مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (5) العدد(4) 2021

Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (5) No. (4) 2021

تعديل خوارزمية Zeus لتخصيص الموارد لتناسب نهجاً تفاعلياً في سحابة من المستشعرات

م. حسين يونس*

(تاريخ الإيداع 17 /3 /2021 . قُبِل للنشر في 18 /5 /2021)

□ ملخّص □

تجمع سحابة المستشعرات CoS بين مفاهيم الحوسبة السحابية، وحوسبة الحواف، وشبكات الحساسات والمشغلات اللاسلكية WSAN في معمارية ثلاثية الطبقات. بتمكين المحاكاة الافتراضية لهذه البنية، سيُنشئ النظام مجموعة من العقد الافتراضية VNs تمثل كيانات برمجية تعبر عن موارده من وحدات المعالجة المركزية، وذاكرة الوصول العشوائي RAM والتخزين، وبذلك يفصل التطبيقات الواردة عن تعقيد البنية التحتية للسحابة. لتقديم هذه الموارد لخدمة التطبيقات لا بد من عمليتين: الأولى "تخصيص الموارد" وفيها تحدد كمية الموارد التي سيقدمها النظام لكل تطبيق، والثانية "توفير الموارد" وفيها تهيأ البنية التحتية وتُنشأ الـ VNs المخصصة بالعملية السابقة. يركز هذا البحث على عملية التوفير التي تشمل نهجين: استباقيٌّ وتفاعلي. تُنشأ العقد الممثلة للموارد في النهج الاستباقي بشكلٍ سابق، وتُسنَد للتطبيقات عند ورودها، أما النهج التفاعلي فينشِئُها وفقاً للطلب. يعتبر النهج التفاعلي الحالة المثالية للعمل من ناحية ترشيد استخدام الموارد، ولكن التأخير الزمني الناتج عن الوقت اللازم لتوفير الـ VNs يجعل الاستباقي أكثر ملائمةً للتطبيقات الحساسة للزمن. في هذا العمل نستهدفُ تعديلَ خوارزمية Zeus الاستباقية لتلائم نهجاً تفاعلياً، مع تعديل نموذج المحاكاة الافتراضية فيها لتحقيق أقصى زمن حقيقيٍّ ممكن، وأكفأ استخدامٍ للموارد.

كلمات مفتاحية: سحابة المستشعرات، حوسبة الحواف، تخصيص الموارد، توفير الموارد، المحاكاة الافتراضية.

* باحث مهندس حائز على درجة الماجستير من المعهد العالي للعلوم التطبيقية والتكنولوجيا – دمشق – سورية.

مجلة جامعة طرطوس للبحوث والدراسات العلمية _ سلسلة العلوم الهندسية المجلد (5) العدد(4) 2021

Tartous University Journal for Research and Scientific Studies - engineering Sciences Series Vol. (5) No. (4) 2021

A Modification of "Zeus" a Resource Allocation Algorithm to Suit a Reactive Approach in a Cloud of Sensors

Eng. Hussein Youness*

(Received 17 / 3 / 2021 . Accepted 18 / 5 / 2021)

□ ABSTRACT □

The cloud of sensors (CoS) paradigm brings together cloud computing, edge computing and wireless sensor and actuator networks (WSAN) in the form of a three-tier architecture. By employing CoS virtualization, the system will create a set of virtual nodes (VNs) representing software entities that abstract its resources of CPUs, RAM and storage, thereby decoupling the incoming applications from the complex cloud infrastructure. To provide these resources to serve the applications, there are two required processes: the first is "Resource allocation", in which the system determines the amount of resources it will provide for each application, and the second is "Resource provisioning", in which the infrastructure is initialized, and the VNs allocated in the previous process are created. This paper focuses on the provisioning process that includes two approaches: Proactive and Reactive. The nodes that represent the resources are created in advance in the Proactive approach, and assigned to the applications as they arrive. As for the Reactive approach, they are created upon request. The reactive approach is the ideal case to operate in terms of saving the resources usage. However, the time delay caused by the time required to provide the VNs makes the proactive approach more suitable for time-sensitive applications. In this work, we aim to modify "Zeus" the proactive algorithm to fit an reactive approach, while modifying its virtualization model to achieve the maximum real time possible, and the most efficient resources usage.

Keywords: Cloud of sensors, Edge computing, Resource allocation, Resource provisioning, Virtualization.

* A Researcher Engineer has a Master's Degree from The Higher Institute for Applied Sciences and Technology - Damascus - Syria.

1. مقدمة

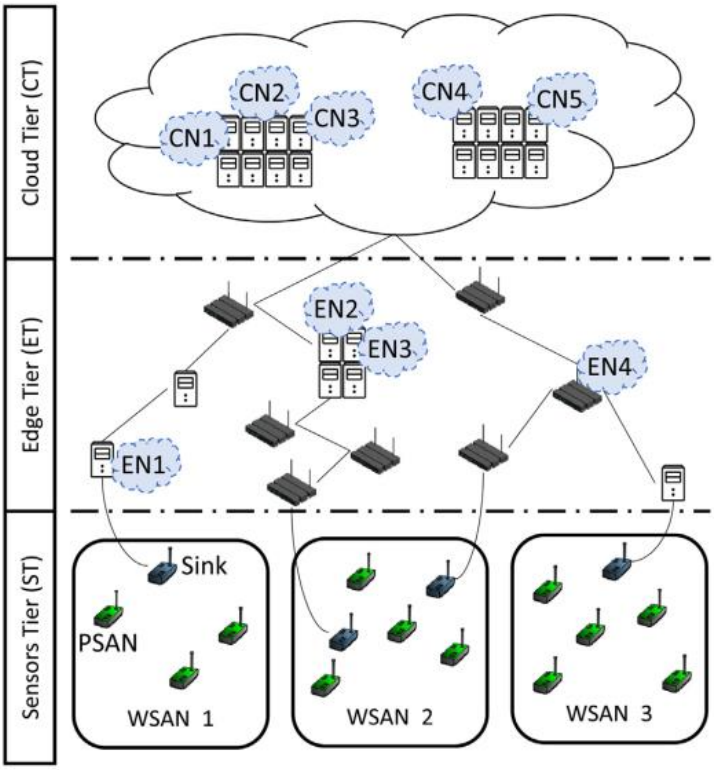
شهدت بدايات القرن الواحد والعشرين انتشاراً واسعاً للعديد من النماذج الجديدة التي أحدثت ثورةً في مجال تكنولوجيا المعلومات والاتصالات. أحد هذه النماذج كان انترنت الأشياء IoT (Internet of Things). ينطوي هذا المفهوم على بنية تحتية شاملة للشبكة، تربط مجموعة واسعة من الأجهزة الفيزيائية والافتراضية التي تسمى بالأشياء الذكية. تتمتع هذه الأشياء بقدرات معالجةٍ، واستشعارٍ، واتصالٍ، لدعم تطوير تطبيقات وخدمات تعاونية. تتسم أجهزة انترنت الأشياء بعدم التجانس، وتتنوع بين الحساسات القابلة للارتداء، إلى المركبات المزودة بأجهزة استشعار؛ يمكن توصيلها بالانترنت عبر وصلات سلكية أو لاسلكية، بمختلف بروتوكولات الاتصالات. بين هذه الأجهزة الذكية واسعة التباين، تلعب المستشعرات الذكية دوراً هاماً في نموذج انترنت الأشياء [1]. المستشعرات الذكية هي أجهزة صغيرة تعمل بالبطارية، تتمتع بإمكانيات المعالجة، التخزين، التحسس، التحرك، والقدرة على الاتصال اللاسلكي. تتيح إمكانيات الاتصال والتشبيك لهذه الأجهزة، تجميعها لإنشاء شبكة مستشعرات ومشغلات لاسلكية WSAN (Wireless Sensor and Actuator Network). تحتوي الـ WSAN –إضافة إلى عقد المستشعرات والمشغلات– على مصباتٍ Sinks، وهي نظرياً عقد بدون قيود على الموارد (مثل المقدرات الحسابية، والطاقة، والاتصالات) كعقد الاستشعار التقليدية، وترتبط ببوابات عبور تتيح الاتصال بين هذه الشبكات وبين الأنظمة والشبكات الخارجية كالانترنت. تتكامل عدة شبكات WSAN عبر الانترنت لتكون بمثابة الركيزة الأساسية لنظام IoT [2].

إن مشاركة البنية الفيزيائية للشبكات WSAN بين عدة تطبيقات هو الحالة النموذجية عند التعامل مع مقياسٍ عامٍّ شامل، ولكن الكمية المتزايدة باطراد للبيانات المنتجة من المستشعرات تتطلب قوة حسابية وقدرة معالجة عالية لتأمين خدمات فعالة زمنياً للمستخدمين النهائيين. دعت هذه الحاجة إلى دمج المنصات السحابية بمواردها الوفيرة مع هذه الشبكات لتُنشِئ أنظمةً معقدةً، موزعةً، موجَّهة للبيانات وعلى مقاييس كبيرة. سمح هذا الدمج بنقل بعض المهام المُسندة من المستشعرات الذكية إلى السحابة، لكن الحساسة منها زمنياً التي تحتاج لاستجابة سريعة، جعلت التأخير الطويل وغير المستقر بين المستشعرات والسحابة في هذا النقل أمراً غير مرغوبٍ به. إضافة لذلك، فإن تطبيقات الـ WSAN عادة ما تتطلب دعماً للتنقل وإدراكاً للمكان، وهو ما لا تدعمه المنصات السحابية عامةً. ثم إن النقل العشوائي للبيانات\المهمات إلى السحابة سيزيد في استهلاك عرض الحزمة في الشبكة، مع الأخذ بالاعتبار أنه في حالات عديدةٍ يمكن معالجة معظم البيانات المنتجة والانتهاء منها محلياً [3].

للتغلب على عيوب التكامل بين شبكات WSAN والسحابة، برز نموذج حوسبة الحواف كحلٍ واعد. كما في السحابة، يقترح نموذج الحواف محاكاةً افتراضية للتجهيزات الفيزيائية على شكل كينونات افتراضية تدعى عقد الحواف EN (Edge Nodes). هذه العقد تعمل على حافة السحابة وتؤمن للمستخدمين النهائيين القدرات الحسابية، التخزين والتشبيك. يمكن للتجهيزات الفيزيائية العاملة على الحواف أن تكون فقيرة بالموارد كالموجهات والمبدلات ونقاط الوصول والمحطات القاعدية أو حتى مستشعرات ذكية، أو يمكن أن تكون غنية بالموارد كمراكز بياناتٍ مصغرة أو تجهيزات كالـ Cloudlet. تقوم عقد الحواف بعدة مهام، فعَلى سبيل المثال، تُجمّع البيانات من المستشعرات الذكية، وتُرشحها

وتُعيد بناءها بشكلٍ أكثر فعالية، رافعةً البيانات الضرورية فقط للسحابة. كما يمكنها مراقبة نشاطات المستشعرات، والتحقق من استهلاكها للطاقة، مع ضمان أمن وخصوصية البيانات المُجمّعة.

بناءً على ما سبق، وسع نموذج الحافة مفهوم الحوسبة السحابية التقليدية إلى أطراف الشبكة، وحقق الجمع بين شبكات الـ WSAN ونماذج الحوسبة السحابية وحوسبة الحواف تعزيزاً للمزايا المشتركة فيها في بنية ثلاثية الطبقات. فمن جانبٍ، استفادت الشبكات WSAN من الموارد الحسابية الكبيرة للسحابة، ودعمِ نموذج السحابةالحافة للتأخير المنخفض، والإدراك المكاني، والتنقل، مما أمّن إدارةً وتكويناً فعالاً للخدمة لاستغلال المستشعرات الذكية وبياناتها المنتَجَة. على الجانب الآخر، مكنت شبكات الـ WSAN نموذجَ السحابةالحافة من توسيع نطاق تعامله مع سيناريوهات حقيقية بطيف واسعٍ من الخدمات بطريقة موزعةٍ وديناميكيةٍ. باستغلال أوجهِ التآزر آنفة الذكر ظهرت سحابةُ المستشعرات(Cloud of Sensors) CoS بمعمارية ثلاثية الطبقات كما في الشكل (1) [4].



الشكل (1): معمارية سحابة المستشعرات CoS.

بُنيت البنية التحتية لـ CoS على مبدأ المحاكاة الافتراضية لـ WSAN [5]، أمّن هذا فصلاً تاماً بين التطبيقات والبنية الفيزيائية لشبكة المستشعرات، وسمح بمشاركة عدة تطبيقات للعتاد الفيزيائي بآن واحد. لقد تم ذلك بتحديد متطلبات كل تطبيق، وإنشاء عقد افتراضية VNs لخدمته. أتاحت هذه التشاركية تخفيضاً للكلفة، وتجنباً للتداخلات السلبية بين التطبيقات المختلفة. بحيث تقوم شبكة فيزيائية مفردة بخدمة عدة مهام (مهمة لكل تطبيق) مما أدى إلى زيادة عائد الاستثمار على البنية التحتية من وجهة نظر مزود الخدمة. لقد سمح تبني هذا المفهوم بتنفيذ شبكات مُقادة بالمهام Mission-oriented بكفاءة، فوفّق بين تكييف شبكة WSAN لكل مهمة محددة من ناحية، وبين الحاجة لاستخدام الموارد بشكل أمثليٍّ من ناحية أخرى.

تماشياً مع مفهوم المحاكاة الافتراضية للـ WSAN، يُشكِّل نظام الـ CoS عدة VNs لتنفيذ التجريد المتوقع من عملية توفير البيانات. تُعرَّف العقدة الافتراضية ككيانٍ افتراضي (مُدْخل حسابي)، هدفها الرئيسي تجريد قدرات البنية التحتية الحسابية والتشبيكية للمستخدمين (وليس فقط البيانات). بما أن العقدة الافتراضية كيان افتراضيٌّ وحيد، فهي تُبسِّط تمثيل البنية الفيزيائية لهم، وتُجرِّدُ العُقَدَ التحتيَّةَ كخدمات برمجية تقدمها للتطبيقات. تتوسط هذه الخدمات التفاعل بين التطبيقات والعتاد الفيزيائي، وعليه تمثل الـ VNs **الموارد** في نظام الـ CoS. نُعرّف مما سبق الموارد على أنها مكونات برمجية تُقدِّم إما بيانات من عقد الاستشعار الفيزيائية، أو تتحكمُ بتشغيل هذا العتاد الفيزيائي وتحريكه.

يُعد تخصيص الموارد موضوعاً مُتداولاً في العديد من مجالات الحوسبة مثل أنظمة التشغيل، والحوسبة الشبكية (Grid Computing)، وإدارة مراكز البيانات. يهدف تخصيص الموارد في الحوسبة السحابية إلى ضمان التلبية الصحيحة لمتطلبات التطبيق من قبل البنية التحتية الفيزيائية التي تُجرَّد عبر آلات افتراضية (Virtual Machines)، مع تخفيض الكلفة التشغيلية لبيئة السحابة لأقصى حد ممكن [6]. يتشابه هذا الهدف مع مفهوم تخصيص الموارد في الـ CoS، الذي يشير إلى عملية تخصيص الكينونات افتراضية VNs لتنفيذ طلبات التطبيق (حمل العمل) الواردة إلى السحابة من المستخدمين، ومحاولة تلبية أقصى قدر ممكن من حاجات التطبيقات، مع مراعاة القيود المفروضة على الـ VNs الناتجة عن البنية التحتية الفيزيائية. يختلف تخصيص الموارد في الـ CoS عن مثيله في الحوسبة السحابية التقليدية بتعامله مع خصوصية البنية الفيزيائية ثلاثية الطبقات (طبقات المستشعرات والحواف والسحابة). إن المورد الرئيسي في الـ CoS الذي تُقدّمه الـ VNs إلى التطبيقات هو البيانات المُحصّلة من البنية الفيزيائية، بعكس القدرات الحسابية والتخزينية وقدرات الاتصال في الحوسبة السحابية التقليدية. بناءً عليه، يجب تصميم حلول تخصيص الموارد في الـ CoS لتحقيق هدفين متعارضين: الأول تلبية أكبر قدر ممكن من المتطلبات التي تفرضها طلبات التطبيق، والثاني البحث عن أقل استهلاك ممكن للموارد المقدمة من الـ VNs.

من الضروري في البداية وجود عملية لإنشاء العقد الافتراضية من أجل تحديد واختيار البنية التحتية الفيزيائية المناسبة للاستخدام، ويُشير مصطلح توفير الموارد (Resource provisioning) في سياق الـ CoS لعملية إنشاء هذه العقد. إن مصطلح "**التوفير**" يدل غالباً على تجهيز أو إعداد بنية تحتية ما لغرض معين. في الحوسبة السحابية التقليدية تكون عملية توفير الموارد مسؤولة عن إدارة الربط بين القدرات الحسابية الفيزيائية ومقابلاتها من الكيانات الافتراضية. هذا الربط في الـ CoS يجب أن يجعل استفادة العقد الافتراضية من القدرات الحسابية الفيزيائية أعظم ما يمكن، مع مراعاة القيود المفروضة على البنية التحتية المادية لـ CoS. يشير توفير الموارد الاستباقي Proactive إلى عملية تأسيس العقدة الافتراضية وتجهيزها قبل تخصيصها لطلب التطبيق، بحيث تُنشَأ العقد الافتراضية من قبل مزود البنية التحتية Infrastructure Provider (InP) بوقت سابق لتنفيذ البنية التحتية لـ CoS ووصول التطبيقات. في حالة توفير الموارد التفاعلي Reactive تُنشأ العقدة الافتراضية (إضافة إلى إعادة استخدام العقد الموجودة سابقاً) استجابة للحاجة لتخصيصها لطلبٍ (بمعنى آخر يتم تصميم VN جديدة لتلبية تخصيص محدد)، وعليه يتم إنشاء العقد الافتراضية بخوارزمية توفير موارد آلية، ديناميكية بزمن حقيقي. عموماً يجب أن تتم عملية التوفير والتعديل على الـ VNs وفقاً للطلب بطريقة مرنة، واضحة، وقابلة للتوسع [7].

يُعرّف التطبيق الذي يقدم الخدمات للمستخدمين، والمتصل مع الـ CoS على أنه مجرى عمل Workflow [8] يصف التفاعلات بين الخدمات التي تقدمها العقد الافتراضية. إن كل تطبيق Application يحتوي على مجموعة من الطلبات Requests، وهي عبارة عن نشاطات Activities في مجرى العمل الموَصَّف، تحتاج إلى موارد البنية

التحتية لسحابة المستشعرات. يُوصَّف الطلب على أنه مجموعة من الأوامر Commands المجردة المُعرَّفة من قبل المُشغّلين، والتي تمثل المتطلبات الوظيفية Functional للتطبيق، فعليه يُكافئ الطلب خدمة Service مجردة واحدة. كما له أيضاً متطلبات أخرى غير وظيفية Non-functional من أبرزها في الأنظمة الموزعة القائمة على بيانات الاستشعار متطلب حداثة البيانات (Data freshness DF)، ومتطلب زمن الاستجابة (Response Time RT). تزداد أهمية متطلب الحداثة في الأنظمة الشاملة على عدة مصادر مستقلة للبيانات، حيث يمكن أن يؤدي تكامل بيانات مختلفة الحداثة إلى مشاكل دلالية تعيق تنفيذ التطبيقات. توجد عدة تعريفات للحداثة في الأدبيات، التعريف الأبرز بينها هو الذي يحدد حداثة بيانات معينة بناءً على الوقت المُنصرم منذ تحصيلها، أما زمن الاستجابة فيمثل الزمن المنقضي قبل أن يبدأ النظام بمعالجة الطلب [9].

يمكن لطلبات التطبيق أن تمتلك علاقات تَبَعيّة، بمعنىً آخر يمكن لها أن تملك علاقات أسبقية فيما بينها، تعتمد على بعضها البعض وفق تسلسل زمني (مثلاً لا يمكن معالجة البيانات قبل جمعها). يُعبَّرُ عن التطبيق بِبَيان مُوجّه خالي الحلقات (Directed Acyclic Graph) نرمز له بـ DAG G [10]، وفق المعادلة (1).

$$G = (V, E, w, c) \quad (1)$$

في هذا البيان تمثل كل عقدة Vertex (في المجموعة V) طلباً ما، في حين يمثل كل قوس موجه (Directed Edge) (في المجموعة E) قيد الأسبقية بين العقد المعنية، بحيث تُنفّذ عقدة الأصل قبل الوجهة. تمثل كل من w و c أوزاناً (موجبة القيمة) مُسندة لعقد وأقواس الـ DAG على التوالي. لكل طلب، يُوصّف الوزن w الكلفة الحسابية، والوزن c كلفة الاتصال. جميع الـ DAGs من كل التطبيقات الواردة تتصل بـ DAG فريد يدعى UG، هدفه تأمين معاينة شاملة لكل التطبيقات الواردة خلال النافذة الزمنية التي يحدث فيها التخصيص.

تصل التطبيقات إلى الـ CoS بشكلٍ متزامنٍ عادةً عن طريق طبقة السحابة أو الحواف، عبر نقاط دخول التطبيقات Application Entry Points (AEPs)، إما عبر الانترنت، أو من أداة للمستخدم (حاسب أو هاتف ذكي إلخ...) موصولة مباشرة إلى عقد الحواف. إن كل عقدةٍ سحابية كانت أم حواف، مرشَّحةٌ محتَمَلةٌ لتشغيل الـ AEP. تقوم البيئة التحتية للـ CoS بتلبية طلبات هذه التطبيقات عن طريق تأمين الخرج لها (الخرج هنا هو بيانات في حال كان الطلب استشعاراً، أو تحكماً إذا كان الطلب تشغيلاً). بناءً عليه، ستقوم البنية التحتية بتنفيذ المهام عن طريق توليد حملِ معالجةٍ معين على العقد الفيزيائية لتأمين الخرج بأقصى درجة حداثة ممكنة. لتجنب إعادة تنفيذ أحمال المعالجة، يمكن إعادة استخدام الناتج من خرج عمليات منفذة سابقاً، إذا كانت تناسب متطلب الحداثة.

على الرغم من المزايا الممكنة للـ CoS، لا تزال هناك العديد من التحديات للتعامل معها من أجل تمكين النموذج بالكامل عند تصميم مثل هذه البنى التحتية. أحد أبرز التحديات يتمثل في تطوير حلول لتنفيذ تخصيص الموارد فيها أي حل مشكلة الأمثَلَة المصاغة رياضياً [11].

2. الدراسة المرجعية

تختلف الأعمال في الأدبيات ذات الصلة بتخصيص الموارد في طبقتي السحابة والحافة لمعمارية الـ CoS باختلاف الخوارزمية المستخدمة لحل مسألة الأمثلة. تقع هذه المشكلة ضمن الصنف القياسي لمشاكل البرمجة اللاخطية الصحيحة المختلطة Mixed Integer Non-linear Programming Problems (MINPP). وفيه تكون متغيرات القرار بقيمٍ صحيحة (0 أو 1) Binary من أجل التخصيصات المكانية التي تحدد العقد المنفذة لقرار التخصيص، وقيماً حقيقية Real من أجل التخصيصات الزمانية التي تحدد اللحظات في الزمن التي تبدأ فيها العقد بمعالجة الطلب.

توجد عدة منهجيات لحل هذه المشكلة مثل نظرية البيان أو اللعبة أو التعلم العميق وتعلم الآلة، بالإضافة للطرق الاحتمالية. إلا أن مُشكلة الأمثَلة المُصاغَة من نوع المشاكل NP-complete، وطبيعتها المتضخمة أسياً عندما تكبر حجوم الـ DAGs تعيق البحث السريع عن حلولٍ مُثلى. يضرّ هذا الجانب بالتطبيقات الحساسة للزمن في سيناريو حوسبة الحواف، التي تتطلب عادةً استجابةً سريعةً. في هذا السياق إذاً، يبرز تحدٍ آخر يتعلق بكيفية حل الأمثِلة العملية لـ MINPP ذات الأحجام العشوائية في وقت متعدد الحدود Polynomial Time. يوجد مجالٌ بحثي كامل يتعامل مع البحث عن حلول سريعة للمشاكل NP-complete، مع التركيز على التقنيات الإرشادية (Heuristic techniques) التي تعطي حلولاً شبه مثالية Near-optimal، وتُظهِر حِملاً حسابياً مُنخفضاً. كما أنَّ معاييراً مثل تواجد الأولويات بين طلبات التطبيق المختلفة، والقدرة على التعامل مع علاقات الأسبقية (التبعيات بين المدخلات والخرجات) بين الطلبات، ومشاركة نتائج التنفيذ المشتركة بينها بحيث تُحَصَّل البيانات مرة واحدة، ذات أهمية عُظمى لتقييم الخوارزميات المقترحة.

إن مفهوم تخصيص الموارد ليس بجديد في الأدبيات، حيث تدرس عدة مجالات بحث مرتبطة ارتباطاً وثيقاً مع الـ CoS هذا المفهوم مثل انترنت الأشياء والـ WSAN والحوسبة السحابية التقليدية مع اختلافات بسيطة في السياق. تتعلق هذه الاختلافات بشكل أساسي بطبيعة الموارد المراد تخصيصها. سنستعرض فيما يلي الأعمال الموجودة المتعلقة ببحثنا في الـ CoS، ولكن نظراً لحداثة عهد هذا المجال سننظر لأحدث الحلول المقترحة بمنظور أوسع، يشمل ذلك انترنت الأشياء والـ WSAN.

من مجال الـ CoS، اقترحت Delgado وآخرون [12] [13] خوارزميةً إرشاديةً، وإطار عمل لأمثَلة أداء تخصيص الموارد. باحثين عن زيادة عدد التطبيقات التي تتشارك الـ CoS لأقصى حد، مع مراعاة متطلبات التخزين المحدود، قوة المعالجة، عرض الحزمة، واستهلاك الطاقة في طبقة المستشعرات، وتتبع خوارزميتهم تقنية برمجةٍ خطيةٍ.

قدم Wang وآخرون [14] إطار عملٍ Edge NOde Resource Management (ENORM) لمواجهة تحدي إدارة الموارد على مستوى الحافة. يُمَكِّنُ النموذج المقترح ENROM الخدماتَ السحابية من تفريغ بعض أعباء العمل على العقد الحافية. وبمقياسٍ آلي Auto-scaling يأخذُ توافرية الموارد في الحافة بالحسبان عند تخصيص أو إلغاء تخصيص الموارد المقدمة لحمل العمل. استطاع ENROM تخفيض تأخير التطبيقات بنسبة تتراوح بين 20% و 80%، ونقل الملفات وتكرارية الاتصال بين السحابة والحافة حتى 95%. صُمِّمَت حلول التوفير والتوسع الآلي في هذا النموذج كحلولٍ مركزية. من سلبيات هذا النموذج عدم اعتباره لعلاقات الأسبقية بين طلبات التطبيق نفسه، وللطلبات المشتركة بين عدة تطبيقات.

اقترح Igor وآخرون [15] خوارزمية Zeus، وهي خوارزمية تخصيص موارد لسحابة من المستشعرات CoS. وسَّع الباحثون نموذجهم السابق Olympus (ثنائي الطبقة) إلى معمارية ثلاثية الطبقات تشمل بذلك السحابة والحافة

والمستشعرات. كما قدموا صياغة رياضيةً من النوع MINPP لمسألة تخصيص الموارد في هذه البنية، إضافة لنموذج محاكاة افتراضية للـ CoS، وفرت به الخوارزمية مجموعة من العقد الافتراضية VNs للتطبيقات وفصلتهم بذلك عن البنية التحتية. تتسم الخوارزمية بسمتين أساسيتين: الأولى هي القدرة على تحديد الطلبات المشتركة للتطبيقات الواردة، وأداء المهام المطلوبة مرةً واحدة وفق أكثر المتطلبات صرامةً، ثم مشاركة النتائج بين التطبيقات توفيراً للموارد. والثانية نهجها الهجين الذي يعزز مفهوم حوسبة الحواف، جاعلاً الخوارزمية قابلةً للتوسع من ناحية العقد الافتراضية والتطبيقات المدعومة، ومناسبةً لتلبية التطبيقات الحساسة للتأخير.

أما في مجال الـ IoT، قدم Yu وآخرون [16] شبكة للمركبات قائمة على السحابة Cloud-based vehicular network، مُدارَةً باستراتيجيةٍ مبنيةٍ على نظرية اللعبة لأمثَلَة تخصيص الموارد. استخدم الباحثون نهجاً لامركزياً، بحيث تتنافس الآلة الافتراضية Virtual Machine VM مع الـ VMs الأخرى على الموارد من وجهة نظرها المحلية. استهدف الباحثون تلبية متطلبات جودة الخدمة QoS، مع مراعاة قيود استخدام موارد البنية التحتية الحسابية والتخزينية.

قدم Xu وآخرون [17] نموذجاً مسمىً Zenith، وهو نموذجُ تخصيص موارد مركزي يعتمد حوسبة الحواف. يسمح نموذجهم لمزودي الخدمة بإبرام عقودِ مشاركةٍ للموارد مع مزودي البنية التحتية للحافة. استناداً على هذه العقود المبرمة، يستخدم مزودو الخدمة خوارزمية تخصيص موارد مُدْركةٍ للتأخير latency-aware تتيح تشغيل الطلبات الحساسة للتأخير حتى اكتمالها. تُبنى الخوارزمية على مزادٍ auction-based يضمن المصداقية وتعظيمِ أرباح كلٍّ من مزودي الخدمة ومزودي البنية التحتية لحوسبة الحواف لأقصى حد.

3. هدف البحث

من خلال الدراسات ذات الصلة وجدنا أن الخوارزمية الأمثل لتخصيص الموارد هي خوارزمية Zeus، فإضافة لكونها تمثل الحل الرياضي الأمثل للمسألة (وفق نتائجها) من بين الأعمال الأخرى المقترحة، فهي أيضاً جمعت مزايا متعددة لم تتوفر جميعها في أي عملٍ من الأعمال الأخرى، كدعمها لحوسبة الحواف من أجل التطبيقات الحساسة للزمن، وأخذها علاقات الأسبقية بين طلبات التطبيقات بعين الاعتبار، وتنفيذ الطلبات المشتركة مرة واحدة ثم تعميم النتيجة، مع قابلية للتوسع من ناحية دعم التطبيقات والـ VNs الخادمة لها، يضاف لذلك نهجها الهجين (اللامركزي جزئياً) حيث أن اللامركزية وعدم التجانس طبيعة متأصّلةٌ في شبكات الـ WSANs، وإن أيَّ نموذجٍ يدمج هذه الشبكات بالحوسبة السحابية ذات الطبيعة المركزية لا بُدَّ من أن يتصدى لهذا التناقض.

تبنت الخوارزمية آلية استباقية لتوفير الموارد، لأنها تسبب حِملاً أقل Lower Overhead كونَ الـ VNs تُنشَأ قبل بدء استقبال الطلبات. إلا أن هذه العقد المنشأة قد تزيد عن حاجة النظام، وبقاؤها دون إسنادٍ سيسبب هدراً للموارد. من هنا برزت الحاجة لآلية تفاعلية تُنشِئ الـ VNs وفقاً لحاجة التطبيقات عند ورودها في أقصى زمنٍ حقيقيٍّ ممكن، مع التصدي لأبرز سلبيتها ألا وهي الحمل العالي على الشبكة، والبطء الزمني الناتج عن الوقت اللازم لتوفير الـ VNs. إن هدف البحث اقتراح نهج تفاعلي للخوارزمية يحل كل من السلبيتين آنفتي الذكر. فلحل مشكلة الحمل العالي على الشبكة سنعدل نموذج المحاكاة الافتراضية فيها. تمثل الـ VN حجر الرَّحى في نظام المحاكاة الافتراضية لسحابة المستشعرات، وتُعرَّف في النموذج الخاص بـ Zeus على أنها

برنامجٌ Program يُنفِّذ مجموعة من تقنيات دمج المعلومات Information Fusion. في هذا النموذج يُبنى سلوك الـ VN (VN behavior) من الصفر، وينقل في الشبكة عند كل عملية توفير لها [18]، ولا بدَّ لتوفير أسرع يخدُم الآلية التفاعلية من تعديل هذا النموذج. لم تدعم الخوارزمية وجود أولويات بين طلبات التطبيق، وهو معيار هام جداً لتعظيم مدخلات النظام، فمثلاً تطبيق التدفئة والتهوية والتكييف HVAC (Heating, Ventilation, and Air Conditioning) يمكن أن يكون أقل حراجة من تطبيق كشف الحرائق، ومستخدمون من شرائح سعرية عليا أولى بتخديم النظام لهم من ناحية الثمن المدفوع من قبلهم.

4. طرائق البحث ومواده

ركزت الأعمال ذات الصلة بتخصيص الموارد في مجال الـ CoS والمجالات القريبة في انترنت الأشياء IoT على مَنحيَين بالبحث: الأول هو صياغة الحل رياضياً بالسعي نحو الخوارزمية الأمثل بالسرعة والدقة، والثاني على نماذج المحاكاة الافتراضية. لتحقيق أهداف هذا البحث سنحاول الدمج بين كلا الاتجاهين. لقد وجدنا سابقاً أن خوارزمية Zeus هي الخوارزمية الأمثل لتخصيص الموارد (حتى تاريخ البحث) مقارنة مع غيرها من الأعمال المتعلقة. ومن نماذج المحاكاة الافتراضية اخترنا نموذج محاكاة خفيف الحمل هو LW-CoEdge [18]. يوافق هذا النموذج أهدافنا في توفير العقد الافتراضية بأسرع وقت ممكن استجابة لطلبات التطبيقات، فأساس مشكلة تخصيص الموارد –الـ VNs في نظامنا– في النهج التفاعلي هو نشرها وتوزيعها بالزمن الحقيقي الأمثل. يرتبط هذا النموذج بمنصة الـ FIWARE [19]، وهي منصة مدعومة من الاتحاد الأوروبي، توفر عدة عناصر تسمى Generic Enablers (GEs)، لبناء منظومات انترنيت الأشياء، وتأمين مستلزمات تطوير تطبيقاتها، موفرة بيئة مفتوحة ومجانية لمقدمي الخدمات والشركات والمؤسسات الأخرى.

بُنيَ LW-CoEdge على تقنيتين أساسيتين: المحاكاة الافتراضية خفيفة الحمل Lightwight Virtualization والخدمات المصغرة Microservices. تسمح الأولى بنشر وتوزيع عناصر النموذج كحاويات Containers، والحاوية نظام محاكاة افتراضية مصغر تُستضاف فيه نواة نظام لينوكس Linux Kernel في نظام التشغيل الأساسي، تلعب فيه دور غلاف حاوٍ لتطبيق ما، بحيث يصبح مستقلاً قائماً بذاته مما يزيد في توفير الموارد. تُكْمل التقنية الثانية تحقيق أهداف النموذج بفصل عناصر النظام بأن تكون لكلٍ منها مسؤوليةٌ محددةٌ لا تتقاطع مع الأخرى. لبناء سحابة المستشعراتCoS بواسطة النموذج LW-CoEdge، وسنوصف إطاره البرمجي [20] كما يلي: يستخدم النموذج لغة البرمجة Java في إطار Spring boot framework لبناء العقد الافتراضية كخدمات مصغرة Microservices كحال بقية عناصر النظام الأخرى. ولتشغيلها تُركَّب هذه الخدمات في حاويات Docker Containers بحيث تؤمن المحاكاة الافتراضية خفيف الحمل. تُضمَّن العقد الافتراضية الممثلة لمختلف أنواع البيانات مسبقاً في كل عقدة حافية. في هذه الحاويات تُنصَّب آلة Java افتراضية بإصدار 64 bit تحتوي الرمز البرمجي لكل عنصر من عناصر النظام. يستخدم النموذج بروتوكول الاتصال REST/HTTP (Representational State Transfer/ Hypertext Transfer Protocol) القياسي للتواصل بين الخدمات المصغرة. أما الرسائل المتبادلة في النظام فتنفذ بصيغة JSON (JavaScript Object Notation). سنستخدم في تجربتنا نوعين من الحساسات (وبالتالي نوعين من البيانات)، حساس للحرارة وآخر للإضاءة. وهما موجدان بشكل افتراضي ضمن إطار الـ FIWARE Framework [21]. تقوم هذه الحساسات بإرسال بيانات التحسس لقاعدة بيانات محلية، ويقوم العنصر

GE Orion Broker بجلبها وتقديمها للعقد الافتراضية. إن حجم الرسالة المطلوب لجلب بيانات التحسس مباشرة من المستشعرات هو 122 byte بشكلٍ وسطي، وحجم الرسالة التي ترسلها الـ VN للنقطة التي ورد منها التطبيق 72 byte. يقدر التأخير الناجم عن الاستعلام عن بيانات التحسس من الـ GE Orion Broker بـ 0.019 ± 0.972 sec وهذه الأرقام مضبوطة مسبقاً وفقاً لإطار العمل. تمت محاكاة إطار الحافة بإنشاء عقد حافية بواسطة بيئة VMware مشغلة على جهاز من النوع Intel (R) Core (TM) i7-8750H CPU @ 2.20GHz وذاكرة وصول عشوائي 16 GB RAM. تُنصب كل عقدة حافية بـ Virtual Machine كنظام تشغيل CentOS 7 64-bit، ويحجز لها 2 GHz من الـ CPU، و 1.5 GB من الـ RAM، و 5 GB من القرص الصلب HD.

5. التعديل المقترح

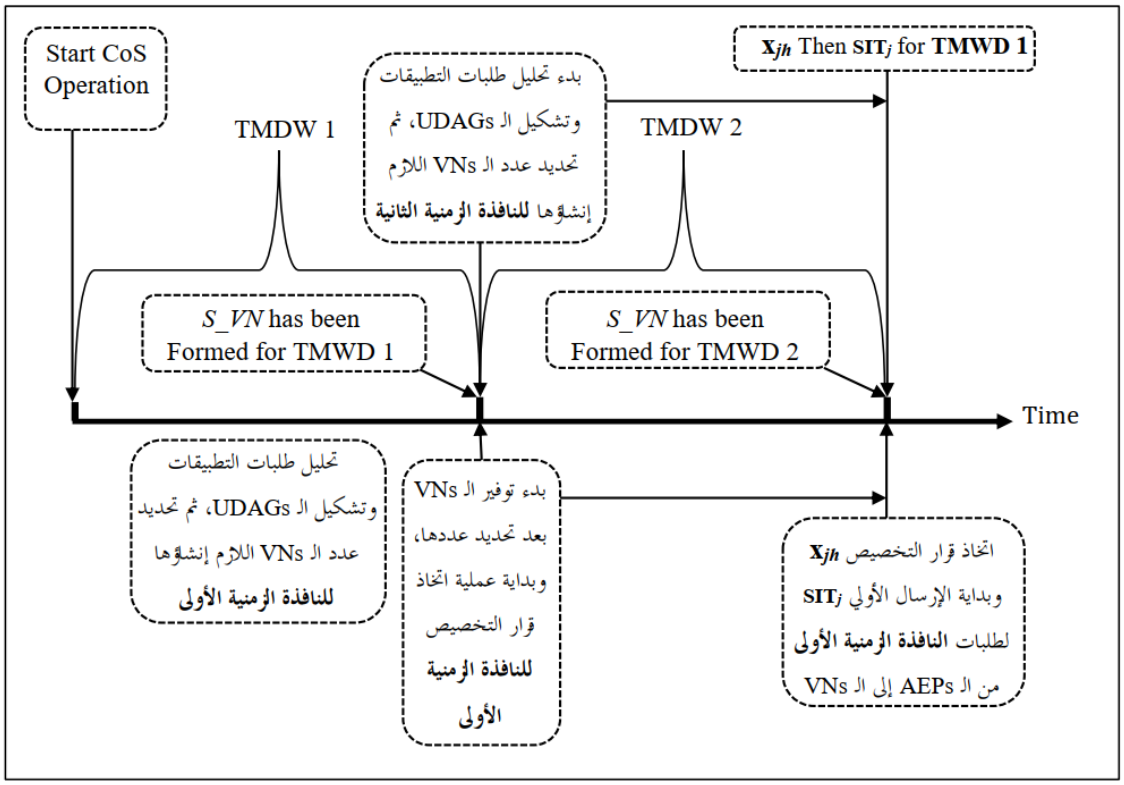
ينقسم التعديل المقترح إلى جزئين، الأول يخص خوارزمية Zeus نفسها، والثاني يخص نموذج المحاكاة الافتراضية فيها. بالنسبة للجزء الأول، تعمل الخوارزمية كحل مباشر online عبر الانترنت، وذلك أنها تتبنى نهجاً استباقياً لتوفير الموارد، بمعنى آخر تكون الـ VNs جاهزة للتخصيص عند قدوم طلبات التطبيقات في أي لحظة. لتمكين نهج تفاعلي لا بد من مضي فترة زمنية محددة تُستقبل فيها الطلبات، ثم يُصار إلى إنشاء الـ VNs. أي لا بدّ من تقسيم الخوارزمية لنوافذ زمنية. اقترحت Oliveira وباحثون آخرون في [22] صياغة لمسألة أمْثَلة تخصيص الموارد في الـ CoS بدون حل هذه المعادلات. لقد قسموا الزمن إلى نوافذ تدعى نوافذ القرار TMDW (Time Decision Window) تُستَقبَل فيها التطبيقات القادمة بمخزنٍ مؤقت Buffer، بحيث يُتَّخَذ القرار لمجموعة من التطبيقات بوقت واحد على كامل نافذة قرار. عندما ينتهي الـ Buffer يكون قرار التوفير والتخصيص قد أُتِّخِذ، ثم يبدأ تنفيذ القرار بإنشاء الـ VNs اللازمة للتطبيقات ثم إسناد الطلبات لها. حدد الباحثون مكتسبات النظام القائم عليها ربحه بتلبية متطلبات الحداثة –كما في Zeus– مع متطلب الاستجابة الزمنية للتطبيق. لتحسين أداء خوارزميتنا المعدلة اقترحنا تقسيم التطبيقات إل شريحتين سعريتين. الأولى لها الأولوية عند الاستجابة لتطبيقاتها. تمثل الاستجابة الزمنية للتطبيق الزمن اللازم الذي سينتظره في الرتل حتى البدء بتخديمه، وبالتالي عند قدوم تطبيق بأولوية وضمان تخديمه سنكون قد عززنا مكتسبات النظام. لم تأخذ Zeus بعين الاعتبار الأولويات بين التطبيقات، وهي الميزة الناقصة عن بعض الأعمال الأخرى كما وجدنا سابقاً.

يمكن حل مسألة الـ MINPP في أيٍّ من طبقتي السحابة أم الحافة، بحيث تكون رؤية حل تخصيص الموارد أكثر شموليةً في السحابة، وأكثر محليةً في الحافة. تعتمد Zeus على الطبقتين لتنفيذ حلٍّ هجين. سيجبرنا حجم مدخلات المسألة وسرعة معالجة الـ CPU المتوفرة على الرضا بحل تقريبيٍّ دون المستوى الأمثل بخوارزمية إرشادية، ولكن بوقتٍ كثير الحدود. إذاً، تنقسم عملية اتخاذ القرار إلى طورين، طورٌ أول مركزي، يكون في نقاط دخول التطبيقات AEPs، وطورٌ لا مركزي يكون في العقد الافتراضية المنشأة VNs. في الخوارزمية المقترحة سنُعنى باتخاذ ثلاثة قرارات تمثل متغيرات القرار، اثنان مكانيان بقيمٍ صحيحة (0 أو 1) هما x_{jh} و y_j ، وواحد زمني بقيمةٍ حقيقيةٍ هو SUT_j. في الطور الأول سيُتخذ القرار x_{jh} الذي يحدد الـ h VN

المضافة في الـ Host h الخاصة بكل طلب j. أما في الطور الثاني فسيُتخذ القرار $\mathbf{y_j}$ الذي يحدد ما إذا كانت الـ VN المخصصة ستُحَدِّث بياناتها، ثم القرار $\mathbf{SUT_j}$ الممثل لبداية هذا التحديث.

5-1. الطور الأول المركزي:

ينقسم تنفيذ الطور الأول الموضح بالشكل (2) على نافذتي قرار زمنيتين. في النافذة الأولى، يبدأ النظام بتحليل طلبات التطبيقات الواردة في الـ AEPs، وتحديد المشترك منها، مشكلاً البيان الموجه الفريد UDAG. ثم تنتهي النافذة بتحديد عدد الـ VNs اللازم إنشاؤها. يتلو ذلك في النافذة الثانية ثلاثة مسارات متوازية: يبدأ الأول بتوفير الـ VNs في النظام، والثاني بعملية اتخاذ القرار $\mathbf{x_{jh}}$، والثالث بتشكيل الـ UDAG الفريد للمجموعة الثانية من الطلبات الواردة، وهكذا دواليك تتوالى النوافذ الزمنية. بنهاية النافذة الثانية يكون القرار $\mathbf{x_{jh}}$ قد أُتخِذَ، وعند تمام توفير الـ VNs، تبدأ الـ AEP بالإرسال الأولي $\mathbf{SIT_j}$ (Start Initial Transmission) للطلبات إليها. يقوم جوهر الآلية التفاعلية على توفير الـ VNs بإنشائها حسب حاجة التطبيقات الواردة بأفضلِ زمنٍ حقيقيٍّ ممكن، بعكس الآلية الاستباقية التي تكون العقد الافتراضية فيها موفرةً سابقاً في النظام، ثم تُستند إلى التطبيقات بمجرد ورودها. لا بدَّ من التأكيد هنا على أن الطلبات في الآلية التفاعلية –القائم عليها بحثنا– تُسند للـ VNs التي اِبتُدِئَ توفيرها بنهاية النافذة الزمنية السابقة، وأن غاية قرار التخصيص توزيع الطلبات على الـ VNs بأفضل شكل ممكن. بمعنىً آخر، لا بدَّ من ضمان مرور نافذة زمنية من بدء تشغيل النظام، قبل ضمان تنفيذ قرار التخصيص.



الشكل (2): التمثيل الزمني للطور الأول.

تمثل الخوارزمية (a-1) في الشكل (3) الخوارزمية المقترحة للنافذة الزمنية الأولى في الطور الأول. تبدأ بتعريف UDAG فارغ (السطر 2)، وتشغيل مؤقت زمني (السطر 3) مدته مساوية لنافذة القرار TMDW (يحددها النظام قبل عملية التخصيص). تصف السطور المتبقية كيفية تشكيل الـ UDAG.

تنقسم متطلبات التطبيقات الواصلة إلى متطلبات قابلة للتفاوض هي حداثة البيانات وزمن الاستجابة للتطبيق، وغير قابلة للتفاوض هي نوع البيانات. تحدد أنواع البيانات المشغلة في النظام مسبقاً، وتتجلى مكاسبه في تلبية متطلبات الحداثة والاستجابة الزمنية للمستخدمين. سنعالج في هذا الطور متطلب الحداثة، وفي الطور الثاني متطلب الاستجابة. تصل التطبيقات إلى الـ AEP (في السطر 6)، التي تبدأ بتقييم الطلبات وتشكيل عقد الـ UDAG بمرورها على كل طلب rq لكل تطبيق على حدة (السطر 7). نتيجة لذلك إما ستولِّد طلباً فريداً جديداً urq في الـ UDAG (السطر 16)، أو ستدمج الطلب rq بـ urq موجود سابقاً (الأسطر 10–15). يرجع التابع cmp_non_neg_rr (في السطر 10) إيجاباً true في حال تطابق نوع البيانات للطلبين rq و urq. أما التابع urq.update_fresher_neg_rr فيختار القيمة الأحدث (الأقرب للصفر) لمتطلب حداثة البيانات بين الـ urq و الـ rq الذي سيدمج معه. يضمن هذا التابع أن قيم الحداثة للطلبات المشتركة موافقةٌ لأكثرهم صرامةً. في السطر (12) يخزن الـ urq أولوية الطلب، وهي أولوية ساكنة Static لا تتغير أثناء التنفيذ، يحددها النظام الفرعي لمدير التطبيقات AMS وهي بدورها تحدد سرعة استجابة النظام للتطبيق الذي ينتمي إليه الطلب أثناء عملية توفير البيانات في الطور الثاني. أما في السطر 13 فيخزن الـ urq معرف الطلب rq_id المدمج معه، وبالعكس في السطر التالي (14) يتتبع الطلب الـ urq الذي دُمِجَ فيه. في الأسطر (17–24) تعود الـ AEP وتتكرر لتشكيل أقواس الـ UDAG الممثلة لعلاقات الأسبقية الفريدة upr بين طلبات التطبيقات، على غرار الـ urq. ثم يعاد ضبط المؤقت الزمني بعد انتهاء النافذة الزمنية.

```
Input: buff_tm, rule_fcen
Output: SVN

1.  def fcen(rule_fcen, buff_tm, SVN): //1st PHASE
2.      UDAG ← {Ø};
3.      buffering_timer.start(buff_tm);
4.      while True:
5.          // Begin UDAG formation
6.          for each app in new_apps_arriving():
7.              for each rq in app.getDAGnodes():
8.                  related_urq_found ← False;
9.                  for each urq in UDAG:
10.                     if cmp_non_neg_rr(rq, urq):
11.                         urq.update_fresher_neg_rr(rq);
12.                         urq.add_related_prty(rq, rq.app_ prty);
13.                         urq.add_related_rq(rq, rq.app_id);
14.                         rq.set_related_urq(urq);
15.                         related_urq_found ← True;
16.                 if !related_urq_found: UDAG.addurq(rq);
17.             for each pr in app.getDAGedges():
18.                 related_upr_found ← False;
19.                 for each upr in UDAG:
20.                     pre_found ← (pr.DAGPre == upr.pre);
21.                     pos_found ← (pr.DAGPos == upr.pos);
22.                     if !(pre_found and pos_found):
23.                         related_upr_found ← True;
24.                 if !related_upr_found: UDAG.addupr(pr);
25.         if buffering_timer.expired():
26.             buffering_timer.reset();
27.         // End UDAG formation
```

(a)

```
Input: buff_tm, SVN, rule_fcen
Output: decision xjh

1.  buffering timer.start(buff_tm);
2.  while True:
3.    //Begin deciding xjh
4.    pairs ← {Ø};
5.    for each urq in UDAG:
6.      for each v in SVN:
7.        if eval_non_negotiable_rr(urq,v):
8.          pairs += {(urq,v)};
9.    x ← choose_best_pairs(pairs, rule_fcen);
10. // End deciding xijp
11. for each urq in UDAG(): urq.fill_tVN(x);
12. for each urq in UDAG():
13.   urq.fill_suc_tVN(UDAG);
14.   dispatch(urq);
15. UDAG← {Ø};
16. if buffering_timer.expired():
17.   buffering_timer.reset();
```

(b)

الشكل (3): النافذة الزمنية الأولى (a)، والنافذة الزمنية الثانية (b) للطور الأول في الخوارزمية المقترحة.

تمثل الخوارزمية (b–1) في الشكل (3) الخوارزمية المقترحة للنافذة الزمنية الثانية في الطور الأول. في هذه النافذة تبدأ الـ AEP بعملية اتخاذ القرار $\boldsymbol{x_{jh}}$ في الأسطر (4–9). من (4) إلى (8) تُشَكَّلُ أزواج من الـ rqs والـ VNs. وفي السطر (7) يقوم التابع eval_non_negotiable_rr بتصفية جميع الأزواج السابقة، بحيث يُبقي على تلك التي تفي بالمتطلبات غير القابلة للتفاوض. يختار التابع choose_best_pairs (السطر 9) أفضل الأزواج وفق قاعدةٍ ما معطاة (parameter rule_fcen). اعتمدنا في بحثنا على قاعدة تقليل الانتظار في الرتل Queue Time Reduction Rule (QTRR) بهدف تحقيق أفضل استجابة زمنية للتطبيقات، بمعنى آخر توزع الطلبات على الـ VNs المنشأة لتحقيق التوازن الأفضل في طول الأرتال. في السطر (11) يُزَوَّدُ كل urq بالمعلومات المتعلقة بالـ VN المخصصة له، في حين يُزَوَّدُ كل urq بالمعلومات المتعلقة بالـ VNs المخصصة لخلفائه successors في السطرين (13,12). في السطر (14) يبدأ الإرسال الأولي SIT للـ urq إلى الـ VN خاصته، ونفترض أن لكل الطلبات زوج واحدٌ ممكنٌ على الأقل من الـ rq والـ VN. يمحو السطر (15) الـ UDAG، ويُعاد ضبط المؤقت في السطر (17) لتبدأ نافذة زمنية جديدة. في القسم التالي نصف الطور الثاني اللامركزي في الـ VNs.

5–2. الطور الثاني اللامركزي:

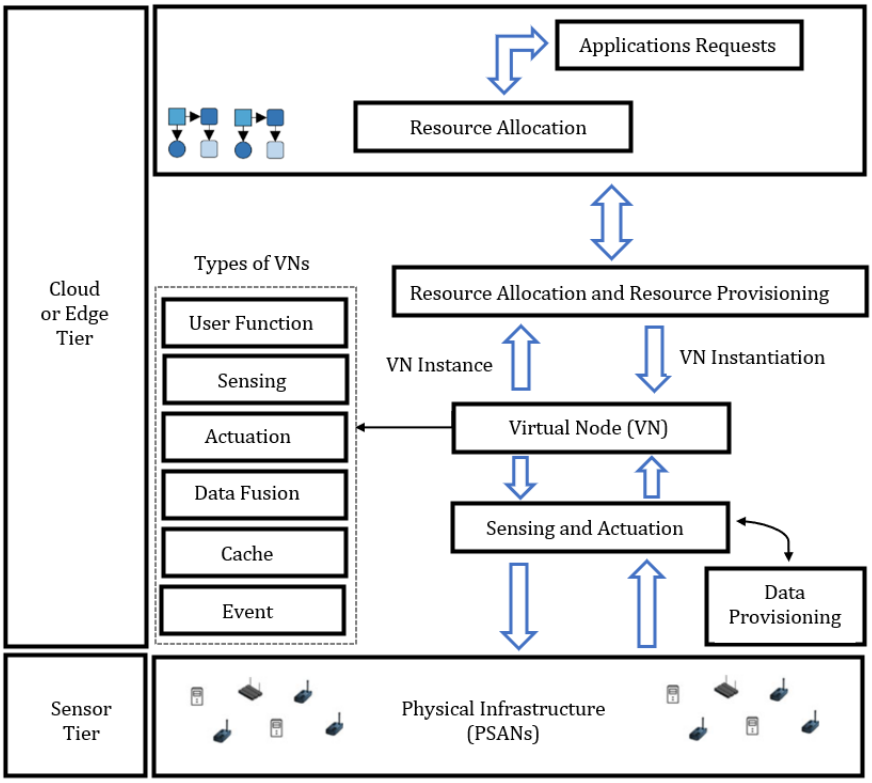
يبدأ الطور الثاني كما في الخوارزمية الموضحة في الشكل (4)، باستحضار خدمة توفير البيانات (السطر 2) حيث تزود هذه الخدمة طلباً واحداً بالبيانات في كل مرة. يُشَكَّلُ في السطر (3) رتل الانتظار الخاص بالـ VN، الذي يبدأ فارغاً. تتكرر القاعدة f_{dec} بشكل دوري بين السطور (4) و (14). تُعيد الـ VN القاعدة من أجل كل طلب urq واصلٍ من الـ AEP (السطر 5)، وفي السطر (6) تضاف الـ urq إلى الرتل وتُرَتَّب حسب أولويتها. إن الأولية في مقترحنا أولوية ساكنة Static وهي عبارة عن خانة ثنائية (0 أو 1) تمثل ما إذا كان الطلب تابعاً لتطبيق ضمن شريحة سعريةٍ خاصة بالاستجابة الزمنية له من قبل النظام. في حال تساوت الأولويات يأخذ الرتل نظام الداخل أولاً يخرج أولاً FIFO (First in First out). يُحذَفُ الـ urq من الرتل بعد إزالة قيد الأسبقية، أي أن الطلب أصبح حراً لتحصيل بياناته. فإذا انتهت خدمة توفير البيانات (السطر 7)، ترسل الـ VN بيانات الخرج إلى كل خلفاء الطلب urq ليُصار إلى إزالة قيد الأسبقية عنهم، وإذا لم يوجدوا، يرسل إلى الـ AEP التي قدِمَ منها. في السطر (9) توضع خدمة توفير البيانات بحالة خمول. تفحص الـ VN حالة الخدمة (السطر10) فإذا كانت خاملةً والرتل لم ينتهِ بعد، تجلب الـ urq التالي من الرتل والمحرر من قيد الأسبقية (سطر 12). بعدها ستقرر الـ VN إجراء تحديثٍ للبيانات من عدمه. وذلك وفق قاعدةٍ معطاة (parameter rule_fdec). استخدمنا في بحثنا قاعدة تجنب الفائدة السلبية Avoid Negative Utility Rule (ANUR) التي تنص على تحديث البيانات إذا كانت قيمة حداثة البيانات التي توفرها الـ VN أكبر من عتبة الحداثة الموجودة في urq. في السطر (14) يُحدد وقت بدء تحديث البيانات بالوقت الحالي الموجود في الـ VN، ثم تبدأ خدمة تزويد البيانات بمعالجة الـ urq المحتاج لتحديث في السطر (15).

```
Input: buff_tm, SVN, rule_fdec
Output: decision yj

1.  def fdec(rule_fdec): // 2nd PHASE
2.    dp = data_provisioning service();
3.    queue ← {Ø};
4.    while True:
5.    for each urq in new_urqs_arriving():
6.      add_to_queue_prty_fst(urq);
7.    if dp.completed():
8.      transmit_data_to_suc_VNs(dp.cmp1_urq);
9.      dp.is_idle = True;
10.   if (dp.is_idle) and (queue.len > 0):
11.     //Begin deciding yj
12.     pri_urq = queue.get_top();
13.     y ← decides _for_dupd(urq, rule_fdec);
14.     pri_urq.SUT = now();
15.     dp.provide data(urq, y);
```

الشكل (4): الطور الثاني في الخوارزمية المقترحة.

أما الجزء الثاني للتعديل المقترح فيتعلق بنظام المحاكاة الافتراضية في Zeus. يُعرِّف هذا النظام الـ VN على أنها برنامجٌ Program يُنفِّذ مجموعة من تقنيات دمج المعلومات Information Fusion. في هذا النموذج يُبنى سلوك الـ VN (VN behavior) من الصفر، وينقل في الشبكة عند كل عملية توفير لها، ولا بدَّ لتوفير أسرع يخدُم الآلية التفاعلية من تعديل هذا النموذج. قسم الباحثون العقد الافتراضية لعدة أنواع، وتختلف هذه الأنواع عن بعضها باختلاف نوع البيانات DT المقدم كما في الشكل (5) [23]. اعتمد هذا النهج في نموذج المحاكاة الافتراضية خفيف الحمل LW-CoEdge الذي اعتمدناه لإجراء التجارب. نظراً لأن مكوناتِ هذا النموذج خدماتٌ مصغرة مُعبَّأةٌ في صورٍ Images خفيفة الوزن، فإن كل عقدة حافية أو سحابية ECN تملك مسبقاً كل الصور اللازمة لتشغيل أي VN من نوع بيانات محدد. بمعنىً آخر إن كل العقد تملك جميع صور الـ VNs الممثلة لجميع أنواع البيانات الموَصَّفة في Catalogue النظام. بهذه الطريقة يُضمَن الحفاظ على عدم تكبيد الشبكة أيَّ حملٍ إضافي يستنزف عرض الحزمة ناتج عن نقل الصور بين الـ ECNs. في Zeus وكون الـ VNs الممثلة لمختلف أنواع البيانات التي يقدمها النظام منشأة سابقاً، لم يكن لتوفيرها هذا الإشكال. أما في نموذج تفاعلي ينشئ الـ VNs عند الطلب، يصبح توفيرها بأقصى سرعة أولوية عظمى. من المهم الإشارة بمكان إلى أن عنصر الـ Catalogue في النظام LW-CoEdge يحوي جميع أنواع البيانات ومجاري العمل Workflows الخاصة بتطبيقاتها بشكل مسبق. بمعنى آخر، كل طلبات التطبيقات الواردة للـ CoS تكون وفق القواعد المضبوطة سابقاً من قبل النظام.



الشكل (5): نموذج المحاكاة الافتراضية المقترح لـ CoS.

6. منهجية تقييم البحث والنتائج

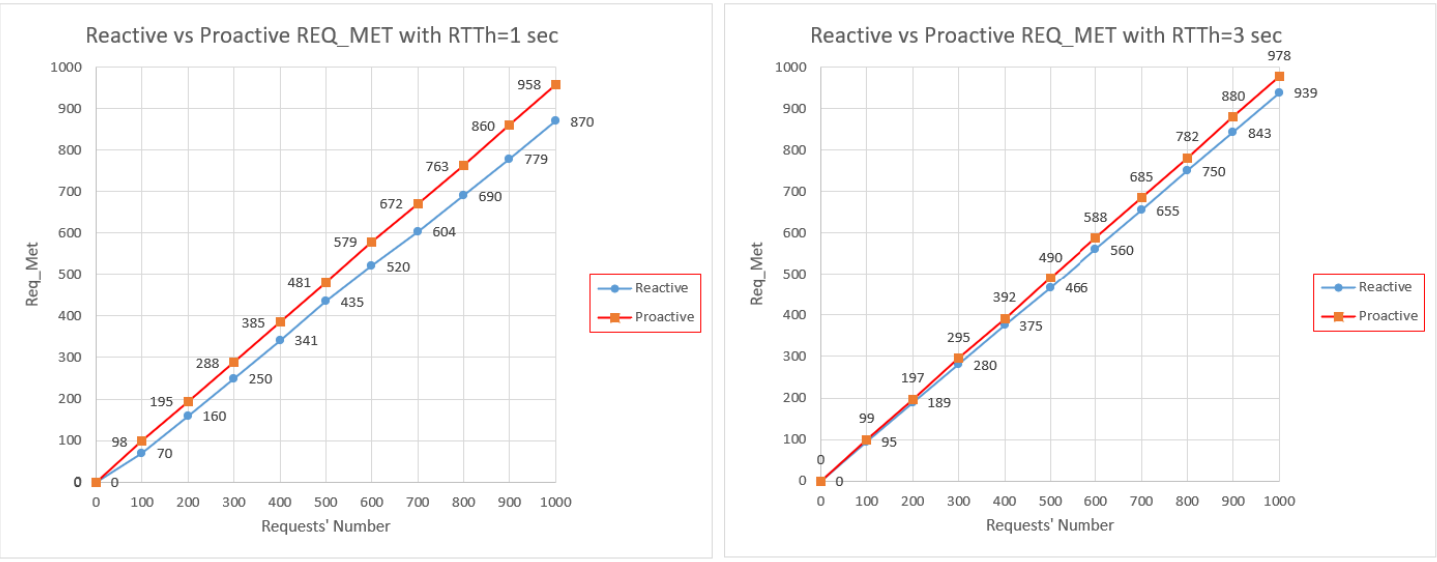
سنستخدم لتقييم البحث منهجية GQM اختصاراً لهدف، سؤال، مقياس Goal, Question, Metric. يحدد الهدف غاية البحث بالنسبة لمن ومِن وجهة نظر مَنْ، بحيث يوصَّف من خلال عدة أسئلة لتعريف الموضوع المراد قياسه. أما المقاييس فتمثل المعلومات الكمية التي يجب جمعها للإجابة عن الأسئلة. يتمثل هدفنا G بتحليل النهج التفاعلي المقترح بغرض تقييم خوارزمية Zeus المعدلة لتخصيص الموارد، نسبةً لأدائها في تلبية طلبات التطبيقات، مقارنةً بالنهج الاستباقي. ونحدد لهذا الهدف ثلاثة أسئلة. السؤال الأول Q1: هل يساعد النهج التفاعلي المقترح على تلبية طلبات التطبيقات الواردة للنظام مقارنة بالنهج الاستباقي؟ السؤال الثاني Q2: هل ساعد النهج التفاعلي في تقليل حالة التوفير الفائض Over Provisioning في النهج الاستباقي؟ السؤال الثالث Q3: هل تساعد الأولوّية المضافة لطلبات التطبيقات ذات الشريحة السعرية الأعلى بتلبية متطلباتها من ناحية العتبة الزمنية للاستجابة RTTh؟

للإجابة عن السؤال الأول سنُصنِّف طلبات التطبيقات لصنفين: ملبىً وغير ملبى. يعتبر النظام أن الطلب قد لُبِّيَ إذا لُبيَت متطلبات نوع البيانات والاستجابة الزمنية للتطبيق. نرمز لعدد الطلبات الملبية بالمقياس REQ_MET. وللإجابة عن السؤال الثاني، فإنَّ أبرز مساوئ النظام الاستباقي بقاء قسم من الموارد (في حالتنا هنا هي الـ VN) دون إسناد. ولقياس ذلك سنعرف المقياس CPU_RESERVED الممثل لموارد المعالجة المحجوزة مقارنة مع النهج التفاعلي. أما في السؤال الثالث، سنقارن عدد الطلبات من التطبيقات ذات الشريحة الأعلى التي لم تلبَّ بوجود الأولوية

Priority ثم بدونها، ونعرف لذلك المقياس REQ_NOT_MET. مع ملاحظة تأثير ذلك على عدد الطلبات الملباة الكلي في الحالتين.

6-2. التجربة الأولى:

سنحاول في هذه التجربة الإجابة عن السؤالين Q1 و Q2 بمقارنة أداء النهجين الاستباقي والتفاعلي. يتضمن السيناريو 4 عقد حافية، مع توليد النظام لـ 1000 طلب تطبيق لنوعين من البيانات DT1 و DT2 (500 طلب لكل نوع). وسيقوم بإنشاء ثماني عقد افتراضية VNs في الحالة الاستباقية (اثنتان في كل عقدة حافية)، في حين ستُنشَأ عند الطلب في الحالة التفاعلية. تتراوح عتبات حداثة البيانات بين 1 sec و 5 sec. للإجابة على السؤال Q1، سنعيد التجربة لقيمتين من العتبة الزمنية للاستجابة RTTh الأولى ثانية واحدة والأخرى ثلاث ثواني. فكانت النتائج كما في الشكلين (6-a,6-b).



(a) (b)

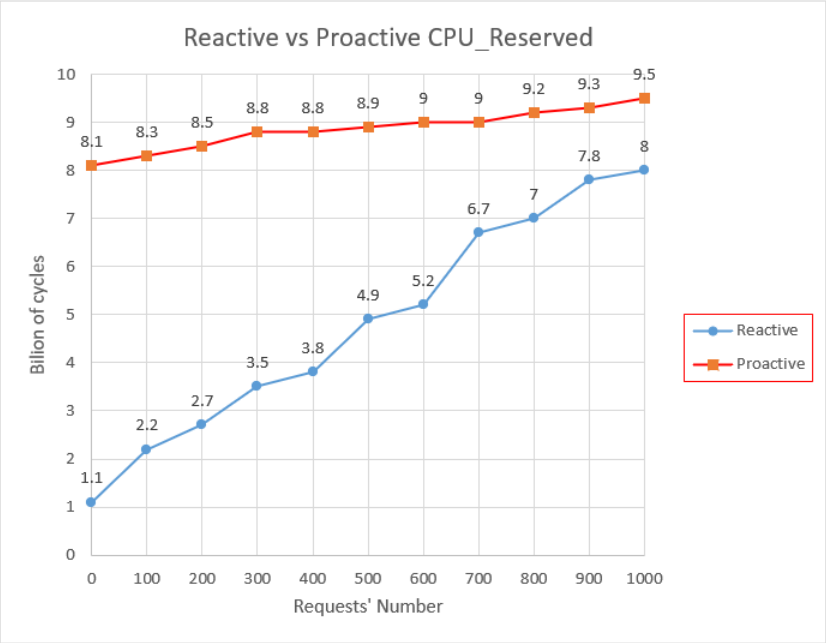
الشكل (6): مقارنة بين النهجين الاستباقي والتفاعلي بعتبة زمنية للاستجابة 1 ثانية (a)، و3 ثانية (b).

نلاحظ أن أداء النهج الاستباقي كان الأفضل في الحالتين، مع تحسن ملحوظ للنهج التفاعلي عند ارتفاع قيمة العتبة الزمنية للاستجابة إلى 3 ثواني. يعود ذلك إلى الضياع الزمني الناجم عن الفترة اللازمة لإنشاء العقد الافتراضية في الحالة التفاعلية؛ أي أن النظام لبَّى الطلبات في هذه الحالة بزمن تجاوز الثانية الواحدة، فاعتبرت

غير ملباة وفق المعيار الذي وضعناه لإجراء هذا القياس. وعليه فإن تطبيقات صارمة الاستجابة وبحاجة لزمن حقيقي تحتاج لنهج استباقي لتلبية متطلباتها بالكفاءة المناسبة.

إن هذا الأداء الأفضل في الحالة الاستباقية ليس مجانيَّ الكلفة. فالنظام قد حجز 8 عقد افتراضية مسبقاً. ولمقارنة الفرق بين استهلاك الموارد بين الحالتين كإجابة على السؤال Q2، سننظر إلى إجمالي عدد الدورات Cycles المستهلكة في الثانية، التي تمثل استهلاك موارد وحدة المعالجة المركزية CPU، كما في الشكل (5). مع الإشارة إلى أن هذا الاستهلاك كان بعد تنصيب جميع عناصر النظام ما عدا الـ VNs، وبعتبة RTTh = 1.

إن المساومة بين النهجين تعود بشكلٍ أساسي لنوع التطبيق المراد تخديمه. فتطبيقاتٌ صارمةُ الاستجابةِ مثل تطبيقات الواقع المعزز، أو الصحة الرقمية ستجبرنا على القبول بتجهيزِ موارد أكثرَ استباقياً، لتجنب الإضرار بمتطلبات جودة الخدمة QoS للتطبيق، في حين أن تطبيقاتٍ أقلَّ حراجةٍ زمنياً مثل تطبيقات الاستعلام عن الطقس، أو قياس مؤشرات البيئة يمكن أن يلبيها النهج التفاعلي بكفاءة وحدٍ أدنى من استهلاك الموارد.



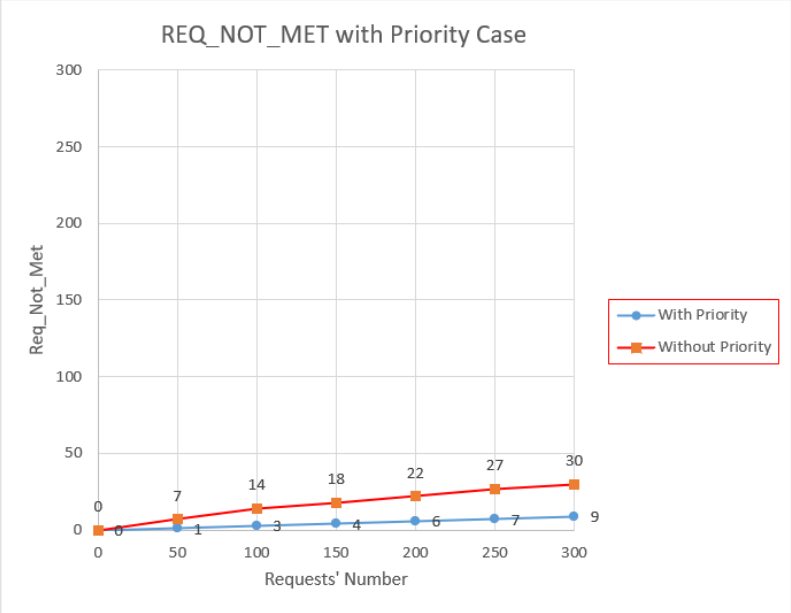
الشكل 5: مقارنة بين النهجين الاستباقي والتفاعلي من ناحية موارد الـ CPU المحجوزة.

6-2. التجربة الثانية:

لتعظيم مكاسب النظام ذو النهج التفاعلي اقترحنا إضافة أولوية لطلبات التطبيقات ذات الشرائح السعرية العليا. ففي نظام يدعم عدة أنواع من التطبيقات، لا بدَّ من تمييز ذوات الأهمية القصوى، كتطبيقات كشف الحرائق، أو تطبيقات تعنى بمراقبة عوارض صحية طارئة كالأزمات القلبية.

لدراسة أثر هذه الأولوية، سنفرض أن 300 طلب من مجمل 1000، تنتمي للنوع DT 1، وتمثل طلبات ذات أولوية مع عتبة استجابة زمنية RTTh = 1 sec. بإضافة الخانة priority إلى طلب التطبيق قبل توجيهه من الـ AppMgr في نموذج العمل LW-CoEdge إلى الـ VN، بحيث تقوم بتلبيته أولاً عند وروده دوناً عن غيره. فكانت النتيجة كما في الشكل (6). نلاحظ أن عدداً قليلاً جداً لم يلبَّ عند وجود الأولوية مقارنة مع عدمها، على الرغم من العتبة الزمنية الصارمة. إلا أن ذلك كان على حساب الطلبات الملباة من النوع DT2 حيث انخفض عددها من 613 قبل تطبيق الأولوية إلى 526 بعدها.

يمكن القبول بذلك إذا كانت الطلبات ذات النوع DT 2 ذات عتبة استجابة أعلى قيمةً، وبالتالي يستطيع النظام تلبيتها لاحقاً بعد تخديم ذوات الأولوية. إذاً لا بدَّ لنظام تفاعلي يحقق أعظم المكاسب من دعم تمييز بين التطبيقات على شرائح مختلفة.



الشكل 6: مقارنة بين عدد طلبات التطبيقات غير الملباة من الشرائح الأعلى بوجود الأولوية وعدمها.

7. الاستنتاجات والتوصيات

1- لا بدَّ لتمكين حل تفاعلي أمثل لتوفير الموارد من دمج الحل الرياضي لمسألة الأمثلة، مع نهج محاكاةٍ افتراضية خفيف الحِمل (كنموذج LW-CoEdge)، يسهل نشره وتوزيعه في عقد الحواف، يجعل عِبْءَ الشبكة (Overhead) أصغر ما يمكن. إن القضية الأهم في النهج التفاعلي هي توفير العقد الافتراضية VNs بأسرع وقت ممكن استجابةً لقرار تخصيصها لتلبية طلب ما، وعليه تتَّضِحُ أهمية تمثيل العقد الافتراضية كحاويات Containers، مما يقلل زمن تشغيلها للحد الأدنى، ويُتَجَنَّبُ استنزاف عرض الحزمة في الشبكة، للفرق الواضح الجَلَي بين كلفة التخزين (الرخيصة الثمن) وكلفة النقل (أغلا ثمناً). كما يفضَّل لتعظيم مدخلات نظام سحابة المستشعرات تمييز التطبيقات وفق شرائح سعرية. ذلك أن استجابة النظام لتطبيقات صارمة الاستجابة الزمنية قد تتغير حسب عدد الطلبات الواردة في نافذة القرار. فعليه ولأجل تخديم جميع التطبيقات بمختلف الأنواع، يجب تمكين الأولويات، لتحقيق المكسب لكلٍّ من مزود الخدمة، والمستخدم.

2- يستحسن أن تكون النافذة الزمنية التي تُحلَّلُ فيها الطلبات الواردة في خوارزمية Zeus أكبر من زمن توفير أضخم صورةٍ لعقدة افتراضية، والذي يتعلق بتعقيد نوع البيانات المطلوب من التطبيق (كاعتماده مثلاً على نتائج سلاسل زمنية)، وإلا سيُصار إلى انتظار الطلب بعد اتخاذ قرار تخصيصه بـ VN معينة، لانتهاء المدة الزمنية اللازمة لتوفيرها. بمعنىً آخر يجب أن يُضْمَن انتهاء توفير الـ VN عند نهاية قرارِ تخصيصِ طلبٍ لها.

3- إن الحالة المثلى لتمكين نظامٍ يخدم طيفاً واسعاً من التطبيقات بمختلف عتبات الاستجابة الزمنية، وبأفضل استهلاك ممكن للموارد، هي دعم نمط هجين يتنقل بين نهجي التوفير الاستباقي والتفاعلي وفق الطلب، فتطبيقاتٌ صارمةُ الاستجابةِ بعتبات استجابة أقل من 100 ms مثل تطبيقات الواقع المعزز أو معالجة البيانات الحية والمباشرة لحساساتِ سياراتٍ ذاتية القيادة إلخ... ستجبرنا على القبول بتجهيزِ مواردَ أكثرَ استباقياً لتجنب الإضرار بمتطلبات جودة الخدمة QoS للتطبيق والتأثير الفادح على تجربة المستخدم النهائي، في حين أن تطبيقاتٍ أقلَّ حراجةٍ زمنياً مثل تطبيقات الاستعلام عن الطقس، أو تطبيقات التحكم بالتكييف، أو قياس مؤشرات البيئة كقياس درجة حرارة المحيطات إلخ... بعتباتٍ زمنية تتراوح بين 3 sec إلى 5 sec يمكن أن يلبيها النهج التفاعلي بكفاءة وحدٍ أدنى من استهلاك الموارد.

4- من الآفاق المستقبلية للبحث بناء نموذج احتمالي يُبَدِّلُ بين النمطين حسْبَ تكرارية وصول طلبات التطبيقات، وحسب عتبات الاستجابة الزمنية لها. كما يمكن اعتبار طول النافذة الزمنية في Zeus متكيفاً لا ثابتاً، يَكبُر في حالة التطبيقات الأقل حراجة زمنياً، بحيث يُجمَّع عددٌ أكبر من الطلبات المشتركة التي تنفذ مرة واحدةً، لتقليل الحاجة لتشغيل المستشعرات، وإطالة دورة حياتها للحد الأقصى.

المراجع

[1] Aleksic, S. (2014, May). *Green ICT for sustainability: A holistic approach*. In 2014 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO) (pp. 426-431). IEEE.

[2] Rawat, P., Singh, K.D., Chaouchi, H. and Bonnin, J.M., 2014. *Wireless sensor networks: a survey on recent developments and potential synergies*. The Journal of supercomputing, Springer, Vol. 68, Issue 1, pp.1-48.

[3] Li, W., Santos, I., Delicato, F.C., Pires, P.F., Pirmez, L., Wei, W., Song, H., Zomaya, A. and Khan, S., 2017. *System modelling and performance evaluation of a three-tier Cloud of Things*. Future Generation Computer Systems, Elsevier Science, Vol. 70, pp.104-125.

[4] Santos, I. L., Alves, M. P., Delicato, F. C., Pires, P. F., Pirmez, L., Li, W. & Khan, S. U., 2018. *A System Architecture for Cloud of Sensors*. In 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech) (pp. 666-672).

[5] Islam, M., Hassan, M.M., Lee, G.W. and Huh, E.N., 2012. *A survey on virtualization of wireless sensor networks*. Sensors, Vol. 12, Issue 2, pp.2175-2207.

[6] Gonçalves, G.E., Endo, P.T., Cordeiro, T.D., Palhares, A.V.A., Sadok, D., Kelner, J., Melander, B. and Mangs, J., 2011. *Resource allocation in clouds: concepts, tools and research challenges*. XXIX SBRC-Gramado-RS.

[7] Gmach, D., Rolia, J., Cherkasova, L. and Kemper, A, 2009. *Resource pool management: Reactive versus proactive or let's be friends*. Computer Networks, Elsevier Science, Vol. 53, Issue 17, pp.2905-2922.

[8] Santos, I. L., Pirmez, L., Delicato, F. C., Khan, S. U., & Zomaya, A. Y., 2015. *Olympus: The cloud of sensors*. IEEE Cloud Computing, Vol. 2, Issue 2, pp. 48-56.

[9] Bouzeghoub, M., 2004, June. *A framework for analysis of data freshness*. In Proceedings of the 2004 international workshop on Information quality in information systems (pp. 59-67).

[10] de Farias, C.M., Pirmez, L., Delicato, F.C., Li, W., Zomaya, A.Y. and de Souza, J.N., 2013. *A scheduling algorithm for shared sensor and actuator networks*. In The International Conference on Information Networking 2013 (ICOIN), IEEE, (pp. 648-653).

[11] Farias, C.M.D., Li, W., Delicato, F.C., Pirmez, L., Zomaya, A.Y., Pires, P.F. and Souza, J.N.D., 2016. *A systematic review of shared sensor networks*. ACM Computing Surveys (CSUR), Vol. 48, Issue 4, pp.1-50.

[12] Delgado, C., Gállego, J.R., Canales, M., Ortin, J., Bousnina, S. and Cesana, M., 2015, December. *An optimization framework for resource allocation in virtual sensor networks*. In 2015 IEEE Global Communications Conference (GLOBECOM) (pp. 1-7).

[13] Delgado, C., Gállego, J.R., Canales, M., Ortín, J., Bousnina, S. and Cesana, M., 2016. *On optimal resource allocation in virtual sensor networks*. Ad Hoc Networks, Elsevier Science, Vol. 50, pp.23-40.

[14] Wang, N., Varghese, B., Matthaiou, M. and Nikolopoulos, D.S., 2017. *ENORM: A framework for edge node resource management*. IEEE transactions on services computing. Vol. 13, Issue 6, pp.1086-1099.

[15] Santos, I.L., Pirmez, L., Delicato, F.C., Oliveira, G.M., Farias, C.M., Khan, S.U. and Zomaya, A.Y., 2019. *Zeus: A resource allocation algorithm for the cloud of sensors*. Future Generation Computer Systems, Elsevier Science, Vol. 92, pp.564-581.

[16] Yu, R., Zhang, Y., Gjessing, S., Xia, W. and Yang, K., 2013. *Toward cloud-based vehicular networks with efficient resource management*. IEEE Network, Vol. 27, Issue 5, pp.48-55.

[17] Xu, J., Palanisamy, B., Ludwig, H. and Wang, Q., 2017, June. *Zenith: Utility-aware resource allocation for edge computing*. In 2017 IEEE international conference on edge computing (EDGE) (pp. 47-54). IEEE.

[18] Alves, M.P., Delicato, F.C., Santos, I.L. and Pires, P.F., 2020. *LW-CoEdge: a lightweight virtualization model and collaboration process for edge computing*. World Wide Web, Springer, Vol. 23, Issue 2, pp.1127-1175.

[19] FIWARE, Dec.2020 : https://www.fiware.org.

[20] LW-CoEdge, Dec.2020. https://github.com/mpitanga/lwcoedge.

[21] FIWARE IoT Agent Node.js Library, Dec.2020. https://github.com/telefonicaid/iotagent-node-lib.

[22] Oliveira, A.C., Calmon, T.S., Delicato, F.C., Pires, P.F. and Dos Santos, I.L., EMC IP Holding Co LLC, 2020. *Resource allocation and provisioning in a multi-tier edge-cloud virtualization environment*. U.S. Patent Application 16/034,432.

[23] dos Santos, I.L., Delicato, F.C., Pires, P.F., Alves, M.P., Oliveira, A. and Calmon, T.S., 2019. *Data-centric resource management in edge-cloud systems for the IoT*. Open Journal of Internet Of Things (OJIOT), Vol. 5, Issue 1, pp.29-46.