة السيد Master node:

تشغل daemon اسمه Nimbus، وتقوم بتوزيع العمل على العقد العمال Worker node، فتسند المهام للآلات، وتراقب الأخطاء.

2- العقدة العامل Worker node :

تشغل daemon اسمه Supervisor، حيث تصغي للعمل المسند لها من قبل Nimbus إلى الآلة الخاصة بها، فعملية العامل Worker process تنفذ العمل الجزئي المسند لها من الطوبولوجيا، وهذا ما يوضحه الشكل (3-4):



[32] الشكل (3-4): بنية العنقود في Apache Storm

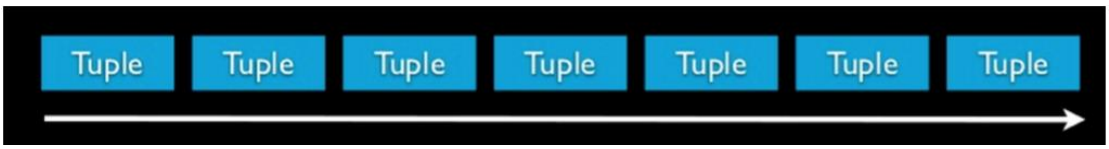
ولدينا خدمة وسيطة تدعى Zookeeper وظيفتها التنسيق والربط بين Supervisor daemons و ال Nimbus daemon، وأيضاً يقوم بحفظ حالة كل منهما عند حصول فشل لسبب ما.

3-4-3 بنية الطوبولوجيا في Apache Storm [32]:

لنعرف أولاً المصطلحات الاختصاصية بالطوبولوجيا في **Storm**:

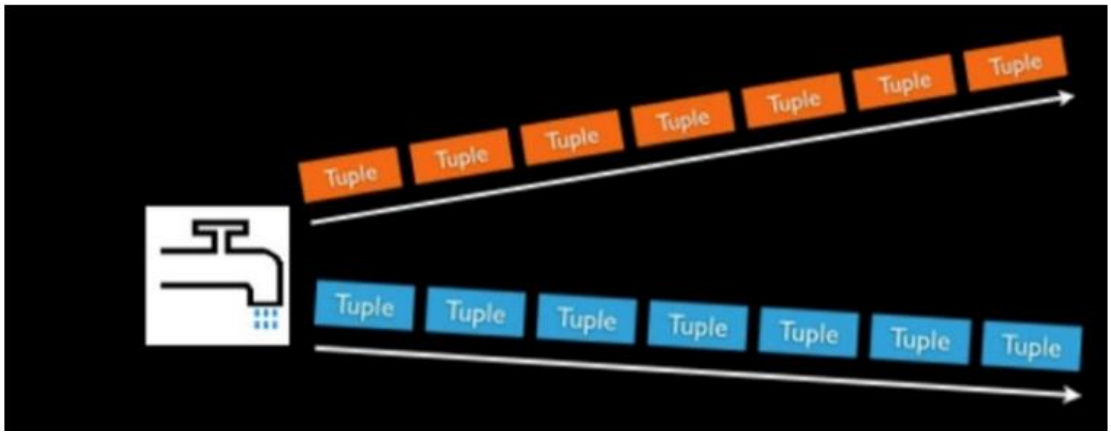
1- Tuples: هي قائمة مرتبة من عناصر المعطيات.

2- Streams: هي تسلسل غير محدود من Tuples، و الشكل(3-5) يمثل توضيحاً لمفهوم الStreams .



[32]الشكل(3-5): مفهوم الStreams

3- Spouts: مصادر ال Streams إلى الطوبولوجيا، و الشكل (3-6) يوضح مفهوم الSpouts .

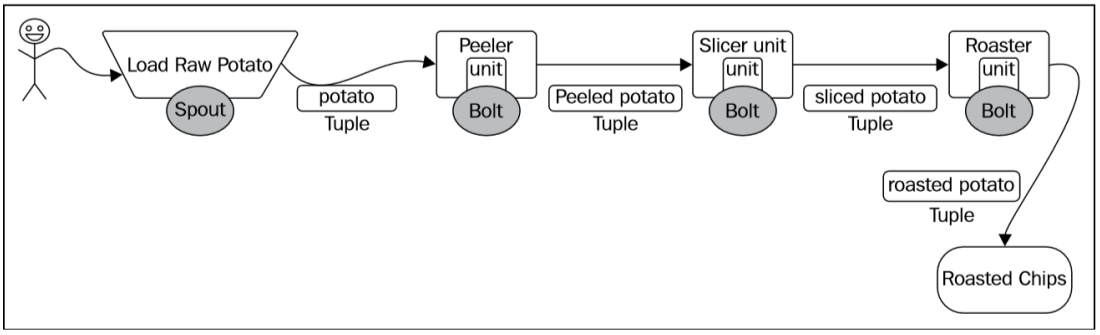


الشكل (3-6): مفهوم مصادر البيانات Spouts[32]

4- Bolts: تعالج ال Streams المُدخل، و تنتج الخرج، حيث تمثل الوظائف التي تقوم بعمليات التصفية و التجميع و التخاطب مع قواعد البيانات .

5- Topologies: هي بيان DAG تتضمن فيه كل عقدة على منطق المعالجة الجزئي، أما الروابط بين العقد إلى فتشير لكيفية تدفق البيانات.

لنوضح الفكرة بالمثال البسيط الموضح في الشكل(3-7)، نفترض أن لدينا خط انتاج للشيبس المقرمش، ويتكون من صنبور وحيد Spout، حيث يتم تحميل البطاطا بشكلها الخام (هنا تمثل مصدر المعطيات بشكلها الأولي قبل إجراء أي عملية معالجة)، ثم ثلاث نقاط Blots وهي حسب الترتيب وحدة التقشير، ووحدة التقطيع، ثم وحدة التحميص، وبذلك يكون الخرج النهائي هو الشيبس المقرمش[33].



الشكل (3-7): مثال يوضح مبدأ عمل الطوبولوجيا في Apache Storm [33]

3-5 نظام التخزين Apache Cassandra [34]:

هو نظام موزع لإدارة قواعد البيانات من نوع NoSQL، صمَّم للتعامل مع البيانات الضخمة، حيث يتمتع هذا النظام بميزات هامة تشجع المصممين على اختيارها، وخاصة لأغراض تخزين البيانات التاريخية، من أهم هذه الميزات:

1- تساعد على حل المهام المعقدة بكل سهولة:

تسجيل الأحداث Event logging، وجمع المقاييس metrics collection، وإجراء الاستعلامات عن البيانات التاريخية، قد تبدو كل هذه المهام مملة، ومع ذلك فهي ذات أهمية قصوى لسير عمل نظم البيانات الضخمة، و يمكن أن تكون عمليات التخزين المركزي للسجلات logs، و المعايير metrics مهمة شاقة للغاية؛ نظراً لتنوع البيانات و مصادرها، و حقيقةً إن بناء مخزن مركزي للسجلات والمقاييس واسترجاع المعلومات التاريخية من هذا التخزين، تعتبر من المهام التي تتعامل معها Cassandra بسهولة عالية، وبمجرد اختيار هيكل الجدول وتصميمه، يمكننا بسهولة توسيع النظام حسب الحاجة.

2- إمكانية تعلم التعامل مع Cassandra خلال فترة زمنية قصيرة:

حيث تنفذ (Cassandra Query Language) CQL لغة استعلام Cassandra، و هيSQL، ولكن جردت من الميزات الأكثر تقدماً، فهذه الأداة قادرة على تقديم أداء جيد باستخدام قائمة محدودة من المتغيرات والأوامر والوظائف، و نظراً لهذه البساطة، فإن مهندس البيانات الكبيرة يستطيع إتقان Cassandra في غضون 30 يومًا تقريباً، وبالتالي تقصير الزمن اللازم لتسويق المنتج بشكل كبير.

3- تخفض Cassandra النفقات العامة overhead:

كما ذكرنا سابقاً، فإن تسجيل الأحداث، وجمع المقاييس، والعمل مع البيانات التاريخية working with historical data هي الاستخدامات الواضحة ل Cassandra، لذلك سيأتي الوقت الذي يكون فيه فريق العمل قادراً على استخدام الأداة إلى أقصى حد، و سيجدون بالتأكيد المزيد من المهام التي يمكن تفويضها إلى Cassandra وأدائها بشكل ممتاز، فتكاليف الإدارة المنخفضة لهذه الأداة تجعل منها مفيدة للغاية؛ لأن فريق العمل سيكون أكثر قدرة على التركيز على مهامه الأساسية، مثل تحسين المنتج ومميزاته، بدلاً من إضاعة وقته في التعامل مع المشاكل و القضايا الجانبية.

4- تقدم Cassandra قدرات الكتابة والقراءة السريعة:

حقيقة إن Cassandra تم تطويرها من أجل ال Facebook، فهي تمتلك القدرة على تحديث ملايين عمليات القراءة والكتابة في كل ثانية، و تقدم مستويات مذهلة من الأداء، و ما هو أهم من ذلك، ما تقدمه من القيم الخطية من دون عناء تقريباً، و بالتالي فإننا بعد قياس قيم أداء القراءة / الكتابة على مخدم واحد ، نستطيع ببساطة حساب عدد المخدمات الإضافية التي يجب إضافتها إلى المجموعة للوصول إلى مستويات الأداء المطلوبة، والتغيير بسهولة بالغة.

5- توفر Cassandra المرونة القصوى والتسامح مع الأخطاء:

إن Cassandra هي مجموعة غير مقادة a masterless cluster، و بالتالي لا توجد نقطة واحدة للفشل، إذ يتم تكرار النسخ المتماثلة للبيانات باستمرار عبر الcluster لضمان أن وقت تشغيل الخدمة هو بنسبة 100٪ بغض النظر عن عدم توفر مؤقت يصل إلى نصف عدد المخدمات، و هذا مفيد بشكل خاص عند تنفيذ التحديثات الدورية، أو عند إجراء بعض عمليات الصيانة على clusters وبغض النظر عما إذا كان مخدم ما أو مجموعة مخدمات، أو حتى مركز البيانات بالكامل خارج الاتصال، فلن يواجه العملاء فترة توقف الخدمة، بل سيظل التطبيق دائماً موجوداً في الخدمة، تمامًا مثل Facebook و Netflix.

و يعزى السبب في اختيار Apache Cassandra لحالة البيانات التاريخية إلى تفوقها على منافسيها من قواعد NOSQL في العديد من حالات الاستخدام مثلMongoDB وCouchbase و HBase، و حقيقة إن مقياس الأداء و غيره منحنا حقيقة موحدة هي أن Apache Cassandra هي الأفضل كقاعدة بيانات من نوع NOSQL، لذلك فإن الشركات التي تحتاج إلى قاعدة بيانات حديثة وموزعة لتطبيقاتها على الويب والجوّال وتطبيقات IOT اعتمدت هذه الأداة[35].

3-6 نظام التخزين Redis :

هو مخزن بيانات في الذاكرة مفتوح المصدر، يستخدم كقاعدة بيانات، وذاكرة تخزين مؤقت، ووسيط رسائل[36]، ويعتبر الأفضل عند الحاجة للتخزين المؤقت Caching كونه يعمل في الذاكرة، إضافة لميزات عديدة نذكر أهمها[37]:

1- سريع للغاية، فقد كتب بلغة C .

2- حاليا يتم استخدامه من قبل عمالقة التكنولوجيا مثل GitHub، Weibo، Pinterest، Snapchat، Craigslist، Digg، StackOverflow، Flickr.

3- مدعوم من غالبية لغات البرمجة.

4- مفتوح المصدر ومستقر.

ويعزى السبب في اختيار Redis لأداء عملية التخزين المرحلي للبيانات لأسباب عديدة أهمها السرعة العالية، وأيضاً كونها مدعومة بمكتبة ربط بسيطة تتيح ربطها مع الأداة Node-Red، وبالتالي تنفيذ نموذج pub/sub بأسلوب سهل و مستقر.

3-7 الأداة Node-Red :

هي أداة مفتوحة المصدر تتيح توصيل الأجهزة مع بعضها، وواجهات برمجة التطبيقات، والخدمات عبر الإنترنت بطرق جديدة[38]، صُممت بشكل أساسي لتطبيقات انترنت الأشياء، وتتمتع بسهولة بناء الكود عن طريق ربط المكونات بشكل تسلسلي(شبيه لنمط البرمجة السلمية) [39] .

ويعزى السبب في اختيار Node-Red كونها تعتبر من التقنيات الحديثة الملائمة لإدارة و عرض الحجوم الكبيرة للبيانات الضخمة في الزمن الحقيقي[44].

الفصل الرابع

تصميم وتحقيق نظام لمراقبة المؤشرات الحيوية للمرضى عن بعد معتمد على انترنت الأشياء

4-1 المقدمة:

إن تصميم هذا النظام يقوم على محورين و هما محور انترنت الأشياءIoT، و محور البيانات الضخمة Big Data، و ينطوي على مقايضات بين الأداء و الدعم من جهة و بين الأداء و الأمن من جهة أخرى، سنشرح في هذا الفصل مكونات البنية الأساسية لنظام مراقبة المؤشرات الحيوية للمرضى عن بعد، و دور كل منها.

4-2 البنية المقترحة لمراقبة المؤشرات الحيوية:

4-2-1 المنتج (Publisher) producer:

يقوم بتحصيل المعطيات من جسم المريض، ويمثّل الحساسات وأجهزة مراقبة المؤشرات الحيوية مثل درجة حرارة الجسم، ومعدل نبضات القلب، ومعدل الأكسجين في الدم وغيرها، حيث تصل هذه البيانات لمداخل حاسوب مصغر MINI computer، ذي تكلفة منخفضة مثل Arduino uno، ليقوم بدوره بإرسال البيانات إلى حاسوب مصغر مثل Raspberry PI، والذي يؤدي دور Gateway، وبالتالي ترسل البيانات عبر بروتوكول التراسل، ولكن في الواقع استعضنا عن المنتج الحقيقي بمحاكي برمجي(كود محقق بلغة جافا)، يولّد القيم لعدد ديناميكي من المرضى، و ذلك ليلائم قابلية التوسع في عدد المرضى، تبعاً لمتطلبات التجربة البحثية و العملية.

4-2-2 بروتوكول التراسل Messaging Protocol:

وظيفته نقل البيانات المتعلقة بالمؤشرات الحيوية للمرضى من الحساسات وأجهزة المراقبة إلى وسيط الرسائل، ولدينا العديد من بروتوكولات التراسل التي تدعم تقنية انترنت الأشياء. تعتمد البنية التي اقترحناها على MQTT وAMQP، ويمثل البروتوكول في التحقيق مكتبة مفتوحة المصدر يتم تضمينها في المنتج، ليتم إرسال البيانات عبر البروتوكول المطلوب، حيث استخدمنا وظائف المكتبة rabbitmq-java-client-bin-3.5.4 التي تدعم ارسال البيانات عبر AMQP، وكذلك وظائف المكتبة org.eclipse.paho.client.mqttv3-1.2.1 التي تدعم ارسال البيانات عبر MQTT.

4-2-3 وسيط الرسائل Message Broker:

يقوم باستقبال الرسائل وإعادة توجيهها، ويشبه مبدأ عمله مكتب البريد وساعي البريد، إذ يختلف عنه فقط بكونه لا يتعامل مع الورق، وإنما يستقبل ويخزن ويوجه البيانات الثنائية (الرسائل)، ولعل أهم دواعي استخدام مثل هذه الأدوات هي ميزة التخزين المؤقت buffering mechanism، ويعتبر RabbitMQ وسيط الرسائل الأكثر استخداماً وبساطة، والذي يمنح النظام ميزة المرونة وقابلية للتوسع scalability.

4-2-4 معالج البيانات الضخمة Apache Storm:

يجري عمليات المعالجة الإضافية، مثل إضافة الوسمة الزمنية time stamp للعينات التي تم جمعها، وأيضاً إجراء عملية تصفية البيانات data filtering مثل استبعاد القيم غير المنطقية والتي تكون ناتجة عن عطل فني في الحساس مثلاً، أو تحديد قيمة قراءة مفقودة لسبب ما مثل سقوط الحساس أو انزياحه عن مكانه المخصص على جسم المريض، ولا يقتصر دور المعالج Bolt، على معالجة تدفق المعطيات، وإرسال الخرج للمعالج التالي، وإنما قد يكون دوره قائماً على تحقيق التخاطب مع قواعد البيانات مثل Apache Cassandra.

4-2-5 نظام التخزين المؤقت للبيانات Redis:

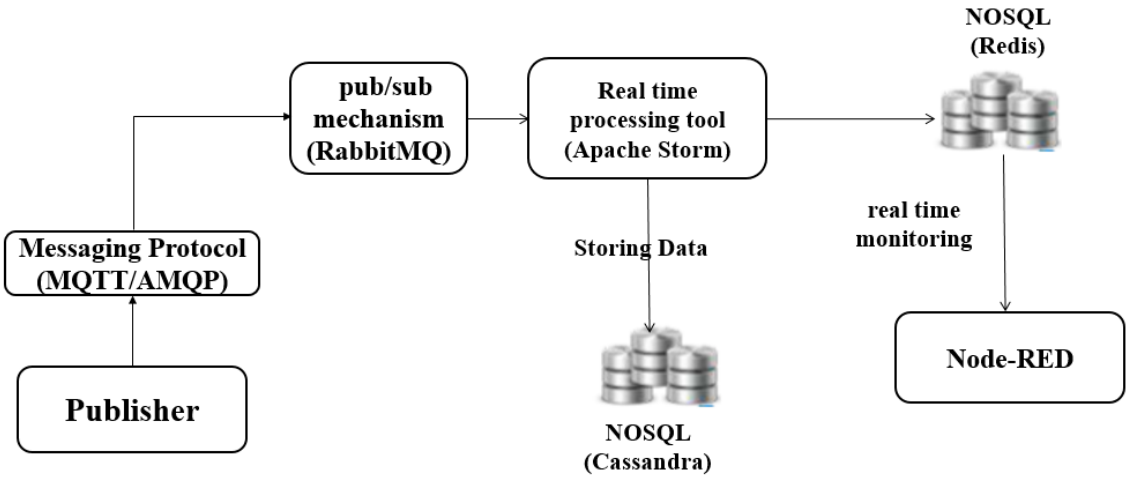
يقوم بعمليات التخزين السريع في الذاكرة، للبيانات التي تمّت معالجتها في Storm، كمرحلة وسيطة ليتم عرض البيانات في الزمن الحقيقي، وبهذه الحالة يستطيع الطبيب المناوب مراقبة حالة أي مريض في الزمن الحقيقي، والاطلاع على قيم المؤشرات الحيوية بشكل لحظي، وكأنه قام بزيارة هذا المريض في اللحظة التي قرأ بها قيم المؤشرات، وبالتالي يستطيع إعطاء توجيهاته عند الحاجة لعناصر التمريض المتواجدين.

4-2-6 نظام تخزين البيانات التاريخية Apache Cassandra:

و يتم فيه تخزين البيانات بعد جمعها ومعالجتها، لتكون جاهزة للاستدعاء في مرحلة لاحقة من قبل طبيب يريد مراجعة حالة المريض؛ مما يمكّن الطبيب من الاطلاع على السجل السابق للمريض (مثلاً ما تعرض له من حالات تسارع في نبضات القلب سابقاً، وتواتر حدوث هذه الحالات، وأيضاً حالات ارتفاع درجة الحرارة، وتواتر حدوثها وسببها)، و بالتالي ستكون هذه البيانات في مرحلة لاحقة دخلاً لنظام مطوّر، يقوم بتحليل البيانات الضخمة، ليقدّم الدّعم للاختصاصيين عند اتخاذ القرارات السريرية، أي إمكانية توقع الحالة الصحية للمريض في المستقبل القريب.

4-2-7 أداة عرض البيانات في الزمن الحقيقي Node-Red:

تؤدي عمليات الإظهار لبيانات المرضى في الزمن الحقيقي، مثل القيمة الآنية ومجموع القيم الإجمالية بشكل تشابهي، و الوسمة الزمنية، و غيرها من التفاصيل، ويوضح الشكل (4-1) المخطط الكتلي لبنية مراقبة المؤشرات الحيوية للمرضى، والتي شرحنا وظائف مكوناتها أعلاه.



الشكل (4-1): المخطط الكتلي للبنية المقترحة لمراقبة المؤشرات الحيوية للمرضى

4-3 منطق المعالجة ضمن Storm Topology:

4-3-1 عملية تصفية البيانات:

تقوم بفلترة وتصفية المعطيات غير المنطقية، والناتجة عن أعطال فنية في الحساسات وأجهزة المراقبة (استبعاد القيم غير الصحيحة)، أو تلك التي قد تنتج بسبب انزياح الحساس عن مكانه الصحيح.

4-3-2 عملية التمييز بين المؤشرات من حيث نوعها:

وهنا نميز حالتين:

1. المؤشرات الحيوية الحرجة: وتتغير قيم تلك المؤشرات بتواتر سريع، ومؤثر في حياة الإنسان، مثل تواتر تغير معدل نبضات القلب، ففي هذه الحالة سيتم تمرير هذا النوع من المؤشرات فوراً إلى المعالج Bolt الذي يقوم بإضافة الوسمة الزمنية، وذلك يتم دون أن نجري عليها أي عملية تجميع (جعلنا معدل أخذ العينات 10 ثوانٍ).
2. المؤشرات الحيوية غير الحرجة: وتتغير قيم تلك المؤشرات بتواتر بطيء، مثل تواتر تغير درجة حرارة جسم الإنسان، وفي هذه الحالة سيتم تمرير هذا النوع من المؤشرات إلى المعالج Bolt الذي يجري عملية التجميع، ونستخدم تابع التجميع Average، حيث وضعنا معدل أخذ العينات 10 ثوانٍ، وبالتالي سيقوم هذا المعالج Bolt بتجميع 60 قيمة خلال 10 دقائق، ثم تحسب المتوسط الحسابي للعينات، وأخيراً ترسل قيمة المتوسط كعينة واحدة كل 10 دقائق للمعالج Bolt الذي يقوم بإضافة الوسمة الزمنية.

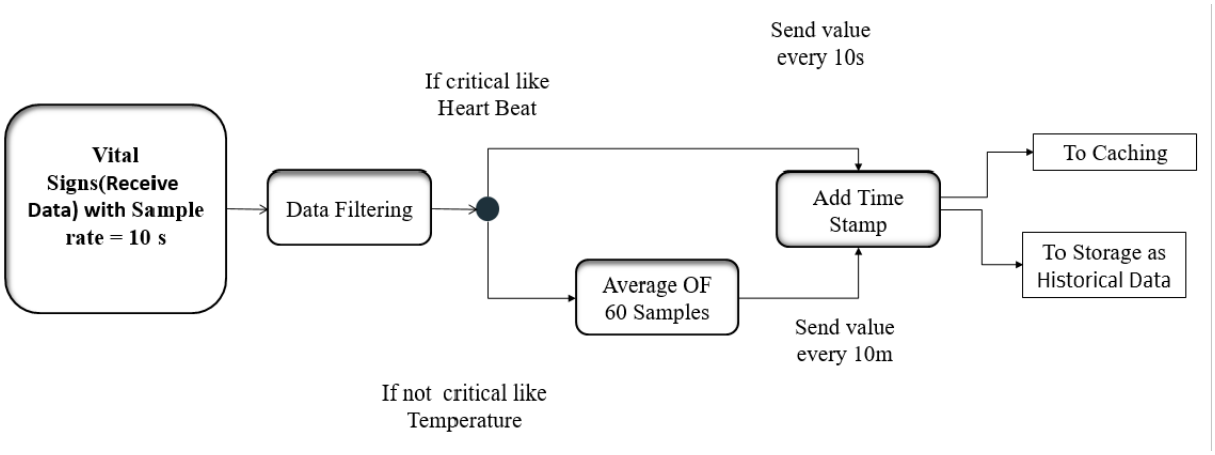
4-3-3 عملية إضافة الوسمة الزمنية أو الطابع الزمني Time Stamp للمؤشر الحيوي.

4-3-4 عملية تخزين بيانات المريض:

وهنا نميز حالتين:

1. التخزين المؤقت للبيانات في Redis، بهدف عرضها في الزمن الحقيقي.
2. التخزين الدائم للبيانات في Apache Cassandra، وذلك بهدف الاستفادة منها كبيانات تاريخية لأغراض تحليل البيانات الضخمة لاحقاً.

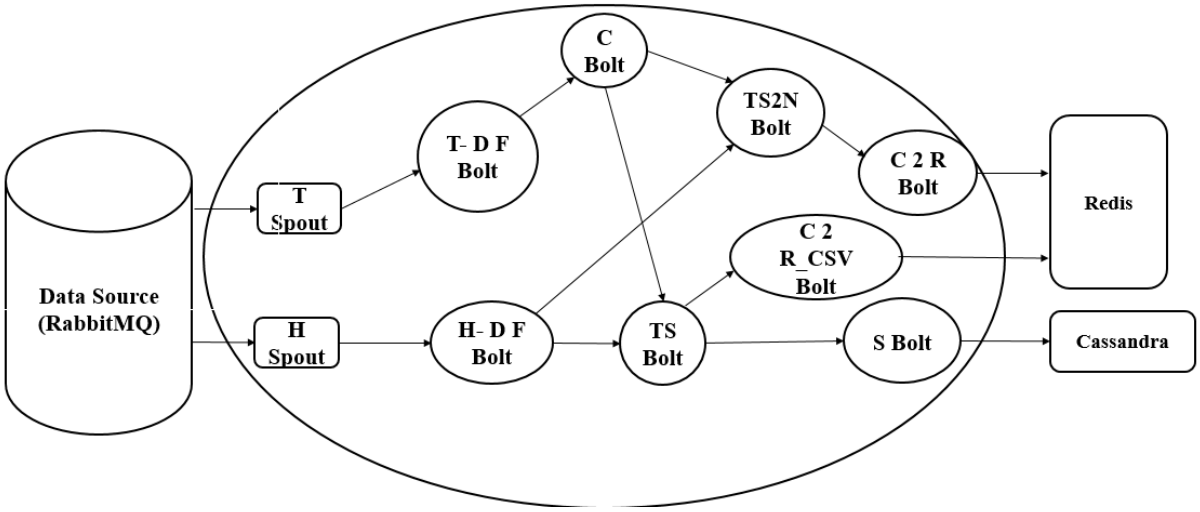
ويبين الشكل(4-2) المخطط الكتلي لمنطق المعالجة المتبع في Apache Storm.



الشكل(4-2): المخطط الكتلي لمنطق المعالجة المتبع في Apache Storm

4-4 التمثيل التقني للطوبولوجيا Storm Topology:

يبين الشكل(4-3) التمثيل التقني للطوبولوجيا التي تمّ تحقيقها، و هي تتكون من اثنين من مصادر البيانات Spouts، الأول وظيفته استقبال معطيات درجة الحرارة من وسيط الرسائل، و الثاني وظيفته استقبال معدل نبضات القلب، بمعنى أننا نمتلك هنا مسارين مستقلين، ثم لدينا المعالج DF Bolt ووظيفته تصفية البيانات، و من ثم المعالج C Bolt و يقوم بعملية تطبيق تابع تجميع Average بالنسبة للمؤشرات غير الحرجة، و أما المعالج TS Bolt فيقوم بإضافة الوسمة الزمنية أو الطابع الزمني Time Stamp للمؤشر الحيوي، و أيضاً يرسل البيانات برسالة واحدة بصيغة CSV إلى المعالج التالي، و بالمقابل فإن المعالج TS2N يقوم بإضافة الوسمة الزمنية أو الطابع الزمني Time Stamp للمؤشر الحيوي، و لكنه يرسل البيانات ضمن أربع Tuple منفصلة، ليتم تخزينها في Redis بصيغة مبسطة تلائم العرض على Node-Red، و لدينا المعالج C2R Bolt فيقوم بالتخزين المؤقت للبيانات في قاعدة البيانات Redis، أما المعالج C2R_CSV Bolt فيقوم بالتخزين المؤقت للبيانات في Redis بالصيغة CSV لاستخدامها في أغراض التحليل مستقبلاً، أما المعالج S Bolt فإنه يقوم بتخزين البيانات التاريخية في Apache Cassandra.



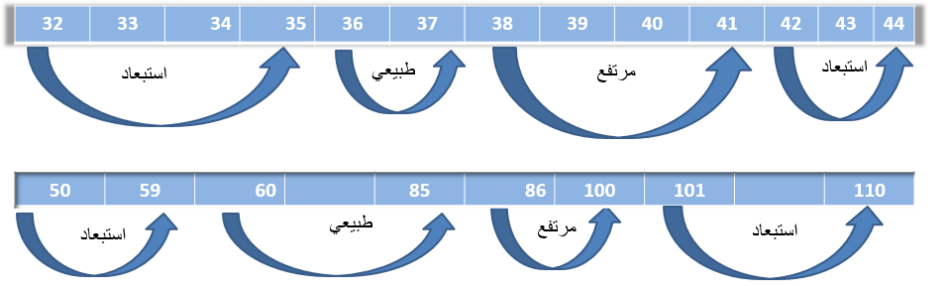
الشكل(4-3): التمثيل التقني للطوبولوجيا في Apache Storm

4-5 التحقيق الفعلي للبنية المقترحة:

4-5-1 تحقيق المنتج:

في البداية قمنا ببناء محاكي برمجي بلغة جافا، وهذا المحاكي يولد قيم المؤشرات الحيوية بشكل عشوائي، وذلك ضمن مجالات فرضية لقيم تحاكي القيم الحقيقية في جسم الانسان (لفئة عمرية معينة من البالغين، وضمن شروط صحية معينة)، ثم يرسل هذه القيم مضافاً إليها معرّف المريض، وبروتوكول التراسل المستخدم بصيغة CSV إلى الأداة RabbitMQ على المنفذ 15672، وقد تم توليد قيم المؤشرات الحيوية بشكل عشوائي ضمن مجالات محددة تم توضيحها في الشكل(4-4).

```
24    int START_T = 32;//  الحد الأدنى لدرجة الحرارة
25    int END_T = 44;//  الحد الأعلى لدرجة الحرارة
26    int START_H = 50;//  الحد الأدنى لمعدل نبضات القلب
27    int END_H = 110;//  الحد الأعلى لمعدل نبضات القلب
```



الشكل(4-4): تصنيف مجالات قيم المؤشرات الحيوية

و قد تم اختيار الصيغة CSV لاعتبارات تتعلق بسهولة معالجة وتحليل البيانات الضخمة من جهة، و طبيعة الشبكات التي صُممت للتعامل مع تطبيقات انترنت الأشياء و أهمها[40]:

1- تعتبر الصيغة الأفضل عندما ترسل كماً هائلاً من البيانات، وعندما يكون عرض الحزمة الترددية Bandwidth مشكلة، وبالتالي تعتبر مناسبة للشبكات المقيدة ومحدودة الموارد.

2- معالجة أسرع للبيانات: حيث لا تحتاج صيغة CSV إلا إلى عملية Split، أما الصيغ الأخرى مثل JSON فتتطلب عملية تفسير لل Syntax.

3- تحليل أسرع للبيانات: فتقنيات البيانات الضخمة مثل Hadoop متكاملة مع الصيغة CSV، في حين أنها تحتاج لوظائف خاصة تجري عملية Parsing عند التعامل مع صيغ أخرى مثل JSON مثلاً.

وأما بالنسبة لخرج هذا الكود الذي يحاكي منتج البيانات فهو موضح بالشكل(4-5)، وقد تم بناؤه على مبدأ القدح الزمني Time Triggered System .

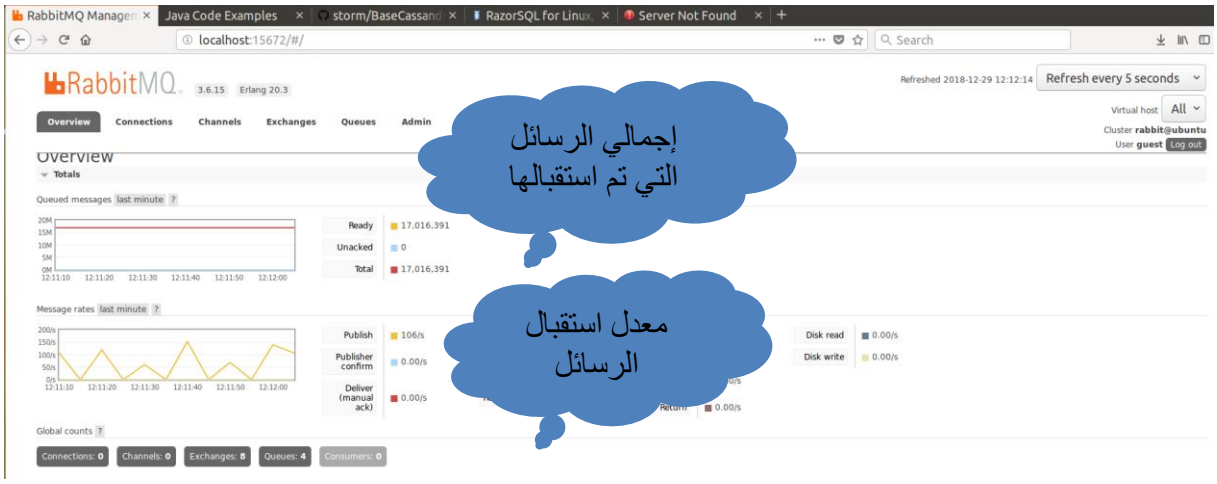
```
Output - MQTT_phao_proucer_1 (run)
 creating connection factory.....
 creating new connection.....
5672
localhost/127.0.0.1
opening channel
Temperature Generated Now : 37
HeartBeat rate Generated Now : 76
Patient for this Vital  ID is : 457
Sending  Body Temperature Data Now :'Re(MQTT),457,37'
Sending Heart rate Data Now : 'Re(MQTT),457,76'
 creating connection factory.....
 creating new connection.....
5672
localhost/127.0.0.1
opening channel
Temperature Generated Now : 36
HeartBeat rate Generated Now : 108
Patient for this Vital  ID is : 245
Sending  Body Temperature Data Now :'Re(MQTT),245,36'
Sending Heart rate Data Now : 'Re(MQTT),245,108'
```

الشكل(4-5): خرج منتج البيانات

4-5-2 تنصيب و إعداد وسيط الرسائل:

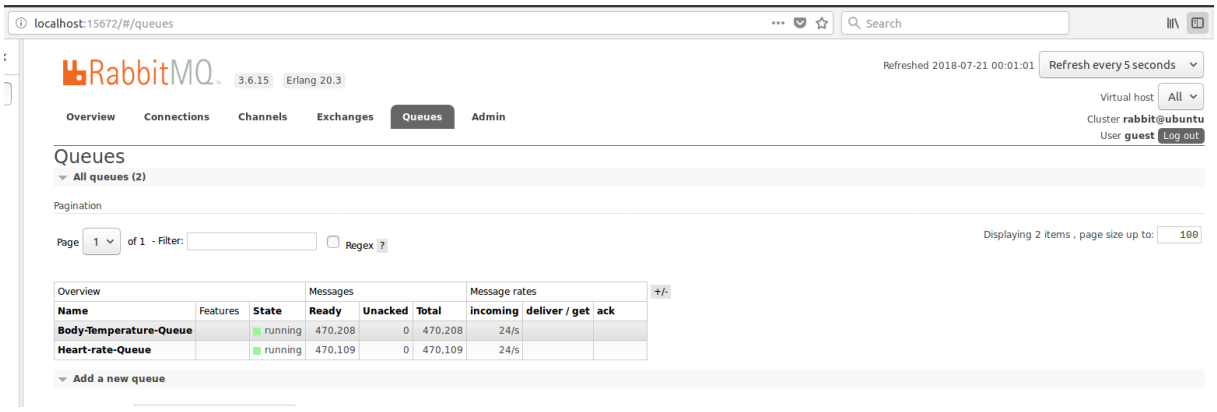
تم تنصيب وإعداد وسيط الرسائل RabbitMQ كما في الشكل(4-6)، ونلاحظ أن واجهة المستخدم تتضمن مخططين، فالمخطط الأول يعبر عن إجمالي الرسائل التي تم استقبالها، والتي تعتبر بحالة جاهزية للمعالجة، كما في المثال نجد أن

العدد بلغ حوالي 17 مليون رسالة، أما المخطط الثاني فيعبر عن معدل استقبال رسائل المعطيات، وفي مثالنا نجد أنه بلغ 106 رسالة في واحدة الزمن، وهذا ما يبينه الشكل(4-6):



الشكل(4-6): واجهة وسيط الرسائل RabbitMQ

أما عندما نفتح على التبويب Queues، فيتم عرض معلومات الأرتال، حيث نجد الرتلين Body-Temperature-Queue و Heart-rate-Queue الموضحين في الشكل(4-7) ، حيث نجد المعلومات الخاصة بكل رتل على حده.



الشكل(4-7): تبويب الأرتال في RabbitMQ

حقيقة إن وسيط الرسائل RabbitMQ يقوم بتتبع المنتج من حيث معدل استقبال المعطيات، حيث نستطيع التحكم بمعدل الاستقبال من خلال تعديل القيمة التي يتم عندها قدح المؤقت الزمني في المنتج، وذلك بتعديل قيمة البارامتر الثالث في التعليمة timer.schedule (وهي القيمة التي تعبر عن معدل تكرار عمليتي التوليد والارسال)، وقد قمنا باستخدام هذا المعامل في عملية التقييم والمقارنة بين البروتوكولين من حيث الأداء.

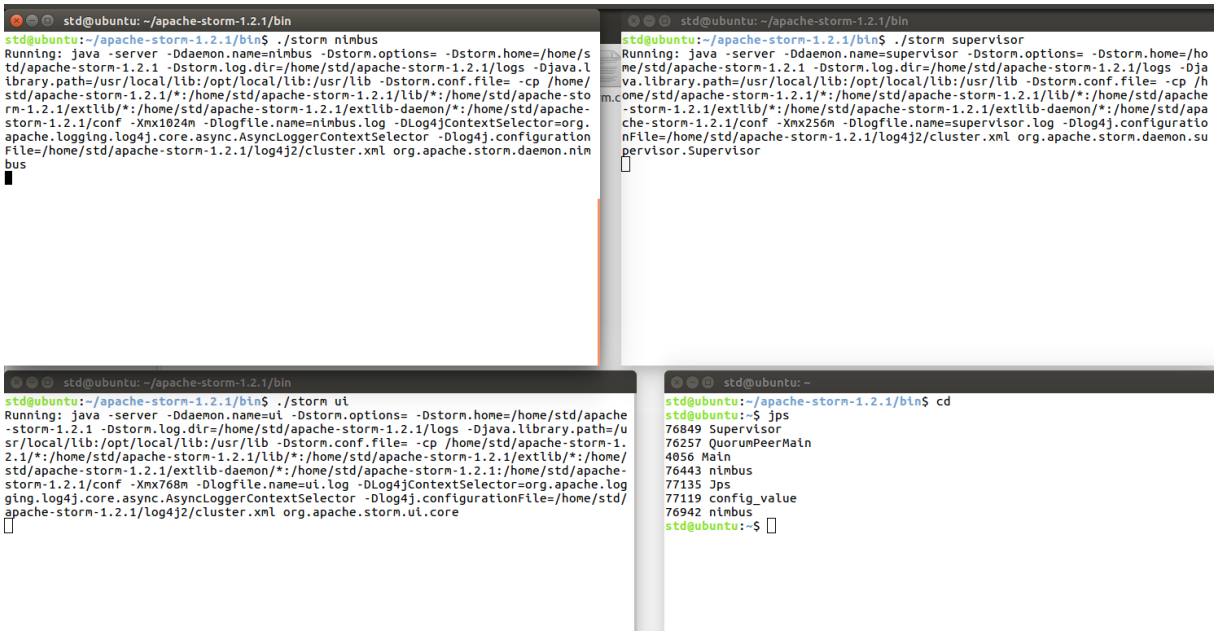
4-5-3 تنصيب و إعداد معالج البيانات الضخمة Apache Storm:

بعد تنصيب هذه الأداة لابد من ضبط الاعدادات الخاصة، بحيث تعمل على عقدة واحدة Single Node، حيث توجد هذه الاعدادات ضمن الملف storm.yaml الموجود في المجلد config، حيث حددنا هنا معلومات إضافية مثل مسار تخزين معطيات Storm، و رقم المنفذ الذي سيستخدمه، و في هذه النقطة واجهتني مشكلة، و هي أن المنفذ المتعارف عليه لهذه الأداة هو 8080، و لكن عند ضبطه كما هو متعارف عليه، كان يرسل رسالة خطأ تفيد بأن المنفذ مشغول، لذلك غيرت رقم المنفذ ووضعته 8072، و بذلك حللت المشكلة، و حقيقة الموضوع أن مثل هذه البرمجيات مخصصة للعمل ضمن عنقود Cluster، و عندها سوف تتغير محتويات هذا الملف، و هذ ما يبينه الشكل (4-8) :

```
########### These MUST be filled in for a storm configuration
storm.zookeeper.servers:
      - "localhost"
#storm.zookeeper.port: 2182

##  nimbus.seeds: ["localhost"]
storm.local.dir: "/home/std/apache-storm-1.2.1/data"

nimbus.host: "localhost"
##nimbus.thrift.port:  6627
ui.port : 8072
## Eng Osama Ebrahem Storm Port
# java.library.path: "/usr/lib/jvm"
supervisor.slots.ports:

- 6700

- 6701

- 6702

- 6703
```

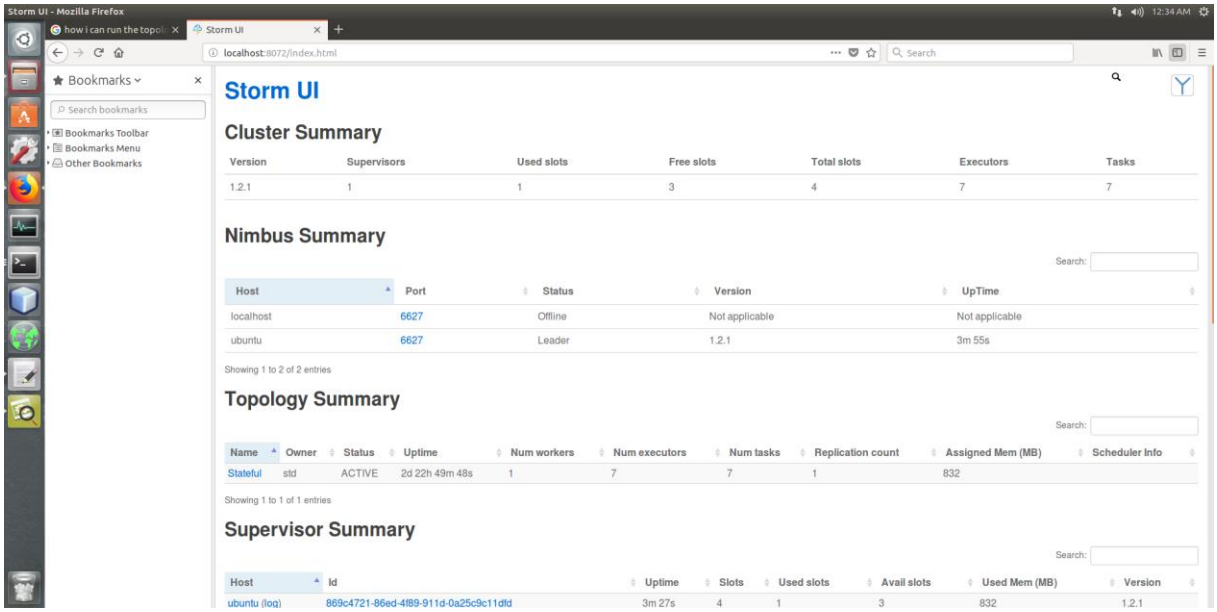
الشكل (4-8): أهم محتويات الملف Storm.yaml

وقبل تشغيل storm UI لدينا بعض الخدمات و الDaemons لابد من إقلاعها، حيث نبدأ بتشغيل خدمة zookeeper من خلال تشغيل الأمر zkServer الموجود في الملف bin، ثم نشغل الDaemons الخاص بالnimbus و الذي يمثل السيد، ثم نشغل الDaemons الخاص بالsupervisor و الذي يمثل العبيد أو العمال، و أخيراً نشغل الDaemons الخاص بواجهة المستخدم، ومن المهم الإشارة إلى أن عمل واجهة Apache Storm يتطلب عمل جميع الDaemons معاً، و أي خلل يسبب توقفها، و هذا ما يبينه الشكل(4-9):



الشكل(4-9): الDaemons المتزامنة لتشغيل Storm UI

ثم نقلع واجهة المستخدم من المتصفح على رقم المنفذ الذي كنا قد حددناه سابقاً في الملف storm.yaml، فتظهر الواجهة كما الشكل(4-10):



الشكل(4-10): Storm UI

عند بناء Storm Topology لابد من تضمين ملفات الJAR الأساسية في NetBeans، والتي تحوي الصفوف المجردة التي تمكننا من بناء Topology ففي البداية نضمن storm-core-1.2.1، و هنا لابد من الانتباه إلى توافق هذا الملف مع نسخة Apache Storm التي تم تنصيبها، لأن المبرمج الذي قد يغفل مثل هذا التفصيل، سيعاني كوده من مشكلة في

توافق أسماء المكتبات والصفوف، فإذا عدنا للشكل(4-8)، نلاحظ أن النسخة التي تم تنصيبها 1.2.1، و تلك النقطة حساسة و لابد من مراعاتها، ثم نقوم بتضمين متطلبات ربط Storm مع الأداة RabbitMQ، فنضمن الملف storm-rabbitmq-1.0.0، و الملف rabbitmq-java-client-bin-3.5.4 و يتضمن صفوف جافا الأساسية التي تتيح التعامل مع RabbitMQ، وسنقوم بشرح و توصيف مبدأ عمل عناصر الStorm Topology.

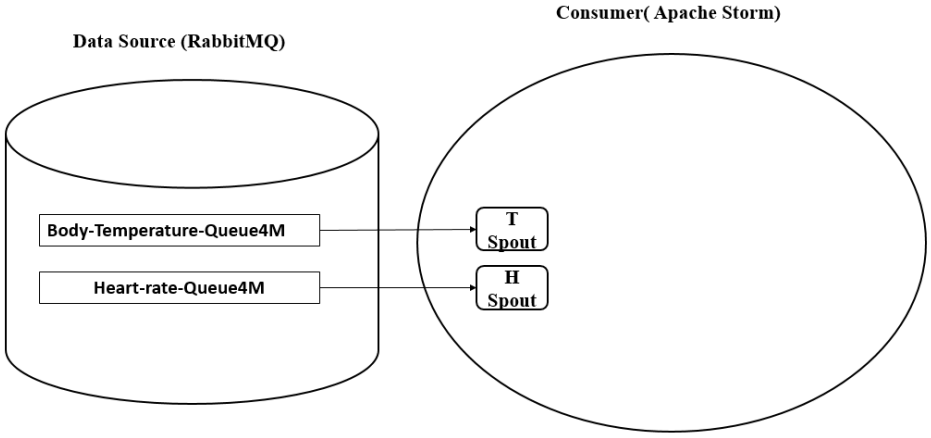
4-5-4 تحقيق الطوبولوجيا في Storm ومكوناتها:

1- مصادر البيانات Spouts:

يقتصر دورها على استقبال المعطيات من وكيل الرسائل RabbitMQ، حيث قمنا ببناء مصدري بيانات مستقلتين هما T Spout,H Spout الموضحان في الشكل (4-11)، ولنوضح دور كل منهما:

A. مصدر البيانات T Spout: يستقبل بيانات درجة الحرارة من الرتل Body-Temperature-Queue4M الموضح في الشكل(4-11)، و يمثّل صفاً Class في جافا يتضمن مكتبات كلتا الأداتين.

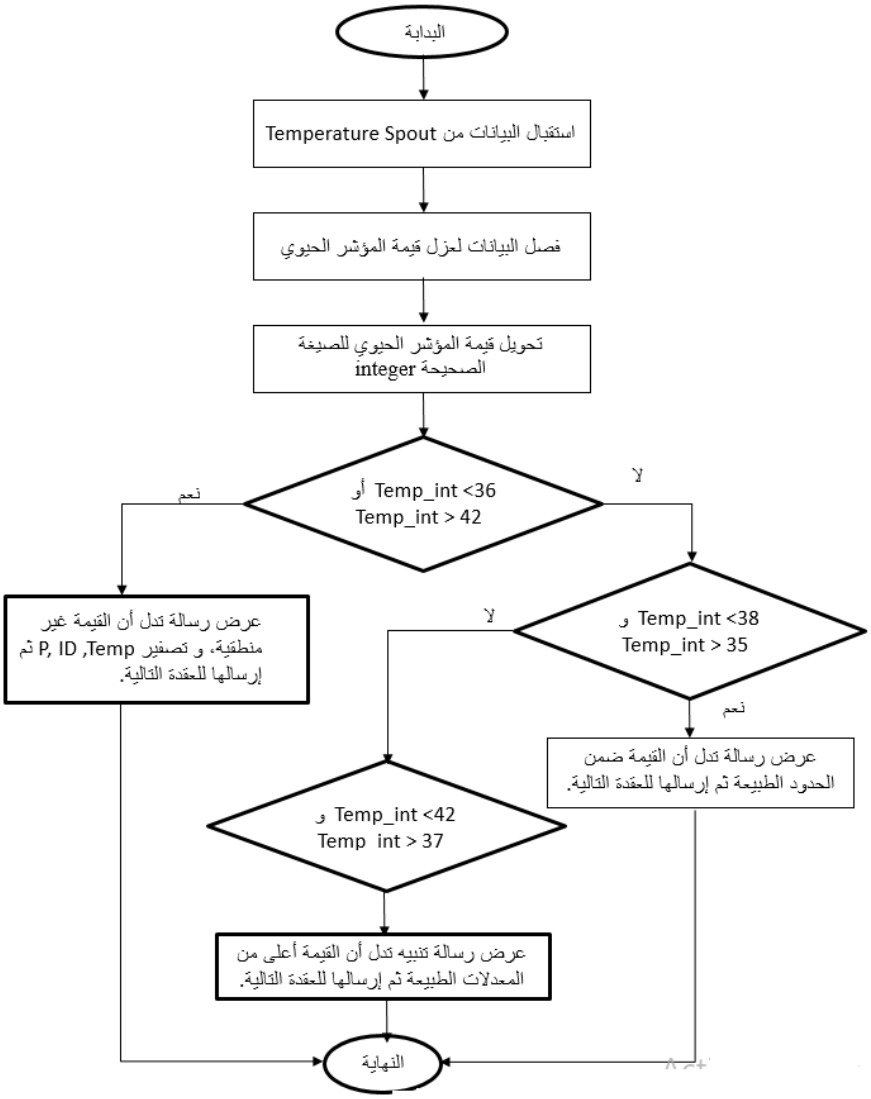
B. مصدر البيانات H Spout: يستقبل بيانات معدل نبضات القلب من الرتل Heart-rate-Queue4M، والشكل(4-11) يوضح هذه العلاقة بشكل واضح .



الشكل(4-11): المخطط الكتلي لارتباط مصادر البيانات في Storm مع الأرتال في RabbitMQ

2- معالج تصفية البيانات Data Filtering Bolt:

ويتم ضمن هذا المعالج استبعاد القيم غير المنطقية، أو تحديد قيمة قراءة مفقودة، وكذلك تحديد حالة المؤشر الحيوي، واتخاذ القرار المناسب، وتتم عملية التصفية لدرجات الحرارة وفقاً للمخطط التدفقي الموضح في الشكل(4-12)، وبالنسبة لتصفية قيم معدل نبضات القلب، فتتم باستخدام الخوارزمية نفسها تبعاً للمجالات الموضحة في الشكل(4-4).



الشكل(4–12): خوارزمية تصفية البيانات

حيث Temp_int متحول يعبر عن قيمة درجة الحرارة ممثلة بالصيغة الصحيحة.

ولنشرح الحالات الأساسية التي تعالجها الخوارزمية السابقة عندما تم تحقيقها بشكل فعلي كما يلي:

1- تحديد القيم الخاطئة: ويتم تصفير القيمة قبل ارسالها للمعالج التالي، وارسال رسالة خطأ ليتم تفقد حالة الحساس أو جهاز المراقبة الخاص بذلك المؤشر الحيوي، فمن أجل درجة حرارة جسم الإنسان يكون خرج هذا المعالج موضحاً في الشكل(4–13)، والذي يوضح أنه من أجل القيمة 33 والتي تعتبر غير منطقية، فإن المعالج ميز بأن القيمة خارج المجال الطبيعي، فحتماً لدينا مشكلة ما، وبالتالي قام بإرسال تحذير لتفقد حالة الحساس وجاهزيته، ثم قام بتصفير الثنائية (معرف المريض، قيمة المؤشر الحيوي).

```
54188 [Thread-26-Temperature-Filtering-op-executor[22 22]] INFO  o.a.s.d.executor - Processing received message FOR 22 TUPLE: source: Temperature-spout:23, str
I Will Separate Patient ID About  Patient Body Temperature  now  .. .
 The Patient Data recieved is  : 200,33
 The Body Temperature of Patient recieved is  : 33
 The Patient ID  recieved is  : 200
I Will Check Body Temperature Data now  .. .
 This Body Temperature Value for this Patient is Out Of Normal Range .. .  check this sensor please
54189 [Thread-26-Temperature-Filtering-op-executor[22 22]] INFO  o.a.s.d.task - Emitting: Temperature-Filtering-op default [0,0]
```

الشكل(4-13): خرج معالج تصفية البيانات بحال استقبال قيمة غير منطقية لدرجة الحرارة

وبالمثل فإن الشكل(4-14) يوضح خرج هذا المعالج من أجل قيمة خاطئة لمعدل نبضات القلب.

```
54198 [Thread-34-Heart-Filtering-op-executor[18 18]] INFO  o.a.s.d.executor - Processing received message FOR 18 TUPLE: source: HeartRate-spout:19, stream: de
I Will Separate Patient ID About  Patient Heart Rate now  .. .
 The Patient Data recieved is  : 200,50
 The Heart Rate of Patient  recieved is  : 50
 The Patient ID  recieved is  : 200
I Will Check Heart Rate Data now  .. .
 This Heart Rate Value for this  Patient is Out Of Normal Range .. .  check the sensor please
54200 [Thread-34-Heart-Filtering-op-executor[18 18]] INFO  o.a.s.d.task - Emitting: Heart-Filtering-op default [0,0]
```

الشكل(4-14): خرج معالج تصفية البيانات بحال استقبال قيمة غير منطقية لمعدل نبضات القلب

2- تحديد القيم الأعلى من المجال الطبيعي: وهنا يتم تمرير القيمة، وارسال رسالة تحذيرية.

ففي الشكل(4-15) نلاحظ أن قيمة درجة الحرارة التي تم استقبالها عبر الSpout بلغت 40، و التي تعتبر أعلى من المعدلات الطبيعية، و تشير لإصابة المريض بحالة التهابية أو جرثومية ما، و في هذه الحالة تقوم العقدة بتمرير القيمة للعقدة التي تليها، مع تنبيه يفيد بارتفاع درجة الحرارة ليتم مراعاته لاحقاً.

```
56129 [Thread-54-Temperature-Filtering-op-executor[21 21]] INFO  o.a.s.d.executor - Processing received message FOR 21 TUPLE: source: Temperature-spout:23, stre
I Will Separate Patient ID About  Patient Body Temperature  now  .. .
 The Patient Data recieved is  : 200,40
 The Body Temperature of Patient recieved is  : 40
 The Patient ID  recieved is  : 200
I Will Check Body Temperature Data now  .. .
 This Body Temperature Value is above of  Normal Range .. . be careful please
56130 [Thread-54-Temperature-Filtering-op-executor[21 21]] INFO  o.a.s.d.task - Emitting: Temperature-Filtering-op default [200,40]
```

الشكل(4-15): خرج معالج تصفية البيانات بحال استقبال قيمة لدرجة الحرارة أعلى من المعدلات الطبيعية

أما في الشكل(4-16) فنلاحظ أن القيمة 90 تعتبر بداية لتسارع معدل نبضات القلب، وبهذه الحالة يتم تمرير القيمة، وارسال رسالة تحذيرية تفيد بارتفاع معدل نبضات القلب.

```
55094 [Thread-48-Heart-Filtering-op-executor[16 16]] INFO  o.a.s.d.executor - Processing received message FOR 16 TUPLE: source: HeartRate-spout:19, stream: defa
I Will Separate Patient ID About  Patient Heart Rate now  .. .
 The Patient Data recieved is  : 200,90
 The Heart Rate of Patient  recieved is  : 90
 The Patient ID  recieved is  : 200
I Will Check Heart Rate Data now  .. .
  This Heart Rate Value is above of  Normal Range .. .  be careful please
55095 [Thread-48-Heart-Filtering-op-executor[16 16]] INFO  o.a.s.d.task - Emitting: Heart-Filtering-op default [200,90]
```

الشكل(4–16): خرج معالج تصفية البيانات بحال استقبال قيمة لمعدل نبضات القلب أعلى من المعدلات الطبيعية

3- تحديد القيم الطبيعية: ويتم قبولهم دون أي إجراء إضافي، وهذا موضح في الشكل(4–17) .

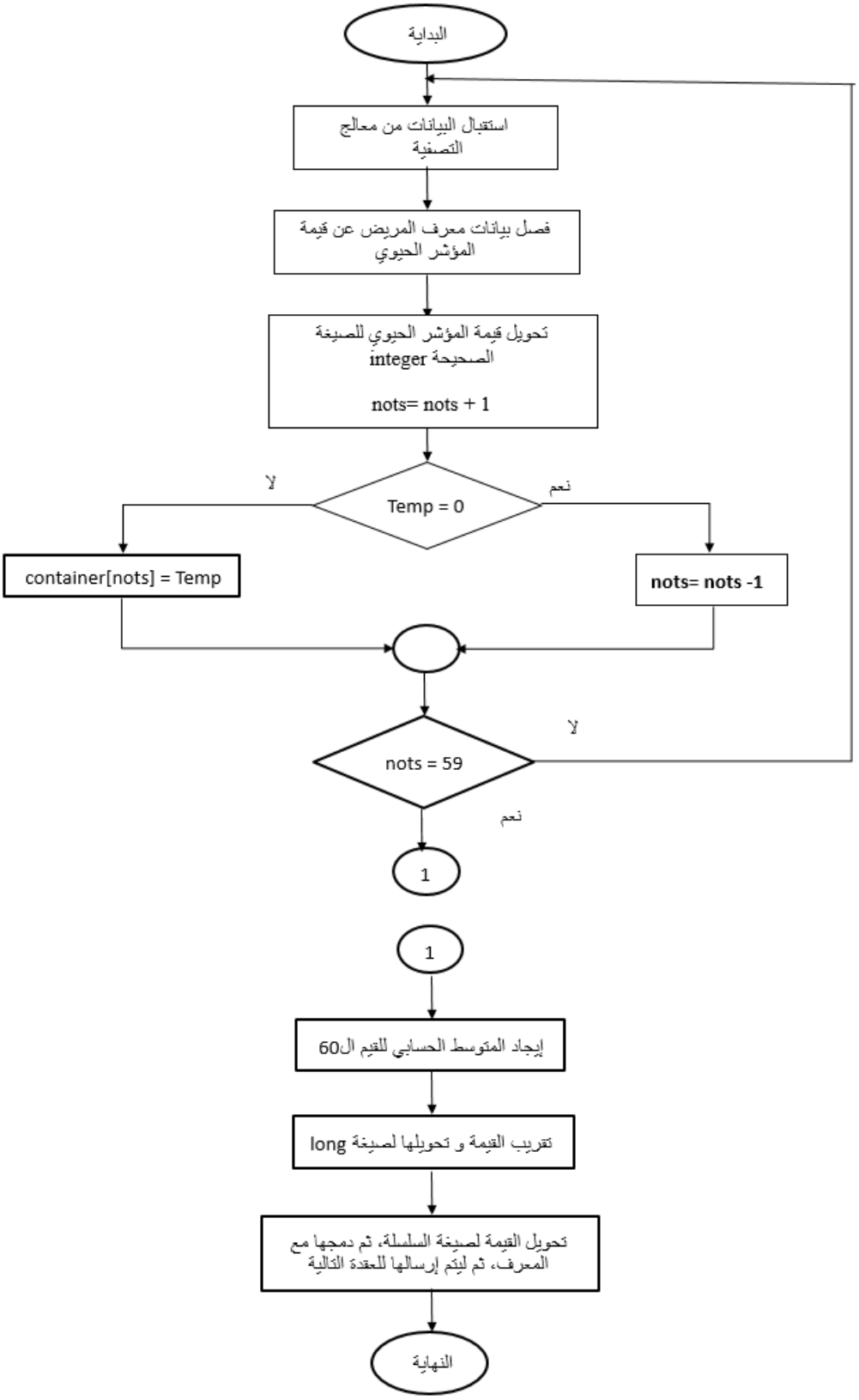
```
57141 [Thread-54-Temperature-Filtering-op-executor[21 21]] INFO  o.a.s.d.executor - Processing received message FOR 21 TUPLE: source: Temperature-spout:23, stre
I Will Separate Patient ID About  Patient Body Temperature  now  .. .
 The Patient Data recieved is  : 200,36
 The Body Temperature of Patient recieved is  : 36
 The Patient ID  recieved is  : 200
I Will Check Body Temperature Data now  .. .
 This Body Temperature Value for this Patient is Normal
57148 [Thread-54-Temperature-Filtering-op-executor[21 21]] INFO  o.a.s.d.task - Emitting: Temperature-Filtering-op default [200,36]
```

```
56084 [Thread-34-Heart-Filtering-op-executor[18 18]] INFO  o.a.s.d.executor - Processing received message FOR 18 TUPLE: source: HeartRate-spout:19, stream: defa
I Will Separate Patient ID About  Patient Heart Rate now  .. .
 The Patient Data recieved is  : 200,72
 The Heart Rate of Patient  recieved is  : 72
 The Patient ID  recieved is  : 200
I Will Check Heart Rate Data now  .. .
  This Heart Rate Value for this Patient is Normal
56084 [Thread-34-Heart-Filtering-op-executor[18 18]] INFO  o.a.s.d.task - Emitting: Heart-Filtering-op default [200,72]
```

الشكل(4–17): خرج معالج تصفية البيانات بحال استقبال قيم ضمن المعدلات الطبيعية

3– معالج التجميع Consolidation Bolt:

يتفحص هذا المعالج عدد عينات درجة حرارة الجسم التي تم تخزينها مع كل تنفيذ لعقدة التصفية، والذي نعبر عنه بقيمة تسمى (number of Temperature samples) nots فعندما يصبح عدد العينات 60، سيتم إيجاد المتوسط الحسابي للعينات، والشكل(4–18) يوضح المخطط التدفقي لهذا المعالج.


البداية
استقبال البيانات من معالج التصفية
فصل بيانات معرف المريض عن قيمة المؤشر الحيوي
تحويل قيمة المؤشر الحيوي للصيغة الصحيحة integer
nots= nots + 1
Temp = 0
لا
نعم
container[nots] = Temp
nots= nots -1
nots = 59
لا
نعم
1
1
إيجاد المتوسط الحسابي للقيم ال60
تقريب القيمة و تحويلها لصيغة long
تحويل القيمة لصيغة السلسلة، ثم دمجها مع المعرف، ثم ليتم إرسالها للعقدة التالية
النهاية

الشكل(4-18): المخطط التدفقي لمعالج تجميع درجة الحرارة

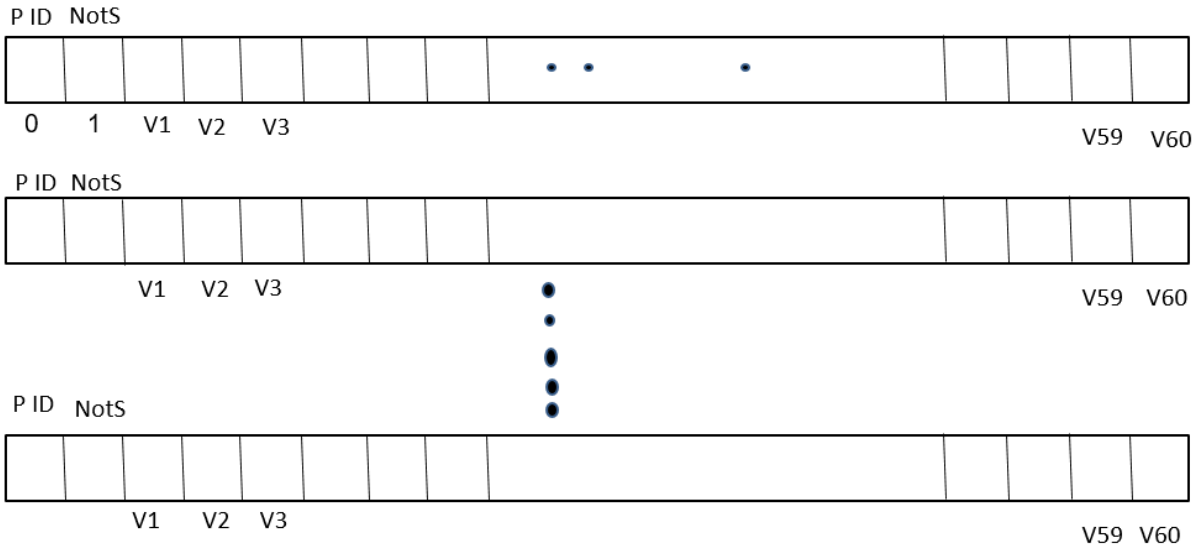
حيث Container تمثل مصفوفة تخزين القيم ال60، و القيمة البدائية لعدد العينات = 0

وعند تنفيذ هذا المعالج بشكل فعلي فإن الخرج ضمن Storm موضح في الشكل(4-19):

```
Output
Generator4M (run) ×  ha_cassandra_Redis (run) ×
total= 2117.0
total= 2154.0
total= 2195.0
total= 2234.0
total= 2275.0
 body temperature ready to send and this final value :37.91666666666664
 body temperature ready to send and this final value After convert to long :38
148873 [Thread-54-Consolidation-op-executor[13 13]] INFO  o.a.s.d.task - Emitting: Consolidation-op default [135,38]
148873 [Thread-54-Consolidation-op-executor[13 13]] INFO  o.a.s.d.executor - TRANSFERING tuple [dest: 6 tuple: source: Consolidation-op:13, stream: default, id
 the Avarage value with P_ID will emitting Now = 135,38
148875 [Thread-54-Consolidation-op-executor[13 13]] INFO  o.a.s.d.executor - BOLT ack TASK: 13 TIME: -1 TUPLE: source: Temperature-Filtering-op:26, stream: def
148875 [Thread-54-Consolidation-op-executor[13 13]] INFO  o.a.s.d.executor - Execute done TUPLE source: Temperature-Filtering-op:26, stream: default, id: {},
```

الشكل(4-19): خرج معالج تجميع درجة الحرارة

ملاحظة: إن المصفوفة السابقة تعالج حالة التجميع لبيانات مريض واحد، أما من أجل عدد ديناميكي من المرضى فقد تم تطوير بنية المصفوفة لتلائم حالة المرضى المتعددين، وهذا ما يوضحه الشكل(4-20):



الشكل(4-20): البنية المعدلة لمصفوفة التجميع والتي تراعي ديناميكية عدد المرضى

4- معالج إضافة الوسمة الزمنيةTS Blot :

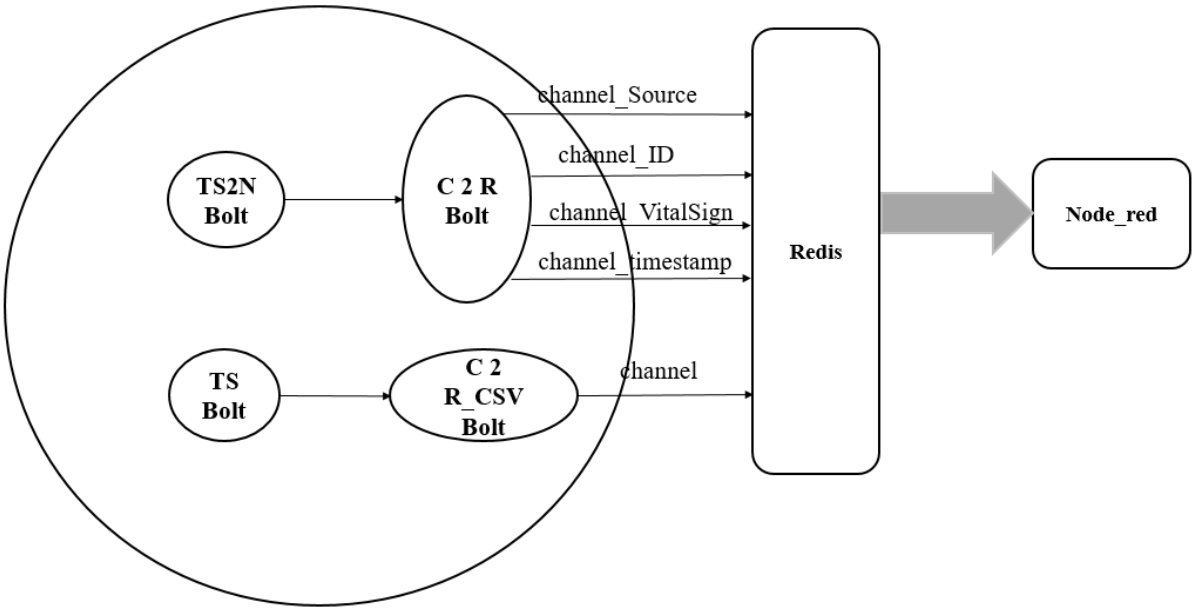
يضيف الوسمة الزمنية Time Stamp لقيمة المؤشر الحيوي، وفي بحثنا يضيف هذا المعالج بياناته اعتماداً على الساعة الفيزيائية للحاسب المضيف، والشكل(4-21) يوضح خرج هذا المعالج.

```
148901 [Thread-36-AddTimeStamp-op_T-executor[6 6]] INFO  o.a.s.d.executor - Processing received message FOR 6 TUPLE: source: Consolidation-op:13, stream: defau
I Will add time stamp now  .. .
 The Patient Data recieved is  : 135,38
 The Body Temperature with Time Stamp is : 38 At  2019-01-22 01:24:43.033
 The Time Stamp after convert to Date format  is : 2019-01-22 01:24:43.033
converted Date to String: 2019-01-22 01:24:43
 we will Send Body Temperature with Time Stamp now : 135,38,2019-01-22 01:24:43
148901 [Thread-36-AddTimeStamp-op_T-executor[6 6]] INFO  o.a.s.d.task - Emitting: AddTimeStamp-op_T default [135,38,2019-01-22 01:24:43]
```

الشكل(4-21): خرج معالج إضافة الوسمة الزمنية

5- معالج التخزين المؤقتC2R Bolt :

ينفذ عملية التخزين المؤقت في Redis من خلال القنوات الأربع المبينة في الشكل(4-22)، بهدف تهيئة البيانات لعرضها على Node-Red، أما الشكل(4-23) فيتضمن تعريف القنوات وربطها ضمن Redis Bolt.



الشكل(4-22): المخطط الكتلي لارتباط القنوات في Redis مع الأداة Node-Red

```
156    public static class RedisBolt implements IRichBolt {
157
158        //  protected String channel = "Storm_Redis";
159        protected String channel_Source = "Storm_Redis:Protocol";
160
161        protected String channel_ID = "Storm_Redis:Patient_ID";
162        protected String channel_VitalSign = "Storm_Redis:VitalSignValue";
163        protected String channel_timestamp = "Storm_Redis:timestamp";
164
165        protected OutputCollector collector;
166
167        protected JedisPool pool;
168
169        public RedisBolt() {
170        }
```

الشكل(4-23): مقطع الكود الذي يبين قنوات Redis

أما الشكل(4-24) فيمثل خرج هذا المعالج .

```
104873 [Thread-42-Caching-op_H-executor[9 9]] INFO  o.a.s.d.task - Emitting: Caching-op_H default [L]
104874 [Thread-42-Caching-op_H-executor[9 9]] INFO  o.a.s.d.task - Emitting: Caching-op_H default [66]
104878 [Thread-42-Caching-op_H-executor[9 9]] INFO  o.a.s.d.task - Emitting: Caching-op_H default [91]
104879 [Thread-42-Caching-op_H-executor[9 9]] INFO  o.a.s.d.task - Emitting: Caching-op_H default [2019-04-27 01:46:31]
```

الشكل(4-24): خرج معالج تخزين البيانات المؤقتة

6- معالج تخزين البيانات التاريخيةS Bolt :

ينفذ هذا المعالج عملية التخزين الدائم للبيانات في Apache Cassandra، كما هو موضح في الشكل(4-25)، والذي يمثل تقريراً بسيطاً للعملية التي تتم في تلك اللحظة، حيث نجد أن الرقم 30704 يمثل رقم العملية الحالية، أما Historical_Data_Storing -op_H-executor فهو منفذ غرض تخزين البيانات التاريخية، في حين أن ما ذكر بعد Emitting و اسم الغرض، فهو وحدة البيانات المرسلة من هذا المعالج إلى Apache Cassandra ، و نلاحظ أنها أرسلت بصيغة CSV ، و في السطر الخامس نجد BOLT ack TASK ، و تعني أن عملية المصادقة على وصول البيانات من معالج إضافة الوسمة الزمنية تمّت بنجاح.

```
30704 [Thread-40-Historical_Data_Storing -op_H-executor[20 20]] INFO
o.a.s.d.task - Emitting: Historical_Data_Storing -op_H default [1, 359, L,
97, 2019-06-01 06:11:13]

30704 [Thread-40-Historical_Data_Storing -op_H-executor[20 20]] INFO
o.a.s.d.executor - BOLT ack TASK: 20 TIME: -1 TUPLE: source:
AddTimeStamp-op_H:3, stream: default, id: {}, [L, 359, 97, 2019-06-01
06:11:13]
```

الشكل(4-25): خرج معالج تخزين البيانات التاريخية

4-5-5 تحقيق تخزين البيانات الموقتة في Redis:

يتم تخزين البيانات بصيغة بسيطة على شكل أربع قنوات مستقلة، ليتم لاحقاً ربط كل قناة في Redis مع Data Source واحد في الأداة Node-Red، كما هو موضح في الشكل(4-26)، ففي السطر الأول نجد إشعار ورود رسالة جديدة، أما السطر الثاني فنجد اسم القناة، و في حين أن السطر الثالث يمثل البيانات الفعلية.

1) "message" 2) "Storm_Redis:Protocol" 3) "Re(AMQP)" 1) "message" 2) "Storm_Redis:Protocol" 3) "Re(MQTT)"	1) "message" 2) "Storm_Redis:VitalSignValue" 3) "95" 1) "message" 2) "Storm_Redis:VitalSignValue" 3) "38"
1) "message" 2) "Storm_Redis:Patient_ID" 3) "273" 1) "message" 2) "Storm_Redis:Patient_ID" 3) "354"	1) "message" 2) "Storm_Redis:timestamp" 3) "2019-06-10 17:51:45" 1) "message" 2) "Storm_Redis:timestamp" 3) "2019-06-10 17:51:46"

الشكل(4-26): صيغة البيانات المخزنة في Redis

4-5-6 تحقيق تخزين البيانات التاريخية في Apache Cassandra:

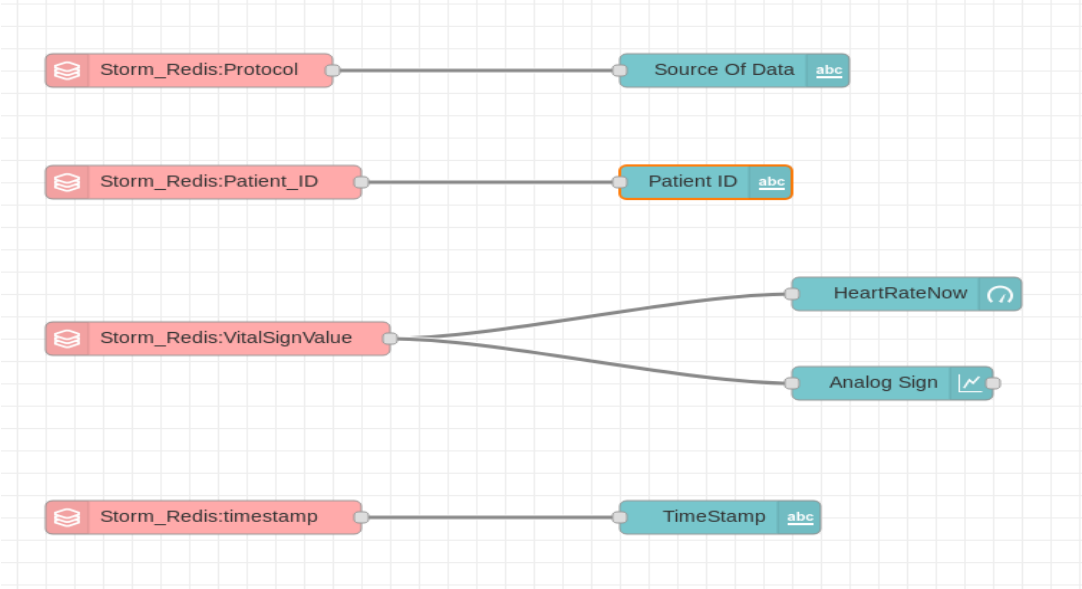
يتم تخزين البيانات على شكل جدول مكون من خمسة أعمدة، تتضمن نفس المعلومات المخزنة في Redis، ولكن مع إضافة عمود يدل على ترتيب التوضع ضمن Cassandra، وهذا موضح في الشكل(4-27).

```
std@ubuntu: ~
File Edit View Search Terminal Help
cqlsh>  USE Cassandra_KEYSPACE;
cqlsh:cassandra_keyspace> select * from TheSimpleEHR  ;

 row_id | patient_id | patient_pos | patient_vitalsign | timestamp
--------+------------+-------------+-------------------+---------------------
     23 |        410 |           L |                90 | 2019-06-01 06:11:56
    114 |        126 |           L |                62 | 2019-04-30 04:09:12
    660 |         71 |    Re(AMQP) |                92 | 2019-04-29 00:09:59
     53 |         29 |           L |                60 | 2019-06-01 06:12:46
    110 |        463 |           L |                83 | 2019-06-01 06:14:12
     91 |        237 |           L |                70 | 2019-06-01 06:13:42
    128 |        207 |    Re(AMQP) |                90 | 2019-04-29 13:18:17
    363 |        237 |    Re(AMQP) |                94 | 2019-04-29 00:02:39
    251 |        157 |    Re(AMQP) |                84 | 2019-04-29 00:00:03
    310 |        490 |    Re(AMQP) |                61 | 2019-04-29 00:01:20
    247 |        148 |    Re(AMQP) |                71 | 2019-04-28 23:59:59
    214 |        141 |    Re(AMQP) |                84 | 2019-04-28 23:59:14
    429 |         47 |    Re(AMQP) |                63 | 2019-04-29 00:04:18
    117 |        149 |           L |                86 | 2019-04-30 04:09:15
    547 |        344 |    Re(AMQP) |                78 | 2019-04-29 00:07:06
    144 |        377 |    Re(AMQP) |                92 | 2019-04-29 13:18:36
    718 |        489 |    Re(AMQP) |                83 | 2019-04-29 00:11:36
    567 |        310 |    Re(AMQP) |                73 | 2019-04-29 00:07:38
    120 |         96 |           L |                67 | 2019-04-30 04:09:19
```

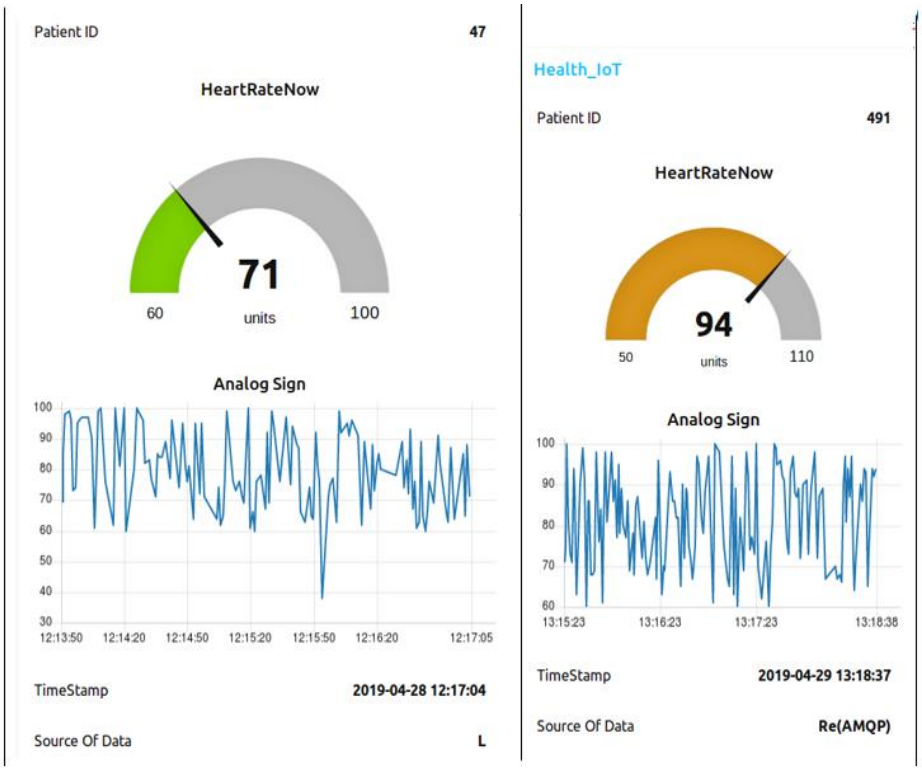
الشكل(4-27): صيغة البيانات المخزنة في Cassandra

4-5-7 تحقيق الإظهار على Node-Red:

يتم عرض بيانات المريض بصيغة بيانية بالنسبة لقيمة المؤشر الحيوي مع مرور للزمن، وصيغة رقمية بالنسبة لقيمة المؤشر الحيوي في اللحظة نفسها (قيمة آنية)، وتمّ استخدام الصيغة النصية لعرض بيانات معرف المريض، والوسمة الزمنية، وبروتوكول التراسل المعتمد، كما هو موضح في الشكل(4-28)، والذي يبين الكود وعملية الربط، أما الشكل(4-29) فيبين الخرج لحالتين مختلفتين لمصدر البيانات.



الشكل(4-28): الكود الرسومي في Node-Red



الشكل(4-29): الخرج النهائي للبيانات على Node-Red

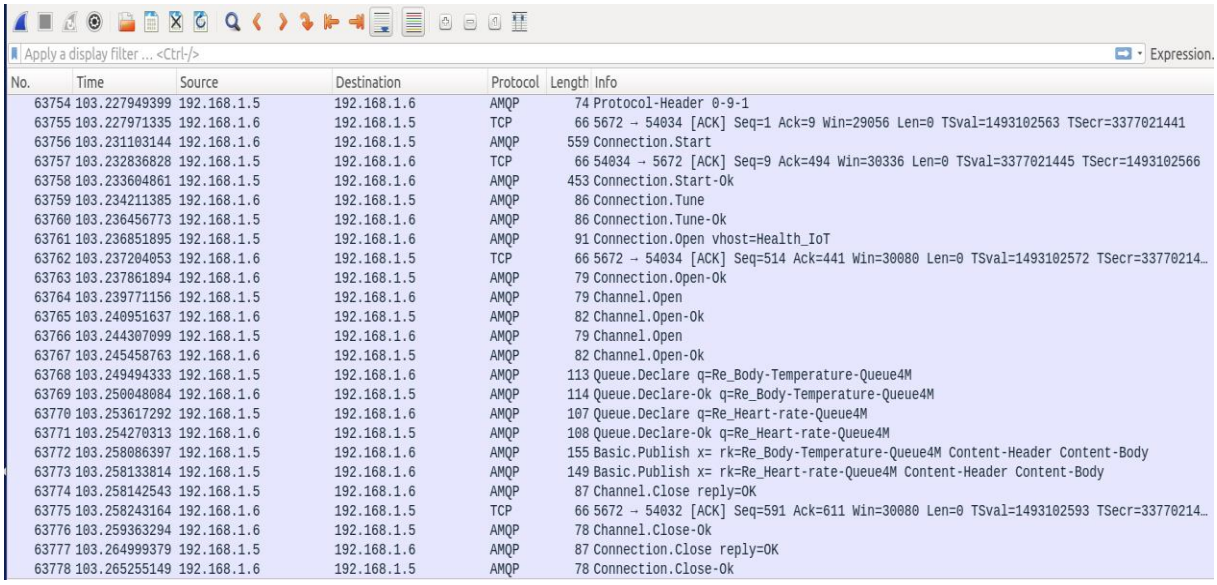
4-5-8 إضافة البروتوكول AMQP للبنية :

يعتبر AMQP البروتوكول الأساسي للتعامل مع وسيط الرسائل RabbitMQ، و الذي لا يحتاج أي إضافة برمجية، و إنما يتطلب URL صحيح و حسب، نحدد من خلاله اسم المستخدم و كلمة السر وعنوان العقدة الهدف و رقم البورت و المضيف الافتراضي، فالصيغة العامة تعرف بالشكل:

BROKER_URL: "amqp://user:password@remote.server.com:port//vhost"

مثال: String URL =“amqp://admin:testing@192.168.1.6:5672//Health_IoT”

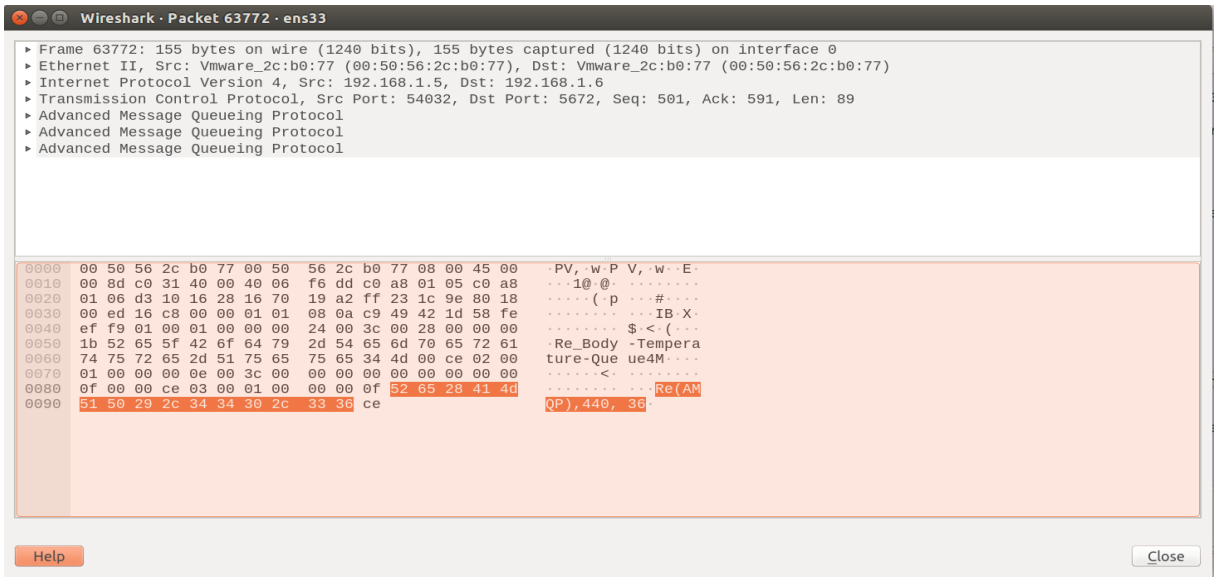
بعد تحقيق عملية الاتصال عن بعد عبر AMQP، قمنا بالتقاط الرزم باستخدام البرنامج Wire Shark، حيث نلاحظ أن المضيف ذا العنوان 192.168.1.5 في السطر الثالث أنشأ اتصالاً عبر AMQP ، فرد عليه المضيف ذو العنوان 192.168.1.6 في السطر الخامس بالإيجاب، ثم في السطر الثامن فتح الاتصال عبر المضيف الافتراضي Health_IoT ، ثم تلاه بعض العمليات للتأكيد على فتح الاتصال و القناة ، في السطر الخامس عشر و ما يليه، تم تعريف الرتلين، ثم أخيراً تمت عملية النشر، ثم تم إغلاق القناة و الاتصال، وهذا ما يوضحه الشكل(4-30).



No.	Time	Source	Destination	Protocol	Length	Info
63754	103.227949399	192.168.1.5	192.168.1.6	AMQP	74	Protocol-Header 0-9-1
63755	103.227971335	192.168.1.6	192.168.1.5	TCP	66	5672 → 54034 [ACK] Seq=1 Ack=9 Win=29056 Len=0 TSval=1493102563 TSecr=3377021441
63756	103.231103144	192.168.1.6	192.168.1.5	AMQP	559	Connection.Start
63757	103.232836828	192.168.1.5	192.168.1.6	TCP	66	54034 → 5672 [ACK] Seq=9 Ack=494 Win=30336 Len=0 TSval=3377021445 TSecr=1493102566
63758	103.233604861	192.168.1.5	192.168.1.6	AMQP	453	Connection.Start-Ok
63759	103.234211385	192.168.1.6	192.168.1.5	AMQP	86	Connection.Tune
63760	103.236456773	192.168.1.5	192.168.1.6	AMQP	86	Connection.Tune-Ok
63761	103.236851895	192.168.1.5	192.168.1.6	AMQP	91	Connection.Open vhost=Health_IoT
63762	103.237204053	192.168.1.6	192.168.1.5	TCP	66	5672 → 54034 [ACK] Seq=514 Ack=441 Win=30080 Len=0 TSval=1493102572 TSecr=33770214…
63763	103.237861894	192.168.1.6	192.168.1.5	AMQP	79	Connection.Open-Ok
63764	103.239771156	192.168.1.5	192.168.1.6	AMQP	79	Channel.Open
63765	103.240951637	192.168.1.6	192.168.1.5	AMQP	82	Channel.Open-Ok
63766	103.244307099	192.168.1.5	192.168.1.6	AMQP	79	Channel.Open
63767	103.245458763	192.168.1.6	192.168.1.5	AMQP	82	Channel.Open-Ok
63768	103.249494333	192.168.1.5	192.168.1.6	AMQP	113	Queue.Declare q=Re_Body-Temperature-Queue4M
63769	103.250048084	192.168.1.6	192.168.1.5	AMQP	114	Queue.Declare-Ok q=Re_Body-Temperature-Queue4M
63770	103.253617292	192.168.1.5	192.168.1.6	AMQP	107	Queue.Declare q=Re_Heart-rate-Queue4M
63771	103.254270313	192.168.1.6	192.168.1.5	AMQP	108	Queue.Declare-Ok q=Re_Heart-rate-Queue4M
63772	103.258086397	192.168.1.5	192.168.1.6	AMQP	155	Basic.Publish x= rk=Re_Body-Temperature-Queue4M Content-Header Content-Body
63773	103.258133814	192.168.1.5	192.168.1.6	AMQP	149	Basic.Publish x= rk=Re_Heart-rate-Queue4M Content-Header Content-Body
63774	103.258142543	192.168.1.6	192.168.1.5	AMQP	87	Channel.Close reply=OK
63775	103.258243164	192.168.1.6	192.168.1.5	TCP	66	5672 → 54032 [ACK] Seq=591 Ack=611 Win=30080 Len=0 TSval=1493102593 TSecr=33770214…
63776	103.259363294	192.168.1.6	192.168.1.5	AMQP	78	Channel.Close-Ok
63777	103.264999379	192.168.1.5	192.168.1.6	AMQP	87	Connection.Close reply=OK
63778	103.265255149	192.168.1.6	192.168.1.5	AMQP	78	Connection.Close-Ok

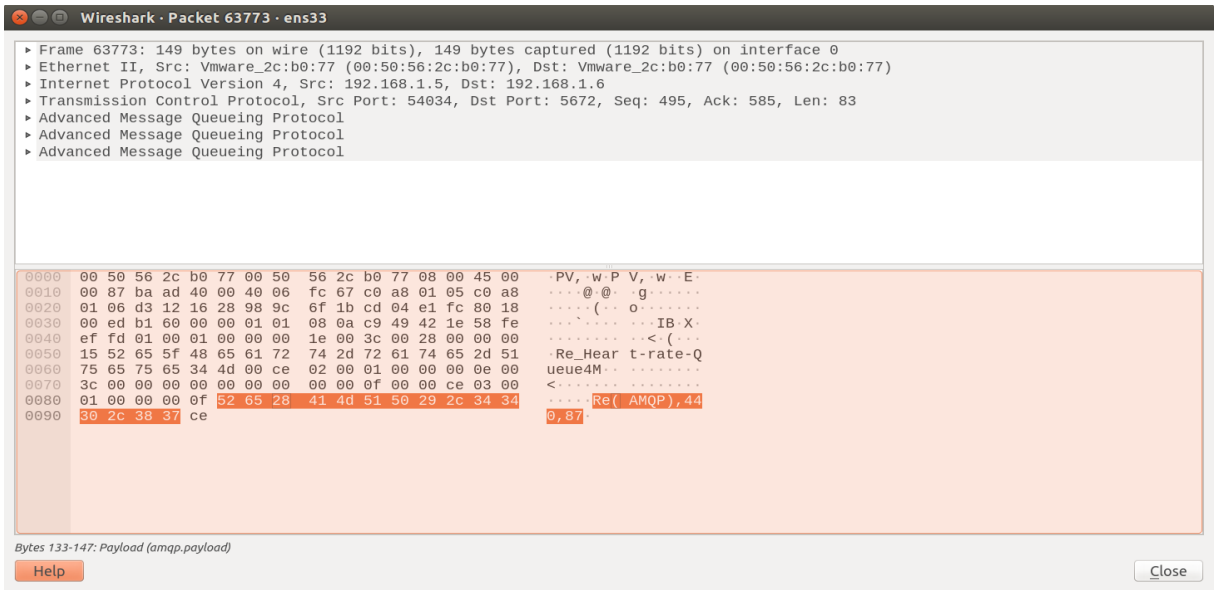
الشكل(4-30): التقاط رزم AMQP باستخدام البرنامج Wire Shark

وعند الدخول لتفاصيل عملية النشر عبر رتل درجة حرارة جسم الإنسان، وجدنا أن اسم الرتل والرسالة واضح ومقروء للإنسان ببساطة، ولاحظنا الثغرة الأمنية التي تعاني منها تقنية انترنت الأشياء بشكل عملي، بما فيها تحديد بروتوكول التراسل وعنوان المرسل وعنوان المستقبل، كل هذه التفاصيل كانت واضحة فمثلاً في الشكل(4-31) نجد اسم الرتل Re_Body-Temperature-Queue4M، والبيانات مرسلة عن بعد عبر AMQP، ومعرف المريض 440، وقيمة درجة الحرارة 36.



الشكل(4-31): محتوى رسالة درجة الحرارة التي نشرت عبر AMQP

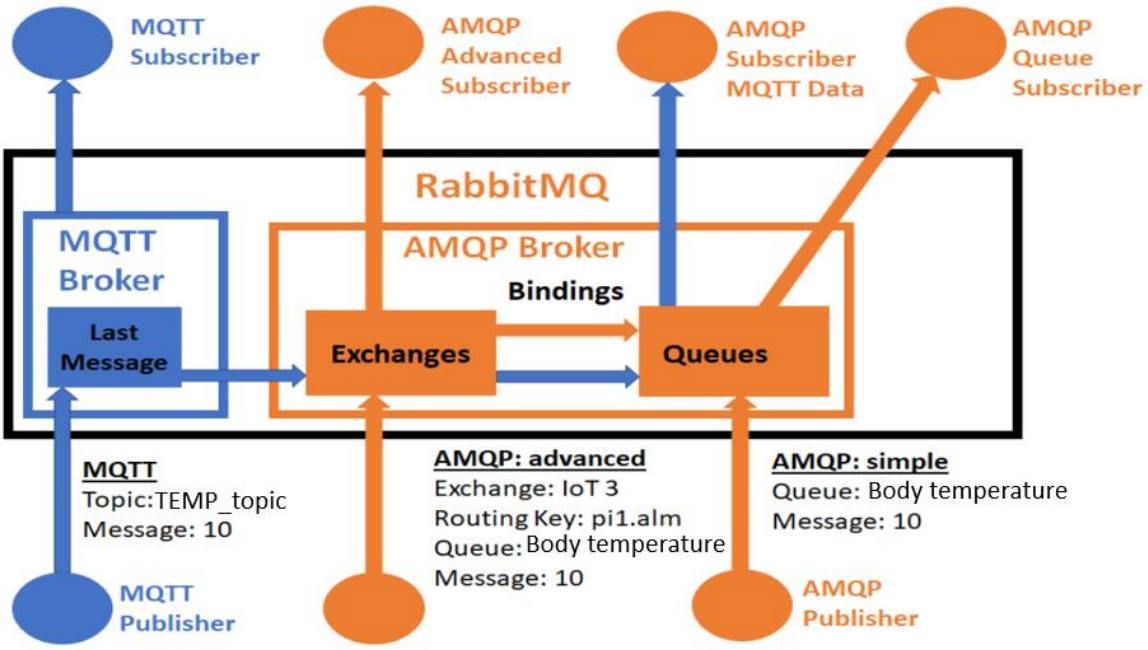
وبالمثل عند الدخول لتفاصيل عملية النشر عبر رتل معدل نبضات القلب، وجدنا أن اسم الرتل والرسالة ونوع بروتوكول التراسل وعنوان المرسل وعنوان المستقبل، كل هذه التفاصيل كانت واضحة فمثلاً في الشكل(4-32) نجد اسم الرتل Re_Heart-rate-Queue4M، والبيانات مرسلة عن بعد عبر AMQP، ومعرف المريض 440، وقيمة معدل نبضات القلب 87.



الشكل(4-32): محتوى رسالة معدل نبضات القلب التي نشرت عبر AMQP

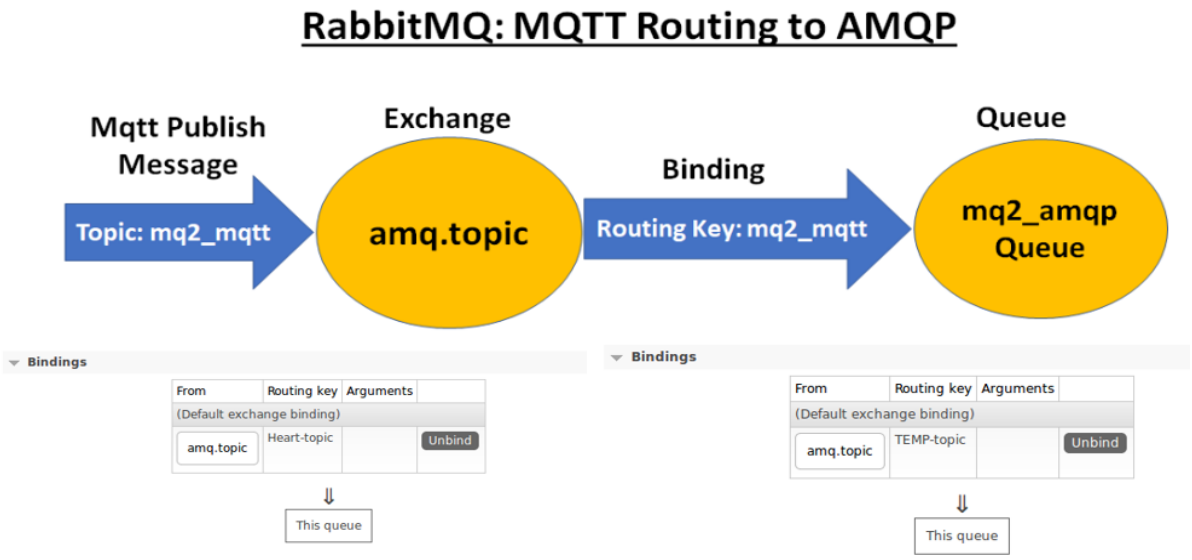
4-5-9 إضافة البروتوكول MQTT للبنية :

يعتمد البروتوكول MQTT على مبدأ الموضوع Topic أثناء تبادل الرسائل، ولذلك يحتاج لإضافة برمجية Plugin لتتم عملية الموائمة مع وسيط الرسائل RabbitMQ كونه يمثل تحقيقاً برمجياً للبروتوكول AMQP، والذي يعتمد على تقنية الأرتال، وهذا ما يبينه الشكل (4-33).



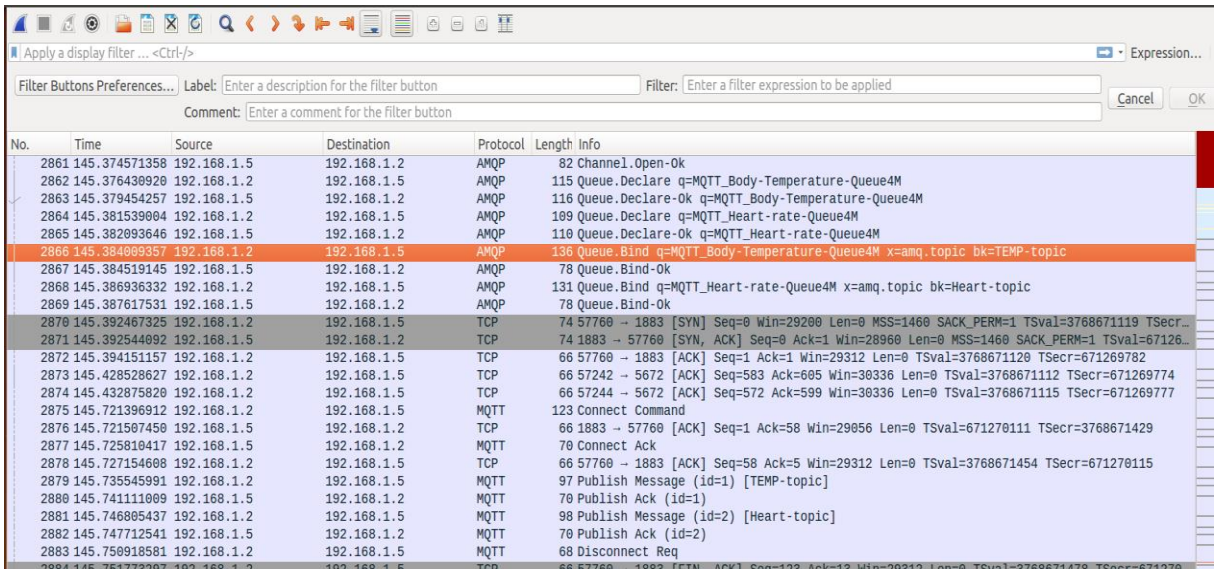
الشكل (4-33): المخطط الكتلي لإضافة MQTT إلى RabbitMQ

ففي الشكل السابق نلاحظ أن الرتل Body temperature يرتبط مع الموضوع TEMP_topic عبر مبدل يستخدم مفتاح التوجيه pi1.alm لتسليم الرسالة ذات المعرف رقم 10، أما الشكل (4-34) فيوضح المبدأ العام لعملية توجيه الرسائل بين البروتوكولين مع المثال الفعلي المنفذ عبر الموضوعين Heart-topic و TEMP-topic المربوطين مع المبدل amq.topic.



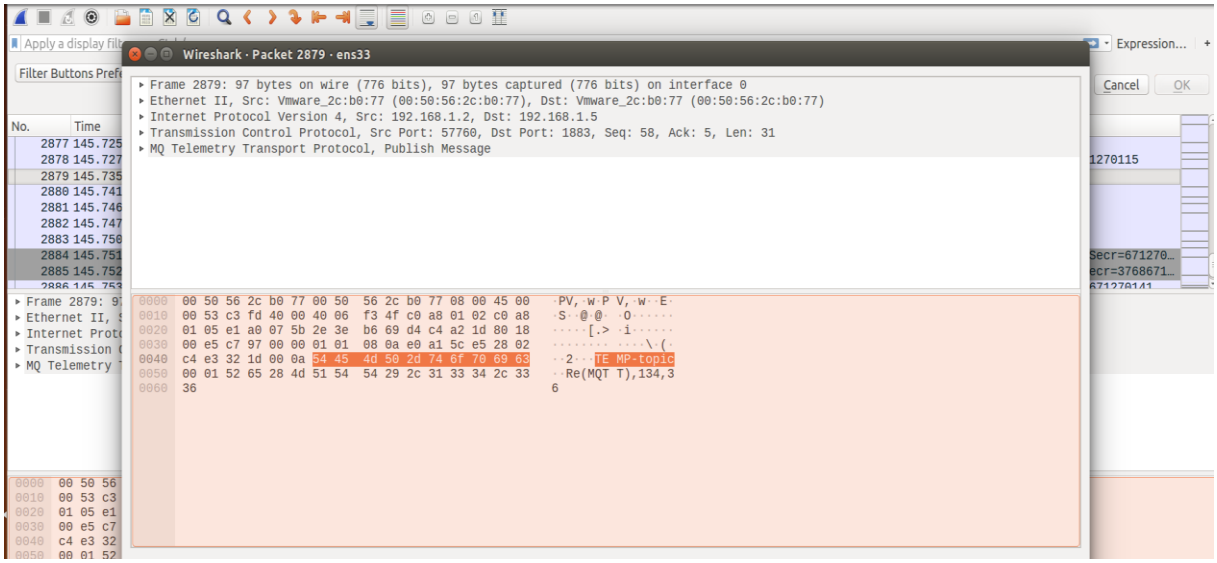
الشكل (4-34): المخطط الكتلي لتوجيه رسائل MQTT إلى AMQP

بعد تحقيق عملية الاتصال عن بعد عبر MQTT، قمنا بالتقاط الرزم باستخدام البرنامج Wire Shark، حيث نلاحظ أن المضيف ذا العنوان 192.168.1.5 في السطر السادس بدأ بتعريف عملية الربط بين اسم الرتل MQTT_Body-Temperature-Queue4M و المبدل amq.topic عبر مفتاح التوجيه TEMP-topic ، و عند إجراء نشر الرسالة في السطر التاسع عشر ميزها بثنائية تتمثل باسم الموضوع و معرّف الرسالة، و هذا موضح في الشكل (4-35).



الشكل (4-35): التقاط رزم MQTT باستخدام Wire Shark

وعند الدخول لتفاصيل عملية النشر عبر موضوع درجة حرارة جسم الإنسان، وجدنا أن اسم الموضوع والرسالة واضح ومقروء للإنسان ببساطة، ولاحظنا الثغرة الأمنية التي تعاني منها تقنية انترنت الأشياء عبر MQTT بشكل عملي، بما فيها القدرة على تحديد بروتوكول التراسل وعنوان المرسل وعنوان المستقبل، كل هذه التفاصيل كانت واضحة فمثلاً في الشكل(4-36) نجد اسم الموضوع TEMP-topic، والبيانات مرسلة عن بعد عبر MQTT، ومعرف المريض 134، وقيمة درجة الحرارة 36.



الشكل (4-36): محتوى رسالة درجة الحرارة التي نشرت عبر MQTT

الفصل الخامس

تقييم أداء بروتوكولات التراسل في البنية المقترحة

5-1 الدراسة التجريبية وتقييم الأداء:

تتألف منهجية تقييم الأداء من الخطوات: اختيار معدل أخذ العينات الذي يناسب طبيعة المؤشر الحيوي، ثم دراسة معدلات استقبال الرسائل تبعاً لتغيير بروتوكول التراسل المستخدم.

5-2 اختيار معامل تقييم الأداء:

إن الهدف من اختيار معامل معدل القدح الزمني كمعامل لعملية التقييم يرتبط ارتباطاً وثيقاً بطبيعة النظام المخصص لمراقبة المؤشرات الحيوية، و الذي يفرض التمييز بينها من حيث معدل أحذ العينات، و الذي يتعلق بطبيعتها (حرجة- غير حرجة)على ثلاثة مستويات:

1- مرحلة جمع البيانات: و لها ارتباط وثيق بالعتاد الصلب (الكلفة الأقل).

2- مرحلة ارسال البيانات: اختيار بروتوكول التراسل الأمثل .

3- مرحلة المعالجة في Apache Storm : تتمثل باستخدام تابع التجميع في الطوبولوجيا بالنسبة للمؤشرات غير الحرجة .

5-3 مواد وطرق البحث:

تمّ استخدام جهاز محمول بمعالج انتل Core i7، يعمل بتردد GH2.7، و بهرمية 64، و ذواكر 16 غيغا بايت، و قد حُققت البنية على نظام تشغيل: Ubuntu 16.04 LTS /64 bit ، و الذي يعمل كآلة افتراضية على Windows 10 Pro /64 bit، أما لغة البرمجة التي استخدمت لكتابة الشيفرات في البرمجيات المفتوحة المصدر المستخدمة لبناء هذا النظام، فهي لغة جافا java 8 .

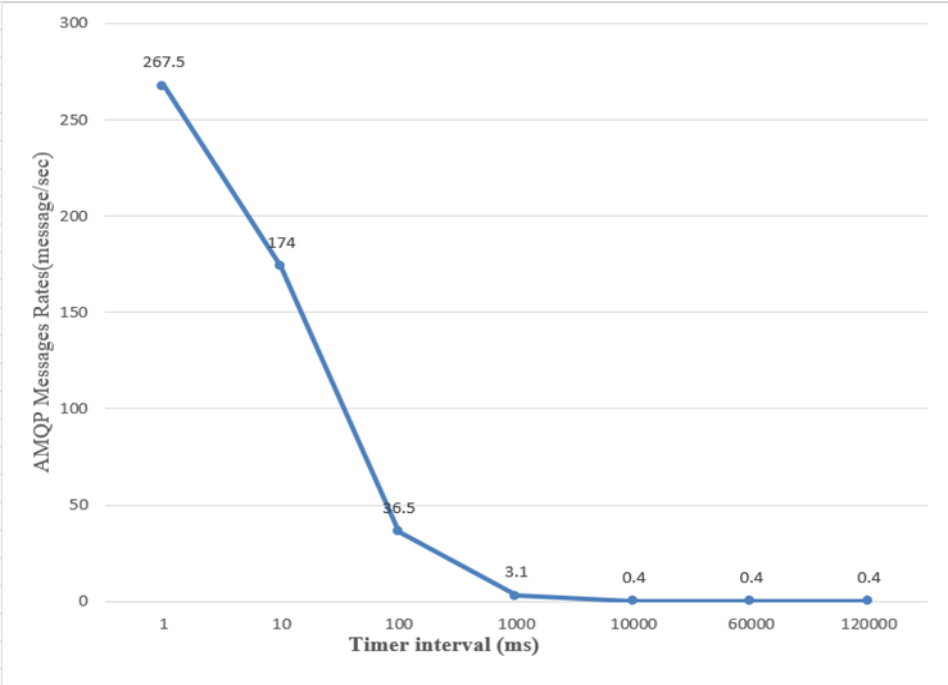
5-4 نتائج التقييم:

بداية لابد من الإشارة إلى أننا اتبعنا الطريقة العلمية من خلال التجريب والملاحظة وذلك تبعاً للقيم التي تقيسها واجهة المستخدم لوسيط الرسائل RabbitMQ وبفترات زمنية متباعدة، والجدول(5-1) يوضح القيم التجريبية لمعدل استقبال الرسائل عبر البروتوكولين مع تغيير معدل الارسال في كل مرة.

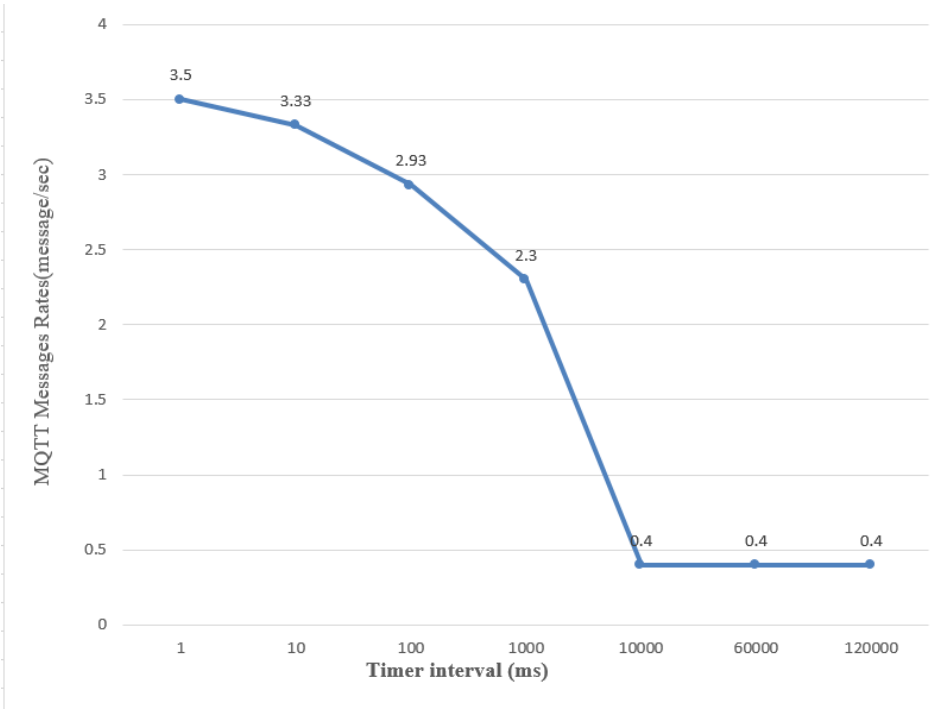
Timer interval (ms)	AMQP Messages Rates(message/sec)	MQTT Messages Rates(message/sec)
1	267.5	3.5
10	174	3.33
100	36.5	2.93
1000	3.1	2.3
10000	0.4	0.4
60000	0.4	0.4
120000	0.4	0.4

الجدول(5-1): مقارنة معدل استقبال الرسائل عبر بروتوكولي التراسل

ونلاحظ أن الشكل (5-2) يوضح تأثر معدل استقبال الرسائل عبر AMQP مع تغيير معدل الارسال، أما الشكل (5-3) فيوضح تأثر معدل استقبال الرسائل عبر MQTT مع تغيير معدل الارسال.



الشكل (5-2): تغير معدل الاستقبال عبر AMQP مع تغير معدل الارسال



الشكل (5-3): تغير معدل الاستقبال عبر MQTT مع تغير معدل الارسال

5-5 مناقشة النتائج:

تظهر النتائج أنه من أجل قيم عالية لإرسال البيانات يتفوق AMQP من حيث الأداء و بفرق شاسع على MQTT، و كلما انخفض معدل إرسال البيانات ينخفض أداء AMQP مقارنة مع MQTT، و لكن عند عتبة معينة يتساوى البروتوكولين من حيث الأداء (10 ثانية)، و كل قيمة أعلى من العتبة يستمر الفرق معدوماً بينهما، و في الواقع إن هذه النتيجة منطقية؛ لأن البنية اعتمدت على وسيط الرسائل RabbitMQ، و الذي يمثل تحقيقاً برمجياً مفتوح المصدر للبروتوكول AMQP بلغة البرمجة Erlang، و بالتالي من المنطقي أن يتفوق AMQP على بقية البروتوكولات المتاحة عبر الإضافات Plugins.

الفصل السادس

تحسين البنية المقترحة وتداعياتها

6-1 تحقيق النسخة الآمنة:

بالعودة للشكل(4-31) نجد أن البرتوكول AMQP يرسل بيانات الرسالة بشكل واضح، وبالتالي البيانات ستكون عرضة للاطلاع عليها من قبل المتنصتين، وأيضاً عرضة للتعديل عليها من قبل أشخاص غير مصرح لهم، لذلك قمنا بتطوير المنتج ليتم تطبيق خوارزمية AES لتشفير الرسالة، ثم يتم حساب ملخص الرسالة المشفرة بتطبيق خوارزمية MD5، ويرسل الملخص مرفقاً بالرسالة المشفرة بصيغة CSV، وهذا ما يوضحه الشكل(6-1) و الذي يمثل خرج المنتج خلال دورة حياة ارسال رسالة واحدة.

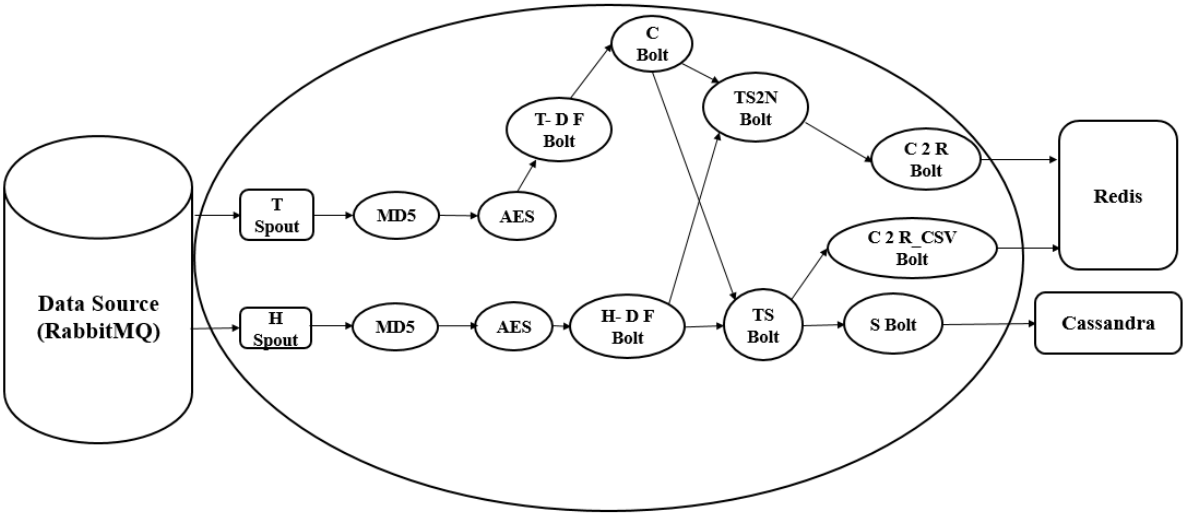
```
 creating connection factory.....
 creating new connection.....
5672
localhost/127.0.0.1
opening channel
Temperature Generated Now : 39
HeartBeat rate Generated Now : 88
Patient for this Vital  ID is : 164
Temperature Before Encryption: MQTT,164,39
Heart Rate Before Encryption: MQTT,164,88
Sending  Body Temperature Data Now :'3grgsV9HXWxkA1qZJ/Zo2Q==,BECE6A43B216500148A9D7F25211D982'
Sending Heart rate Data Now : '6LI6d8o6Q1b0KikmlmrW1g==,025A156AD7FCF1A341F505D9027B79EF'
Decrypted Text After Decryption: MQTT,164,39
Decrypted Text After Decryption: MQTT,164,88
```

الشكل(6-1): خرج المنتج ذو النسخة الآمنة

وبالمقابل فإن هذا التعديل يتطلب تعديل الطوبولوجيا في Storm، فهي تمثل دماغ النظام، والشكل(6-2) يوضح إضافة معالجين أساسيين لكل مسار وهما:

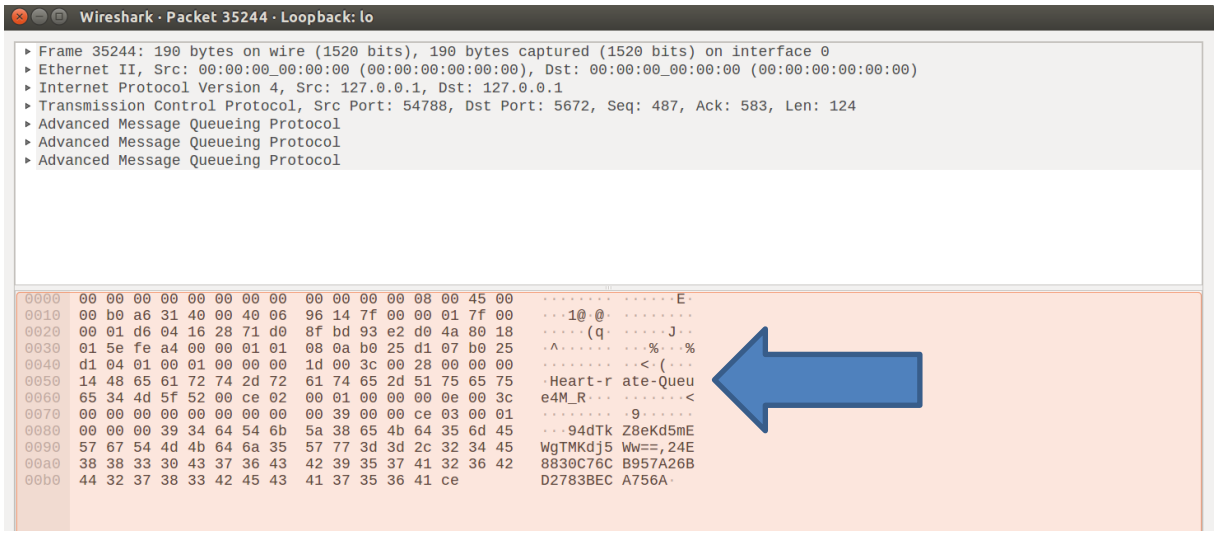
1- المعالج MD5 Bolt: يقوم بحساب ملخص الرسالة ليتأكد من سلامتها وعدم التعديل عليها.

2- المعالج AES Bolt: يقوم بفك تشفير الرسالة باستخدام AES، ثم تمريرها للمعالج التالي في المسار.



الشكل(6–2): التمثيل التقني للطوبولوجيا في Apache Storm بنسخته الآمنة

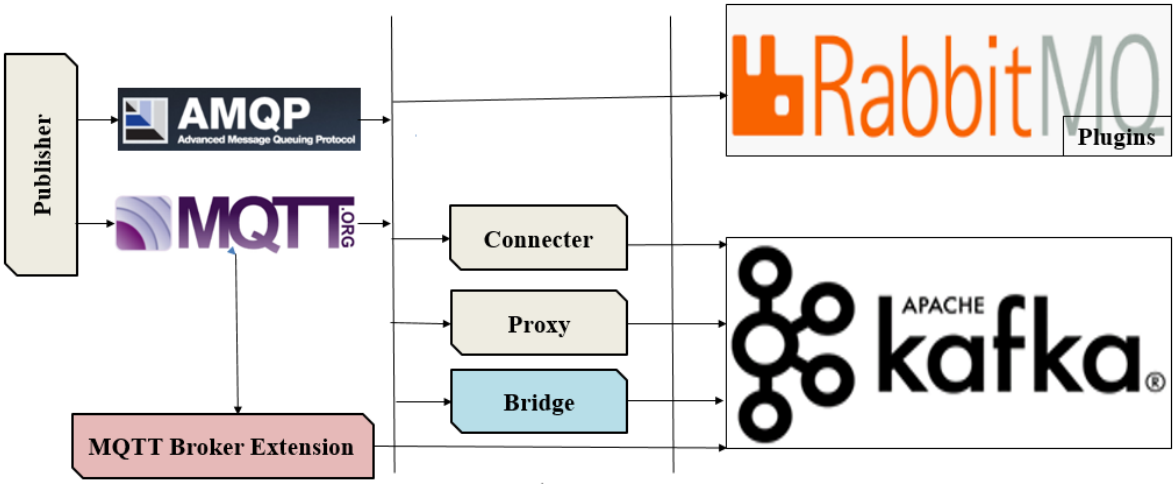
بعد تحقيق الكود الموافق للطوبولوجيا بنسختها الآمنة، قمنا بالتقاط الرزم باستخدام البرنامج Wire Shark، حيث نلاحظ أن معامل توجيه الرسالة والمتمثل باسم الرتل يبقى مقروءاً بشكل واضح ويقتصر التشفير على الرسالة، وهذا موضح في الشكل (6–3)، هذه مشكلة بحثية حقيقية تعاني منها تقنية انترنت الأشياء، حيث نجد أنه من غير المقبول تشفير معامل توجيه الرسالة، بمعنى أننا لا نستطيع تشفير اسم الرتل أثناء توجيه الرسائل في البروتوكول AMQP أو اسم الموضوع أثناء توجيه الرسائل في البروتوكول MQTT.



الشكل(6–3): التقاط رزم AMQP المشفرة باستخدام Wire Shark

6-2 إضافة وسيط الرسائل Apache Kafka:

إن وسيط الرسائل Apache Kafka يتفوق من ناحية الأداء و الإنتاجية بخمس مرات على وسيط الرسائل RabbitMQ [41]، لذلك كان لابد من طرح مسألة تحسين الإنتاجية للبنية، و بالتالي العمل على إضافة Apache Kafka كونه المنافس الأقوى و الحل البديل لوسيط الرسائل RabbitMQ ، و لكن السؤال الهام الذي يطرح نفسه هل Apache Kafka يمثل الحل الأنسب لحالة الاستخدام هذه كونه يمنحنا إنتاجية عالية؟ حقيقة إن مسألة الدعم تتدخل في هذه الحالة لترجح كفة RabbitMQ، حيث نلاحظ في الشكل(6-4) أن وسيط الرسائل RabbitMQ يقدم الدعم لبروتوكولات التراسل عبر إضافات Plugins تتطلب فقط تفعيلاً، وبعض الإعدادات، أو بشكل مباشر، ولكن هذا لا يحسم المسألة بشكل مباشر، حيث تم اقتراح أربعة مناهج لحل مشكلة ربط Apache Kafka مع بروتوكولات التراسل، وسندرس في هذا الفصل إيجابيات وسلبيات هذه المناهج وأثرها في الناحية البحثية والتطبيقية لعملنا.



الشكل(6-4): المخطط الكتلي لربط وسيطي الرسائل مع بروتوكولي التراسل

6-2-1 نهج دمج Apache kafka مع MQTT [42]:

هناك أربعة أساليب هندسة مختلفة لتطبيق هذا النوع من الجسور:

1- **Kafka Connect for MQTT:** و هي إضافة تسمح ل Kafka باستيعاب البيانات من MQTT و بالتالي تؤدي دور وسيط MQTT، و لكن هذا النهج يعاني من قيود على الأداء و قابلية التوسع و غير ملائم لمعالجة كميات كبيرة من رسائل MQTT.

2- **MQTT Proxy** : هو تطبيق خفيف يقوم بدفع رسائل MQTT إلى Kafka، و لكن المشكلة الحقيقية تكمن في أنه ليس وسيطMQTT كامل المواصفات، لأنه لا يستند لنموذج إلى pub / sub، و لكن إذا لم يكن فقد الرسالة عاملاً مهماً، وإذا لم يتم استخدام ميزات MQTT المصممة للاتصال الموثوق به في إنترنت الأشياء، فقد يكون نهج البروكسي بديلاً خفيفاً.

3- Build Your Own Custom Bridge:

يتم في هذا النهج إنشاء تطبيق باستخدام مكتبة عميل MQTT مفتوحة المصدر، ومكتبة عميل Kafka مفتوحة المصدر، التحدي الرئيسي في هذا النهج هو أن التطبيق المخصص ليس مصمماً عادةً ليكون متسامحاً مع الخطأ ومرناً، حيث يصبح هذا الأمر مهماً إذا تطلب حل إنترنت الأشياء ضماناً شاملًا لمرة واحدة على الأقل أو مرة واحدة تماماً لتسليم الرسالة، وأيضاً سيتطلب التطبيق المخصص جهود تطوير كبيرة ليعمل بشكل صحيح.

4- MQTT Broker Extension :

هي إضافة تسمح لوسيط الرسائل بإضافة وتضمين بروتوكول Kafka الأصلي، ويسمح هذا النهج لحل إنترنت الأشياء باستخدام تطبيق MQTT أصلي وتطبيق Kafka أصلي، بمعنى أن وسيط الرسائل يعمل بكامل المواصفات، ولكن هذا الحل ليس مجانياً وليس مفتوح المصدر.

6-2-2 مناقشة جدوى دمج Apache kafka مع MQTT:

هنا لابد من الأخذ بعين الاعتبار الموازنة بين المحورين اللذين تعتمد عليهما البنية وهما محور انترنت الأشياء و محور معالجة البيانات الضخمة، و نقطة أخرى لابد من مراعاتها و هي الموازنة بين الجانب البحثي للعمل من جهة و طبيعة الجانب التطبيقي و التي تفرض بعض القيود مثل الوثوقية في تسليم الرسائل من جهة أخرى.

1- مناقشة النهج الأول: هو نهج مرفوض بشكل قطعي لأن مشكلة البحث ترتبط بشكل وثيق بمعالجة البيانات الضخمة التي تولدها تقنية انترنت الأشياء، فضلاً عن القيود التي يعاني منها على الأداء و قابلية التوسع.

2- مناقشة النهج الثاني: قد يكون مقبولاً في الحالة التي يعمل فيها النظام مع مؤشرات حيوية غير حرجة، وبالتالي لا يكون فقدان بعض الرسائل مشكلة مؤثرة في الحالة الصحية للمريض، أما بالنسبة للمؤشرات الحيوية الحرجة فلا يعتبر حلاً مقبولاً.

3- مناقشة النهج الثالث: تم بناء هذا النظام على مبدأ تقديم ضمان من نوع مرة واحدة تماماً لتسليم الرسالة، لذلك هذا النهج غير مناسب كونه يفتقر لميزة التسامح مع الأعطال,

4- مناقشة المنهج الرابع: قد يكون حلاً جيداً، ولكن هذا الحل ليس مجانياً وليس مفتوح المصدر.

6-2-3 نهج دمج Apache kafka مع AMQP[43]:

عند الربط مع AMQP فليس لدينا سوى منهج وحيد وهو أن يتم إنشاء تطبيق باستخدام مكتبة عميل AMQP مفتوحة المصدر، ومكتبة عميل Kafka مفتوحة المصدر، التحدي الرئيسي في هذا النهج هو أن هذا الجسر البرمجي لا يعمل بشكل مباشر على Ubuntu و إنما يعمل كآلة افتراضية فوقه، و التحدي الآخر هو أن التطبيق المخصص ليس مصمماً عادةً ليكون متسامحاً مع الخطأ ومرناً، حيث يصبح هذا الأمر مهماً إذا تطلب حل إنترنت الأشياء ضماناً شاملًا لمرة واحدة على الأقل أو مرة واحدة تماماً لتسليم الرسالة، وأيضاً سيتطلب التطبيق المخصص جهود تطوير كبيرة ليعمل بشكل صحيح.

مما سبق نستنتج أن RabbitMQ هو الحل الأنسب من الناحية البحثية والناحية التطبيقية، فهو فضلاً عن كونه يتفوق من ناحية الدعم، فإنه يتمتع بكونه يقدم حلاً مستقراً وقابلاً للتوسيع ومتسامحاً مع الأعطال.

الفصل السابع

التوصيات وآفاق المستقبل

7-1 النتائج والتوصيات:

تمّ في هذا البحث تصميم نظام معتمد على انترنت الأشياء وتحقيقه، وتقييمه لمراقبة بعض المؤشرات الحيوية للمرضى عن بعد، حيث تمّ اعتماد مبدأ الأمثلة في اختيار المكونات خلال مرحلة تحليل المتطلبات وفقاً لشروط البيئة، وتمّ إجراء مقارنة بين البروتوكولين AMQP و MQTT وفقاً لهذه البنية و الشروط التي تفرضها، و بينت نتائج هذه المقارنة أنه من أجل المؤشرات الحيوية الحرجة (مثل معدل نبضات القلب)، والتي تتطلب قيماً عالية لإرسال البيانات يجب اختيار حساس أو جهاز قياس يدعم AMQP، و الذي يتفوق في الأداء على MQTT بفرق شاسع .

كما بينت النتائج أنه من أجل المؤشرات الحيوية غير الحرجة (مثل درجة حرارة الجسم)، والتي لا تتطلب معدلاً عالياً لإرسال البيانات (10 ثانية وأكثر)، فلا فرق في الأداء بين البروتوكولين، لذلك نختار البروتوكول الذي يدعم كلفة الحساس الأرخص.

وأيضاً بينت دراسة مناهج ربط بروتوكولات التراسل مع Apache Kafka أن وسيط الرسائل RabbitMQ هو الأفضل لمثل هذه البنى والتي تجمع بين محوري انترنت الأشياء والبيانات الضخمة.

7-2 آفاق المستقبل:

1- يمكن إضافة بروتوكولات أخرى للمقارنة مثل Websocket و XMPP و HTTPS أو H2ot، و الذي ما زال قيد التحقيق لليوم .

2- يمكن بناء جسر لربط Apache kafka يدعم بروتوكولات التراسل الأفضل، ويراعي الميزات الأساسية للنظم مثل قابلية التوسيع والتكيف مع الأعطال وغيرها.

3- تطوير البنية من نظام يعالج تدفق البيانات الضخمة إلى نظام تنبؤي للحالة الصحية للمريض، أي بناء نظام يمثل جيلاً مطوراً من نظم دعم القرارات السريرية Clinical decision support system، يقوم بتحليل بيانات المرضى

المخزنة في قاعدة البيانات، والتنبؤ بالحالة الصحية للمريض، وبالتالي القدرة على إعطاء نظرة مستقبلية عن حالة المريض، أي لن نكتفي بتشخيص حالته في الوقت الحاضر.

4- تطوير AMQP أمنياً ليتم تشفير اسم الرتل، وتطوير MQTT ليتم تشفير اسم الموضوع أثناء إرسال البيانات، مع مراعاة سلامة عملية التوجيه.

قائمة المراجع

المراجع:

[1] "eHealth," [Online]. Available: https://en.wikipedia.org/wiki/EHealth. [Accessed 4 4 2019].

[2] "Internet of things," [Online]. Available: https://en.wikipedia.org/wiki/Internet_of_things. [Accessed 1 6 2019].

[3] StreamSets, “IoT Reference Architecture for Hadoop” . (2016, October).

[4] Lopez Reasarch. (2013, November). " An introduction to the internet of things (IoT) “. San Francisco.

[5] M. Minelli, M. Chambers, A. Dhiraj, (2012). “Big Data, Big Analytics”.

[6] M. Grimaldi, M. Greco, A. De Mauro, "A formal definition of Big Data based on its essential features," 2016.

[7] (n.d.).” What is HPC … and why you care”.

[8] A. Priyadharshini. (2017, April).” An Intelligent Patient Monitoring Through Iot By Mqtt” . SSRG International Journal of Computer Science and Engineering , pp. 19-23.

[9] G. Alfian, M. Syafrudin, M. Ijaz, A. Syaekhoni, N. Fitriyani, J. Rhee, "A Personalized Healthcare Monitoring System for Diabetic Patients by Utilizing BLE-Based Sensors and Real-Time Data Processing," 6 jul 2018.

[10] S. Yadav, E. Reddy B, K. Srinivasa,"Cloud-Based Healthcare Monitoring System Using Storm and Kafka," jun 2018.

[11] N. El aboudi, L. Benhlima, "Big Data Management for Healthcare Systems: Architecture, Requirements, and Implementation," 21 jun 2018.

[12] A. Abdelgawad, K. Yelamarthi, A. Khattab, "IoT-Based Health Monitoring system for Active and Assisted Living," 16 october 2017.

[13] StreamSets, “IoT Reference Architecture for Hadoop” . (2016, October).

[14] S. Chatterjee, "The Brain of an IoT System: Analytics Engines and Databases," 1 11 2015. [Online]. Available: https://blog.persistent.com/index.php/2015/10/01/the-brain-of-an-iot-system-analytics-engines-and-databases/. [Accessed 30 5 2019].

[15] P. Marizq, "Artificial Intelligence and Telemedicine in the Field of Well-Being - Reading the Algerian Reality," 11th Annual Scientific Conference, pp. 628-644, 23 April 2012.

[16] M. Marks, D. Stojanovic, S. Pllana, J. Molina, M. Krzyszton, A. Sikora, A. Jarynowski, F. Hosseinpour, A. Jakobik, A. Stojnev Ilic, A. Respicio, D. Moldovan, C. Pop, L. Salomie, "Medical Data Processing and Analysis for Remote Health and Activities Monitoring," 26 3 2019.

[17] L. Wang, C. Alexander, "Healthcare Driven by Big Data Analytics," 2018.

[18] N. Naik, "Choice of Effective Messaging Protocols for IoT Systems: MQTT, CoAP, AMQP and HTTP," 2017 IEEE.

[19] Perros, P. K. (2017). A Comparison of IoT application layer protocols through a smart parking implementation.

[20] Yuang Chen, T. K. (2016). Performance Evaluation of IoT Protocols under a Constrained Wireless Access Network. IEEE.

[21] Robert Steele, A. C. (2013, June). The Internet of Things and Next-generation Public Health Information Systems.

[22] (n.d.). Retrieved 12 4, 2017, from MQTT.ORG : http://mqtt.org/

[23] 5 Things to Know About MQTT – The Protocol for Internet of Things . (n.d.). Retrieved 12 4, 2017, from https://www.ibm.com/developerworks/community/blogs/5things/entry/5_things_to_know_about_mqtt_the_protocol_for_internet_of_things?lang=en

[24] T. Kellogg, (2015, February 20). Retrieved 12 4, 2017, from Can HTTP/2 Replace MQTT?

[25] T. Salman, (2015, 11 30). Networking Protocols and Standards for Internet of Things . Retrieved 12 4, 2017, from https://www.cse.wustl.edu/~jain/cse570-15/ftp/iot_prot.pdf

[26] (n.d.). Retrieved 12 4, 2017, from MQTT 101 – How to Get Started with the lightweight IoT Protocol: https://www.hivemq.com/blog/how-to-get-started-with-mqtt

[27] "RabbitMQ for IoT". (2019, 4 4). Retrieved from Fun Tech Projects: https://funprojects.blog/2018/12/07/rabbitmq-for-iot/

[28] "Advanced Message Queuing Protocol," Wikipedia, the free encyclopedia, 3 2018. [Online]. Available: https://en.wikipedia.org/wiki/Advanced_Message_Queuing_Protocol. [Accessed 3 6 2019].

[29] "RabbitMQ by Pivotal," [Online]. Available: https://www.rabbitmq.com/tutorials/tutorial-one-python.html. [Accessed 8 6 2019].

[30] Apache Storm tutorils point. (2018, 3 1).

[31] Why use Storm? (2018, 4 1). Retrieved from Apache Storm: http://storm.apache.org/

[32] A. Shukla, (2017, feb 16). Tutorial: Apache Storm.

[33] S. Saxena, (2015, March). "Real-time Analytics with Storm and Cassandra". BIRMINGHAM - MUMBAI.

[34] "Apache Cassandra". (2018, 4 2). Retrieved from Wikipedia The Free Encyclopedia: https://en.wikipedia.org/wiki/Apache_Cassandra

[35] "NoSQL Performance Benchmarks". (2019, 8 1). Retrieved from DATASTAX: https://www.datastax.com/nosql-databases/benchmarks-cassandra-vs-mongodb-vs-hbase

[36] "Redis," [Online]. Available: https://redis.io/. [Accessed 30 5 2019].

[37] P. Gogia, (2018, 2 14). Redis: What and Why? Retrieved from codeburst.io: https://codeburst.io/redis-what-and-why-d52b6829813

[38] "Node-Red," [Online]. Available: https://nodered.org/. [Accessed 30 5 2019].

[39] S. Soppin, (2017, 9 12). "Everything you need to know about Node-RED". Retrieved from Open Source For U: https://opensourceforu.com/2017/09/node-red/

[40] "Which one is best CSV or JSON in order to import big data (PHP)" . (2019, 7 7). Retrieved from stackoverflow: https://stackoverflow.com/questions/26156646/which-one-is-best-csv-or-json-in-order-to-import-big-data-php

[41] "Understanding When to Use RabbitMQ or Apache Kafka". (2018, 4 15). Retrieved from DZone/Integration Zone: https://dzone.com/articles/understanding-when-to-use-rabbitmq-or-apache-kafka

قائمة المراجع

[42] I. Skerrett, (2019, 4 24). “Streaming IoT Data and MQTT Messages to Apache Kafka”. Retrieved from DZone IoT Zone : https://dzone.com/articles/streaming-iot-data-and-mqtt-messages-to-apache-kaf

[43] “strimzi/strimzi-kafka-bridge” . (2019, 9 1). Retrieved from GitHub : https://github.com/strimzi/strimzi-kafka-bridge

[44] Visualizing Real Time Stream Data using Node-RED. (2019, 3 2). Retrieved from Treselle System: http://www.treselle.com/blog/visualizing-real-time-stream-data-using-node-red/

Syrian Arab Republic

Al-Baath University

Faculty of Informatics Engineering

Systems and Computer Networks Department

Designing, Implementing and Evaluating an IoT Based System to Remotely Monitor Some Vital Signs of Patients

A thesis submitted in partial fulfillment of the requirements for the Master's Degree in Systems and Computer Networks Engineering

Prepared By:

Eng. Osama Ali Ebrahem

Supervised By:

Dr. Zainab Khallouf

Assistant Professor in the Department of Systems and Computer Networks Engineering

Dr. Suhel Alhamoud

Instructor in the Department of Software and information Systems Engineering

1440 e -2019 m