تطوير خوارزمية لموازنة حمل الخوادم في بيئة الشبكات المعرفة برمجياً باستخدام التعلم الآلي

ميساء الضاهر[1] د. ماهر عباس[2] د. محسن عبود[3]

الملخص

تعد الشبكات المعرَفة بالبرمجيات (SDN) إطاراً مرناً وفعالاً لإدارة موارد الشبكة وتوجيه حركة البيانات، مما يجعلها بيئة مثالية لتطبيق استراتيجيات موازنة الحمل بين الخوادم، ومع تطور تقنيات الذكاء الاصطناعي أصبح دمج التعلم الآلي مع SDN خطوة متقدمة نحو تحسين القرارات الشبكية بشكل ذاتي وتنبؤي. هذا الدمج يفتح المجال لشبكات أكثر ذكاءً وتكيفاً قادرة على مواجهة التغيرات اللحظية في الحمل الشبكي. وانطلاقاً من ذلك، تقترح هذه الدراسة خوارزمية جديدة لموازنة الحمل بين الخوادم في بيئة SDN، تعتمد على التنبؤ بزمن استجابة الخوادم باستخدام تقنيات التعلم الآلي بناءً على مؤشرات أداء الخوادم النشطة، مثل استخدام وحدة المعالجة المركزية، استخدام الذاكرة، عرض النطاق الترددي، وعدد الاتصالات النشطة بهدف اختيار الخادم الأمثل الذي يُتوقع أن يقدم أقل زمن استجابة. تم تقييم أداء الخوارزمية المقترحة ضمن بيئة ديناميكية وباختبارات متنوعة ومقارنتها مع خوارزميات Traditional Least Response، Least Bandwidth، Round Robin، وخوارزمية Naive Bayes، ثم تم تحليل النتائج إحصائياً، وقد أظهرت النتائج تفوق الخوارزمية المقترحة على جميع الخوارزميات المُقارن معها.

الكلمات المفتاحية: الشبكات المعرَفة برمجياً، موازنة الحمل، أداء الخوارزميات، التعلم الآلي.

1 ميساء الضاهر طالبة دكتوراه في كلية الهندسة المعلوماتية، قسم هندسة الشبكات والنظم الحاسوبية، جامعة حمص.

2 د. ماهر عباس أستاذ مساعد في كلية الهندسة المعلوماتية، قسم هندسة الشبكات والنظم الحاسوبية، جامعة حمص.

3 د. محسن عبود مدرس في كلية الهندسة المعلوماتية، قسم هندسة الشبكات والنظم الحاسوبية، جامعة حمص.

Developing an Algorithm for Server Load Balancing in an SDN Environment Using Machine Learning

Mayssaa Aldaher1, Dr. Maher Abbas2, Dr. Mohssen Abboud3
[1] PhD Student, Department of Network Engineering and Computer Systems, Homs University, Syria (E-mail: maldaher@homs-univ.edu.sy).
[2] Assistant Professor, Department of Network Engineering and Computer Systems, Homs University, Syria (E-mail: mabbaas@homs-univ.edu.sy)
[3]Lecturer, Department of Network Engineering and Computer Systems, Homs University, Syria (E-mail: mabboud@homs-univ.edu.sy).

Abstract

Software-Defined Networking (SDN) is a flexible and efficient framework for managing network resources and routing data traffic, making it an ideal environment for implementing server load balancing strategies. With the advancement of artificial intelligence (AI) technologies, integrating machine learning with SDN has become a significant step towards improving network decisions autonomously and predictively. This integration paves the way for smarter and more adaptive networks capable of handling real-time changes in network load. Therefore, this study proposes a novel server load balancing algorithm in an SDN environment. This algorithm relies on predicting server response times using machine learning techniques based on active server performance metrics, such as CPU utilization, memory utilization, bandwidth, and the number of active connections. The goal is to select the optimal server expected to offer the lowest response time. The performance of the proposed algorithm was evaluated in a dynamic environment using various tests and compared with Traditional Least Response, Least Bandwidth, Round Robin, and Naive Bayes algorithms. The results were then statistically analyzed, and the results showed that the proposed algorithm was superior to all the algorithms compared with it.

KEYWORDS: SDN, Load Balancing, Algorithms performance, Machine Learning.

1. المقدمة

مع تزايد استخدام الشبكات المعرفة بالبرمجيات (SDN) في الشبكات الحديثة، أصبحت إدارة موارد الشبكة بكفاءة أمرًا بالغ الأهمية. وتُعدّ موازنة أحمال الخوادم أحد أبرز التحديات في هذه البيئات، إذ يؤثر بشكل مباشر على أداء الشبكة وسرعة استجابتها، كما يضمن الاستخدام الأمثل للموارد المتاحة. ونظرًا للطبيعة الديناميكية لشبكات SDN، تواجه العديد من تقنيات موازنة الأحمال التقليدية تحديات متزايدة في مواكبة تطور الشبكات الحديثة، لا سيما من حيث التكيف مع ظروف الشبكة المتغيرة [1].

يتأثر أداء الخوادم بمؤشرات تشغيلية متنوعة، مثل استخدام وحدة المعالجة المركزية والذاكرة، واستهلاك النطاق الترددي، وعدد الاتصالات النشطة. ورغم شيوع مراقبة هذه المؤشرات، إلا أن الطرق الحالية عادةً ما تتعامل معها بشكل منفصل أو تعتمد على خوارزميات مبسطة لا تعكس بدقة حجم العمل الفعلي على الخوادم. علاوة على ذلك، غالبًا ما يتم إغفال التأثير المُجتمع لهذه المؤشرات على سرعة استجابة الخادم الذي يعتبر مؤشر رئيسي لجودة الخدمة. توفر تقنيات التعلم الآلي منهجاً عملياً لنمذجة هذه التفاعلات المعقدة. مع ذلك، فإن الدراسات التي تُطبّق هذه التقنيات على موازنة أحمال الخوادم في بيئات الشبكات المعرفة بالبرمجيات (SDN) محدودة نسبيًا.

حثت الدراسات السابقة على التكامل بين الشبكات المعرفة بالبرمجيات والتعلم الآلي، حيث أبرزت أن هذا التقارب يمثل اتجاهاً تقدمياً في تحسين شبكات مراكز البيانات (DCNs) [2]. يتيح هذا التكامل بين المجالين الاستفادة من القدرات التنبؤية لنماذج التعلم الآلي إلى جانب التحكم المركزي الذي توفره SDN، مما يعد بتأثير تحويلي على قدرة الشبكة على التكيف وإدارة الموارد [3]. كما تؤكد هذه الدراسات أن هذا المجال لا يزال ناشئًا ويتطلب مزيدًا من الاستكشاف الشامل، فهو يحمل الإمكانات ليس فقط لرفع الكفاءة التشغيلية من خلال موازنة الحمل الذكية المتوقعة ولكن أيضًا لإحداث ثورة في نماذج الإدارة التي تحكم أنظمة مراكز البيانات على مستوى العالم [2].

2. هدف البحث

انطلاقاً من توصيات الدراسات السابقة بتقاطع التعلم الآلي وشبكات SDN في سياق موازنة الأحمال من أجل التحرك نحو أنظمة تكيفية مدفوعة بالذكاء تتقدم عدة خطوات على التدابير التقليدية التفاعلية، وبما أنَ معظم الأبحاث التي دمجت التعلم الالي في SDN ركزت على موازنة حمل المسارات، بينما لاتزال موازنة حمل الخوادم باستخدام التعلم الآلي أقل استكشافاً رغم أهميتها في تعزيز كفاءة مراكز البيانات، ومن هذا المنطلق اخترنا التركيز على موازنة حمل الخوادم نظراً لندرة الأبحاث في هذا المجال، وهذا ما يجعل دمج التعلم الآلي فيها خطوة تستحق التعمق والدراسة وفرصة بحثية واعدة لتقديم حلول ذكية قابلة للتطبيق في بيئات الشبكات الحديثة. بناء على ذلك سيكون هدف البحث اقتراح خوارزمية جديدة للتنبؤ بزمن استجابة الخوادم في بيئات الشبكات المعرفة بالبرمجيات اعتماداً على مجموعة من المؤشرات التشغيلية النشطة للخادم. يعتمد النموذج المقترح على خوارزمية الغابة العشوائية (Random Forest) لتحليل العلاقة بين مجموعة من العوامل الأساسية، مثل استخدام وحدة المعالجة المركزية، واستهلاك الذاكرة، وعرض النطاق الترددي، وعدد الاتصالات النشطة، بهدف بناء نموذج تنبؤي قادر على تقدير زمن الاستجابة بدقة أعلى، فالهدف من هذا النهج هو تحسين عملية اتخاذ القرار في موازنة الأحمال داخل بيئات SDN الديناميكية.

3. مواد وطرق البحث

سنستخدم في التجارب محاكي Mininet هو برنامج محاكاة يستند إلى Linux من أجل النمذجة السريعة للشبكات المعرفة برمجياً [4] يستخدم على نطاق واسع في أبحاث SDN ومنها الدراسة [5] . أيضاً سنستخدم Siege وهي أداة اختبار لحمل HTTP يمكن استخدامها لإرسال طلبات مئات الزبائن المتزامنين إلى مخدم الويب في وقت واحد للتحقق من كيفية أدائه تحت الضغط ومدى التوفر الذي يمكن أن يوفره الخادم، توفر بعد اكتمال الاختبار معلومات مفيدة مثل التوفر وعدد الطلبات الناجحة وعدد الطلبات الفاشلة والمزيد من المعلومات [6]. أيضاً سيتم استخدام Psutil لقياس إحصائيات الخادم ومراقبة أدائه خلال تنفيذ التجارب، وهي مكتبة متعددة المنصات تُستخدم لاسترجاع المعلومات حول العمليات الجارية واستخدام النظام [7]. كما سيتم استخدام أداة D-ITG [8] لتوليد traffic مغاير

عن http حيث سيتم تشغيلها بالتوازي مع أداة Siege التي تولد http traffic الذي سيُطَبق موازنة الحمل عليه، لمحاكاة بيئة شبكة واقعية، مما يسمح بتقييم أداء موازنة حمل http ضمن سياق عملي أقرب إلى الواقع.

4. الدراسات السابقة

حظيت مسألة موازنة حمل الخوادم باهتمام بحثي واسع في مجال الشبكات الحاسوبية. ومع ظهور الشبكات المعرَفة بالبرمجيات (SDN) أُعيدت صياغة هذه المسألة ضمن إطار أكثر مرونة، مستفيداً من الفصل بين مستوى التحكم ومستوى البيانات وما يتيحه ذلك من رؤية شاملة لحالة الشبكة وإدارتها مركزياً. ركزت بعض الأبحاث في هذا المجال على استخدام خوارزميات موازنة حمل تقليدية مثل Random [9]، Round Robin و Weighted Round Robin [10] بعد تكييفها للعمل ضمن بيئة SDN ورغم بساطة هذه الخوارزميات وسهولة تطبيقها إلا أنها غالباً ما تهمل الحالة الفعلية لموارد الخوادم أو ظروف الشبكة الديناميكية مما قد يؤدي إلى توزيع غير عادل للحمل. وهذا ما جعل بعض الباحثين يستخدم خوارزميات تعتمد على تحليل مؤشرات الأداء الفردية للخوادم لموازنة الحمل بينها. حيث اعتُبر في [11] أن عدد الاتصالات النشطة هو أحد العوامل الأساسية التي تُؤثر على أداء الخادم بينما في [12] تم الاعتماد على عرض النطاق الترددي للخوادم. في حين اعتمدت [13] على أوقات استجابة الخوادم لتوزيع الحمل وهنا لابدَ من الإشارة أنه تم قياس وقت الاستجابة بطريقة تقليدية بالاعتماد على إرسال وحدة التحكم لرزم packet out بفاصل زمني معين وتسجيل وقت الإرسال ثم انتظار الرد من الخادم وتسجيل وقت الاستلام. أما في [14] استُخدم خوارزمية Server Load Balancing والتي يتم فيها اختيار الخادم الذي لديه أقل قيمة لوحدة المعالجة المركزية. في [15] تم توزيع الحمل بين الخوادم في بيئة SDN باستخدام خوارزمية Least Packet Load حيث يتم توجيه الطلب إلى الخادم الذي يتلقى أقل عدد من الرزم. أحياناً الاعتماد على عامل واحد فقط قد لا يعكس الطبيعة المعقدة لأحمال الخادم خاصة تحت ظروف مرور متباينة. لتحسين دقة تقدير حمل الخادم قامت دراسات أخرى بدمج مؤشرين أو أكثر ففي [16] تم الاستناد إلى وقت استجابة الخادم ونوع محتوى الطلب لإجراء عملية الموازنة بين الخوادم. أما في [17] يتم اختيار الخادم الأفضل بناءً على إحصائيات وقت

الاستجابة وحجم المرور عبر منافذ المبدل. في [18] تم موازنة الحمل بين الخوادم في شبكة SDN باستخدام بارمترات استخدام المعالج، استخدام الذاكرة، استخدام I/O بالاعتماد على المنطق الضبابي. في [19] تم موازنة الحمل لكن في بيئة SDN هجينة بالاعتماد على مؤشرات CPU load، I/O read، I/O write، link upload، link download. من خلال النظر في عدة عوامل، توفر الأساليب السابقة صورة أكثر تمثيلاً لحمل الخادم. ومع ذلك، تعتمد هذه الطرق على قواعد ثابتة أو أوزان محددة مسبقاً، دون الاستفادة من تقنيات التعلم الآلي لفهم العلاقة الديناميكية بين هذه المؤشرات وتأثيرها الفعلي على أداء الخادم. في المقابل بدأت الأبحاث الحديثة بالاستفادة من تقنيات التعلم الآلي لتحسين موازنة الحمل في شبكات SDN. ففي [20] تم استخدام نموذج التعلم الآلي logistic regression للتنبؤ بنوع الطلب الأمثل الذي يمكن للخادم التعامل معه عند إعطائه الميزات، تم تدريبه على مجموعة بيانات تحوي 200 نقطة، كل نقطة بيانات عبارة عن مجموعة تحتوي على وقت الرحلة (trip time) واستخدام الذاكرة واستخدام المعالج والعلامة التي تكون إما 0 أو 1 المقابلة لنوع الطلب Media أو Text، لم يتم ذكر آلية توليد dataset. أما في [21] تم تنفيذ موازنة الحمل في SDN باستخدام الانحدار الخطي. تحتوي مجموعة البيانات الخاصة بنمذجة التابع الخطي على ثلاثين عينة، كل عينة تحتوي على معلومات وحدة المعالجة المركزية وذاكرة RAM وعدد الطلبات. يتم حساب الانحدار الخطي وفقاً لوحدة المعالجة المركزية وذاكرة الوصول العشوائي الخاصة بالخادم والنتيجة هي التنبؤ بعدد الطلبات من كل خادم. لم يتم تحديد آلية الحصول على الثلاثين عينة بدقة. في [22] تم تطبيق موازنة الحمل على شبكة معرفة بالبرمجيات باستخدام خوارزمية Naïve Bayes بناءً على البيانات الواردة من الخوادم (استخدام المعالج، استخدام الذاكرة، استخدام القرص) بعد ذلك يتم إجراء العمليات الحسابية عليها لتحويلها إلى شكل لغوي بناءً على مجموعة بيانات محددة مسبقاً مؤلفة من 27 حالة للحصول على حالة الخادم، وبعد الحصول على حالة الخادم يتم توزيع المرور بناءً على أقل استخدام للموارد من الخادم. بالرغم من استفادة الأبحاث السابقة من تقنيات التعلم الآلي لتحسين موازنة الحمل في شبكات SDN، مع ذلك لم تركز معظم هذه الدراسات على تأثير مؤشرات الموارد المختلفة على وقت استجابة الخادم باعتباره عامل أساسي لجودة الخدمة، أو اقتصرت على مجموعة محدودة

من المؤشرات. بناءً على ما سبق، يتضح أنّ هناك حاجة إلى نموذج تعلم آلي قادر على نمذجة العلاقات المعقدة للتأثير المشترك لمجموعة من مؤشرات موارد الخادم الأساسية للتنبؤ بوقت الاستجابة له بدقة عالية.

لابدَ من الإشارة إلى أنَ الدراسة رقم [23] تضمنت تحليلاً دقيقاً ومقارناً وشاملاً لأكثرية الجوانب التقنية للخوارزميات المستخدمة في مجال موازنة حمل الخوادم في شبكات SDN من حيث تصميم وآلية عمل كل خوارزمية، الطوبولوجيا المستخدمة، البارامترات المقاسة، محاكيات الشبكات وأدوات قياس الأداء المستخدمة، الخوارزميات المُقارن معها والافاق المستقبلية التي ذكرتها كل دراسة، وهذا ما وفر أساساً نظرياً قوياً وساهم في توجيه منهجية هذه الدراسة الحالية.

5. بنية الشبكات المعرفة برمجياً (SDN)

يتكون نموذج SDN من ثلاث طبقات، وهي طبقة البنية التحتية ويطلق عليها أيضاً طبقة البيانات، تتكون من أجهزة التبديل (مثل المبدلات، والموجهات، وما إلى ذلك). تتمثل وظائف أجهزة التبديل في أنها أولاً مسؤولة عن جمع حالة الشبكة وإرسالها إلى وحدات تحكم كما أنها مسؤولة عن معالجة الرزم استناداً إلى القواعد التي توفرها وحدة التحكم [24]. الطبقة الثانية هي طبقة التحكم والتي تمثل ذكاء الشبكة، تسمى العقد الموجودة في طبقة التحكم بوحدات التحكم. يسمح هذا الحل لوحدة التحكم بالحفاظ على رؤيا عامة للشبكة المتوضعة في طبقة البنية التحتية [25]. الطبقة الأخيرة هي طبقة التطبيق التي تحتوي على تطبيقات SDN مصممة لتلبية متطلبات المستخدم [24].

6. وحدة التحكم POX

يوجد حاليًا العديد من تطبيقات وحدات التحكم المتوفرة، وهي مفتوحة المصدر وتستند إلى لغات برمجة مختلفة. تم اختيار وحدة التحكم pox لإجراء التجارب في هذا البحث، والتي تم تطويرها بالكامل باستخدام Python، وبالتالي فهي تدعم Linux و Mac OS و Windows. هدفها الرئيسي هو تسريع عملية التطوير دون الحاجة إلى القلق بشأن مشاكل الأداء [26].

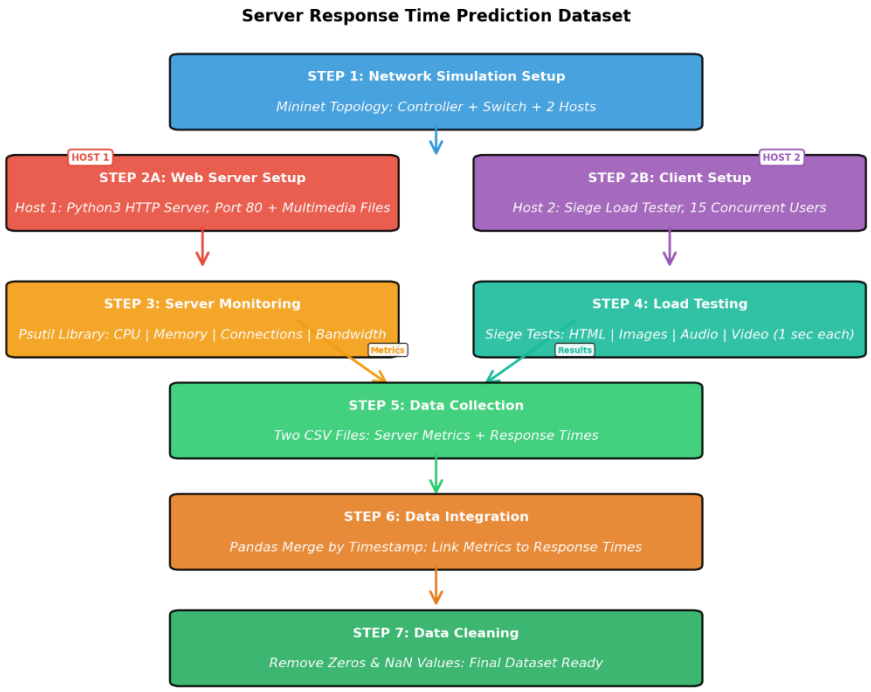
7. توليد مجموعة البيانات

تم توليد مجموعة بيانات تلبي احتياجات الدراسة للتنبؤ بوقت استجابة الخادم بناءً على المؤشرات النشطة للخادم. ولتوليد مجموعة البيانات تم القيام بالعديد من الخطوات:

- تم استخدام محاكي Mininet لتوليد طوبولوجيا شبكة وقد تم استخدام الطوبولوجيا الافتراضية المكونة من وحدة تحكم ومبدل ومضيفين.
- تم استخدام أحد المضيفين لتشغيل خادم ويب يستخدم بروتوكول HTTP على منفذ 80 باستخدام python3 على مجلد يحوي ملفات html وصور وفيديو وملفات صوتية بأحجام متنوعة حتى يقوم الخادم بتقديم جميع الملفات الموجودة في ذلك الدليل، كما تم كتابة وتشغيل برنامج بايثون على الخادم يقوم بحساب استخدام المعالج واستخدام الذاكرة وعدد الاتصالات وعرض النطاق الترددي للخادم كل ثانية باستخدام مكتبة Psutil وتوجيه هذه الإحصائيات وتخزينها بشكل مستمر في ملف CSV.
- أما المضيف الثاني في الطوبولوجيا تم استخدامه كعميل حيث تم تشغيل عدد من المستخدمين المتزامنين عليه بعدد 15 (العدد الافتراضي) يقومون بطلب الملفات التي يقوم الخادم بتقديمها وموجودة في الدليل كما هو موضح سابقاً، ومن أجل ذلك تم استخدام الأداة Siege لإرسال الطلبات واختبار حمل الخادم، تم تنفيذ الاختبارات بشكل متتابع وفوري واحد تلو الآخر مباشرة في وقت متقارب جداً، بهدف تمثيل حمل مستمر من جهة العملاء يعكس السلوك الواقعي للنظام ويرصد التأثيرات التراكمية على أداء الخادم. مدة كل اختبار ثانية واحدة لضمان تزامن موثوق مع مؤشرات أداء الخادم حيث توفر الأداة قياسات زمن استجابة على مستوى التجربة ككل، فالثانية الواحدة تحقق توازن منهجي بين دقة القياس وإمكانية التزامن مع مؤشرات الخادم فهي أقل وحدة زمنية تضمن تزامناً مقبولاً بين القياسات على العميل والسيرفر.
- تم توجيه نتائج اختبارات siege إلى ملف CSV. قسم من الاختبارات طُلب فيها الوصول إلى صفحة html، وفي قسم آخر طُلب الوصول إلى صورة وفي القسم الثالث من الاختبارات طُلب فيه الوصول إلى مقطع صوتي وفي القسم الأخير من الاختبارات طُلب فيه الوصول

إلى مقطع فيديو بهدف توفير محتوى متكامل للمستخدمين عبر HTTP حتى تكون الإحصائيات المخزنة في ملف CSV متنوعة.

- بعد ذلك تم دمج ملفي CSV السابقين الناتجين عن الاختبارات باستخدام مكتبة pandas بناءً على تطابق قيم حقل الطابع الزمني في ملف CSV جديد، ثم تم تنظيف ملف CSV الجديد عن طريق حذف القيم الصفرية وقيم nan وحذف الأعمدة غير الضرورية باستخدام مكتبة pandas. يوضح الشكل التالي خطوات توليد مجموعة البيانات بصورة موجزة:



الشكل (1) مخطط صندوقي يوضح خطوات توليد مجموعة البيانات.

وهذا جزء من ملف output.csv النهائي وقد احتوى هذا الملف على 882 سطر من البيانات:

timestamp	cpu_usage	memory_usage	connections	bandwidth_sent	bandwidth_recv	total_bandwidth	Resp Time
2025-01-24 10:55:39	10.6	25.1	41	9184	6714	15898	0.01
2025-01-24 10:55:41	21.5	25.1	57	11080	8254	19334	0.02
2025-01-24 10:55:43	6.5	25.1	62	3380	2430	5810	0.04
2025-01-24 10:55:45	5.1	25.1	69	4732	3402	8134	0.01
2025-01-24 10:55:55	10.9	25.4	110	7180	5280	12460	0.01
2025-01-24 10:55:57	13.8	25.6	131	8310	6030	14340	0.07
2025-01-24 10:55:59	15.8	25.2	145	10272	7554	17826	0.01
2025-01-24 10:56:01	8	25.2	157	8178	5898	14076	0.02
2025-01-24 10:56:03	6	25.2	160	2094	1524	3618	0.01
2025-01-24 10:56:13	9.1	25.3	223	10140	7290	17430	0.05
2025-01-24 10:56:15	4.5	25.2	234	8376	6096	14472	0.03
2025-01-24 10:56:17	12	25.3	245	6760	4860	11620	0.02
2025-01-24 10:56:19	19.3	25.5	268	15680	11524	27204	0.01
2025-01-24 10:56:21	10.7	25.3	278	7024	5470	12494	0.08

الشكل (2) جزء من مجموعة البيانات المولدة.

8. الخوارزمية المقترحة

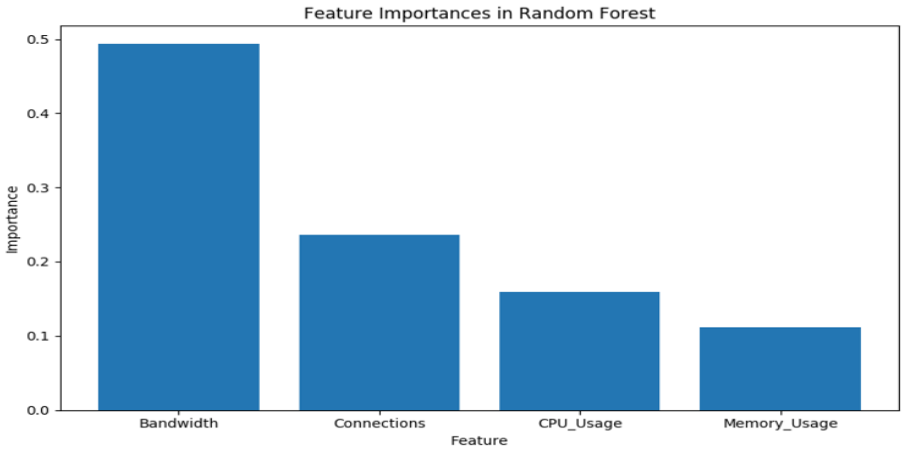
8-1 طريقة العمل

إنَ هدف البحث كما أُشير سابقاً هو تطوير خوارزمية تتنبأ بوقت الاستجابة بناءً على المؤشرات النشطة للخادم ولتحقيق هذا الهدف:

أولاً تم توليد مجموعة بيانات وهذا ما تم شرحه بالتفصيل في الفقرة السابقة، ثم تم إنشاء نموذج تعلم آلي باستخدام الغابة العشوائية (30 شجرة، عمق 10) وتدريبه على مجموعة البيانات المولدة. تم اعتماد تقسيم البيانات بنسبة 90:10 (90% للتدريب و10% للاختبار) استناداً إلى النتائج التجريبية الواردة في الدراسة [28]، والتي أثبتت أن هذا التقسيم حقق أعلى دقة أداء عند اختباره على مجموعة بيانات مكونة من 304 حالة؛ وهو ما يُعد نطاقاً قريباً من حجم البيانات المستخدم في هذا البحث المؤلف من 882 سجلّاً، فالهدف في هذا البحث هو تعظيم قدرة النموذج على التقاط الأنماط المعقدة والعلاقات غير الخطية بين موارد النظام المتمثلة في عرض النطاق الترددي، وعدد الاتصالات، واستخدام المعالج، واستخدام الذاكرة، من جهة، ووقت الاستجابة من جهة أخرى. وقد أثبتت النتائج العملية فيما بعد صحة هذا الاختيار، حيث حقق النموذج أداءً تنبؤياً دقيقاً ومتسقاً عند دمجه مع وحدة التحكم POX في بيئة التشغيل الفعلية، مما يُبرر اختيار هذه النسبة ويُؤكد ملاءمتها المثلى لهذا البحث.

تم حفظ النموذج ليتم استرجاعه في وحدة التحكم POX بحيث يتم تطبيقه على البيانات الجديدة للمحافظة على دقة واستقرار النتائج.

ولابدَ من الإشارة أنه تم تجربة عدة نماذج تحليلية، من بين جميع النماذج التي تم اختبارها، ظهر نموذج الغابة العشوائية كأفضل نموذج في تحقيق التوازن بين دقة التنبؤ وتفسير البيانات. حقق النموذج قيمة RMSE بلغت 0.0227 (5.27% من النطاق الكلي)، مما يدل على دقة تنبؤ عالية بفروق ضئيلة بين القيم المتنبأة والحقيقية. كما سجل معامل التحديد R^2 قيمة 0.7466، أي أنه يفسر حوالي 74.66% من التباين في البيانات، وهو أداء جيد في سياق تحليل البيانات المتعددة الأبعاد (استخدام المعالج، الذاكرة، عرض النطاق الترددي، وعدد الاتصالات). وبهدف الكشف عن العلاقات الكامنة وراء هذا الأداء، تم تقييم مساهمة كل متغير مستقل في ناتج النموذج عبر مقياس أهمية المتغيرات (Feature Importance) لفهم مدى تأثير كل ميزة على زمن الاستجابة في مجموعة البيانات المستخدمة في هذه الدراسة، وقد أظهرت النتائج أنَ تأثير موارد النظام الأربعة على وقت الاستجابة كان كما هو موضح في الشكل التالي:



الشكل (3) أهمية المتغيرات في نموذج الغابة العشوائية.

ثانياً تم تنفيذ برنامج بايثون على كل خادم يقوم بجمع وحساب إحصائيات الخادم (عرض النطاق الترددي، عدد الاتصالات، استخدام الذاكرة، استخدام المعالج) باستخدام أداة Psutil وإرسالها إلى وحدة التحكم POX بشكل مستمر كل خمس ثوانٍ عن طريق UDP Socket (وهي الطريقة المتبعة في جمع إحصائيات الخادم في الدراسة رقم [29]علماً أن الأكواد غير موجودة وتم تحقيقها). بعد ذلك،

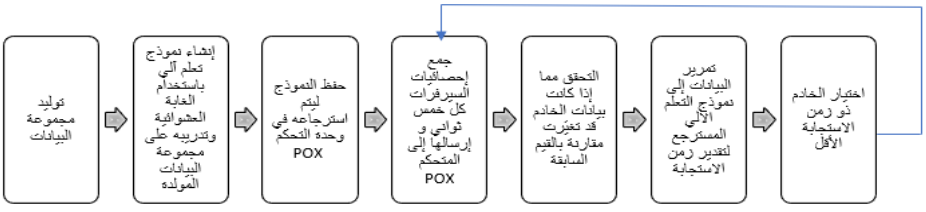
عندما تقوم وحدة التحكم باستقبال UDP Packet يتم معالجة هذه الرزمة كرزمة تحوي إحصائيات الخادم، لقد تم تخزين إحصائيات الخوادم ضمن dictionary، Keys لهذا dictionary هي عناوين IP الخاصة بالخوادم أما قيمة كل key هي قاموس داخلي يحتوي إحصائيات الخادم.

ثالثاً يتم التحقق مما إذا كانت بيانات الخادم قد تغيَرت مقارنة بالقيم السابقة وعند اكتشاف تغير في بيانات الخادم، يتم تحويلها إلى مصفوفة NumPy، بعد ذلك يتم تمرير هذه البيانات إلى نموذج التعلم الآلي الذي تم استيراده لتقدير زمن الاستجابة المتوقع لكل خادم، ثمّ يتم تخزين النتائج وربط كل IP خادم بزمن الاستجابة المتوقع. وهنا لابدّ من الإشارة إلى أنّ زمن التنبؤ لا يتجاوز بضعة مللي ثواني في الظروف العادية كون حجم النموذج نسبياً صغير (30 شجرة، عمق 10)، ويجري التنبؤ بشكل دوري كل 5 ثوانٍ عند استلام إحصائيات الخوادم، ويُخزَّن لاستخدامه لاحقاً. أما قرار اختيار الخادم الأمثل فيتم فوراً عند وصول رزمة من العميل باستخدام آخر التنبؤات المخزنة، حيث يبلغ متوسط زمن هذا القرار 0.08 مللي ثانية، فالنظام يفصل بين التنبؤ الدوري والقرار الفوري، التنبؤ يتم دورياً عند استلام الإحصائيات، والقرار يتم فوراً عند وصول الرزم باستخدام آخر التنبؤات المخزنة ولا يتأثر بزمن التنبؤ.

بناءً على ما سبق، عندما تأتي رزمة إلى موازن الحمل يتم اختيار الخادم ذو زمن الاستجابة الأقل وفقاً للتالي:

$$Best_server = Min(response_time(server_{(I)})) \quad where\ I=1, 2 \ldots n$$

يوضح الشكلان التاليان مراحل عمل الخوارزمية: الأول مخطط صندوقي للتسلسل العام، والثاني Pseudo-code للتفاصيل التنفيذية:



الشكل (4) مراحل تطوير الخوارزمية المقترحة.

```
ALGORITHM: توجيه الرزم إلى الخادم الأقل زمن استجابة
BEGIN
//==== المرحلة 1: التهيئة (مرة واحدة) ====
    dataset ← generateDataset()
    model ← train (RandomForest, dataset)
    save (model, "model.pkl")
    load (model, POX_Controller)

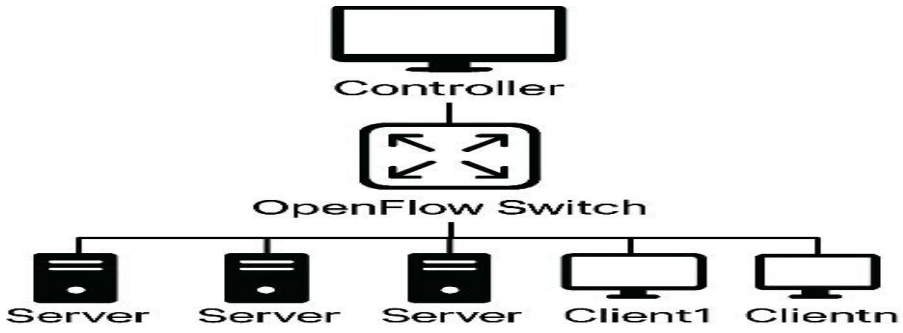
    predictions ← {} // Dictionary: {server_IP: زمن الاستجابة}
//==== المرحلة 2: تنبؤ دوري (كل 5 ثوانٍ) ====
    PROCEDURE PeriodicPrediction()
        WHILE systemRunning DO
            stats_dict ← POX.receiveStatsFromServers // الخوادم ترسل
        الإحصائيات، المتحكم يستلم هذه الإحصائيات
            FOR EACH (server_ip, server_stats) IN stats_dict.items() DO
                IF server_stats.changed() THEN
                    input ← convertToNumpy(server_stats)  // [[CPU, Memory,
Conn, BW]]
                    result ← model.predict(input) //  تقدير زمن الاستجابة
                    predictions[server_ip] ← result //  Dictionary  تحديث
القيمة بالـ
                END IF
```

```
        END FOR
      END WHILE
  END PROCEDURE
//==== المرحلة 3: قرار فوري (عند وصول رزمة من العميل) ====//
  PROCEDURE OnPacketArrival(packet)
      bestServer ← min(predictions, key=predictions.get) //  Dictionary
أقل قيمة بالـ
      forward(packet, bestServer) //          توجيه فوري
  END PROCEDURE
END
```

الشكل (5) Pseudo-code للخوارزمية المقترحة.

8-2 طوبولوجيا الاختبار والخوارزميات المُقارن معها

لاختبار أداء الخوارزمية المقترحة وفعاليتها تم استخدام طوبولوجيا SDN تم إنشاؤها باستخدام محاكي Mininet عن طريق برنامج نصي مكتوب بلغة python مؤلفة من وحدة تحكم POX تعمل كموازن للحمل ومبدل OpenFlow وثلاثة أجهزة مضيفة تعمل كخوادم ويب وعدد من الأجهزة المضيفة التي تعمل كعملاء كما في الشكل التالي:



الشكل (6) طوبولوجيا الاختبار.

المبررات لاختيار هذه الطوبولوجيا هي أنها الطوبولوجيا المستخدمة في أكثرية الدراسات السابقة [23] لأنه ما يهم هو المبدل المتصل مباشرة بالخوادم فهو المبدل الذي سيستقبل قواعد موازنة التدفق من وحدة التحكم ويقوم بناءً على هذه القواعد بتوزيع الطلبات بين الخوادم، وقد ذكرت إحدى الدراسات [30] وجود مبدل OpenFlow عالي الجودة هو المبدل HP Aruba 2920 24G PoE+ الذي لا يتجاوز سعره 1200.002 دولار أمريكي، وله معدل نقل يبلغ حوالي 128 جيجابت في الثانية ويسمح بتوصيل أكثر من 20 خادمًا، لكن هذا المبدل توقف الدعم الفني له لكن يوجد بدائل حديثة له توفر أداء عالي أيضاً.

وانطلاقاً من استنتاجات الدراسات السابقة والتي أكدت أن الخوارزميات الديناميكية يمكن أن تتفوق على الخوارزميات الثابتة عندما تكون ظروف الشبكة متغيرة قمنا بتنفيذ عدد من التجارب في ظروف شبكة متغيرة ومقارنة العمل المقترح مع الخوارزميات التالية:

- **Traditional Least Response [13]**

هي خوارزمية ديناميكية تعتمد على قياس وقت الاستجابة بطريقة تقليدية بالاعتماد على إرسال وحدة التحكم لرزم packet out بفاصل زمني معين وتسجيل وقت الإرسال ثم انتظار الرد من الخادم وتسجيل وقت الاستلام [13]، بعد ذلك يتم اختيار الخادم صاحب أقل زمن استجابة.

- **Least Bandwidth [30] [31]**

هي خوارزمية ديناميكية، تقوم باختيار الخادم الذي يستهلك أقل قدر من المرور (Traffic) على الشبكة للاستجابة للطلب التالي. تم الحصول على عرض النطاق الترددي للخوادم عن طريق إرسال وحدة التحكم لطلب إحصائيات التدفق إلى المبدل كل 5 ثوانٍ.

- **Round Robin [30] [31]**

هي خوارزمية ثابتة تعتمد على توزيع الطلبات بشكل دوري بين الخوادم، بحيث يتم إرسال كل طلب جديد إلى الخادم التالي في القائمة، ثم تعود إلى البداية بعد الوصول إلى آخر خادم.

- **Naïve Bayes [22]**

إلى جانب المقارنة الأساسية التي سيتم إجراؤها بين الخوارزمية المقترحة والخوارزميات الثلاث السابقة سنقوم بتنفيذ مقارنة إضافية مع خوارزمية تعلم آلي وهي Naïve Bayes ضمن جميع السيناريوهات المطبقة وذلك بهدف تقييم أداء النموذج المقترح ضمن سياق خوارزميات الذكاء الاصطناعي المستخدمة سابقاً في المجال ذاته. تم اختيار خوارزمية Naïve Bayes بناء على دراسة سابقة [22] تناولت موضوع توزيع الأحمال ضمن شبكات SDN باستخدام تقنيات التعلم الآلي، سبب اختيار هذه الخوارزمية هو أنَ تلك الدراسة وفرت مجموعة البيانات التي اعتمدت عليها ولكنها لم توفر الكود البرمجي المستخدم في تنفيذ النموذج. وبناءً عليه، تم إعادة بناء خوارزمية Naïve Bayes من الصفر، بالاعتماد على الخطوات المنهجية الموضحة في مخطط التدفق الخاص بالدراسة لضمان توافق النموذج المعاد تنفيذه مع المنطق الذي اعتمدته تلك الدراسة وتحقيق أعلى درجة ممكنة من الدقة. ولابدّ من الذكر أن خوارزمية Naïve Bayes هي واحدة من أهم خوارزميات التعلم الآلي المستخدمة في مهام التصنيف، تستند إلى مبرهنة بايز وتُستخدم لتوقع تصنيف عنصر معين بناءً على مجموعة من الميزات كما تمتاز ببساطتها وسرعتها ودقتها [32].

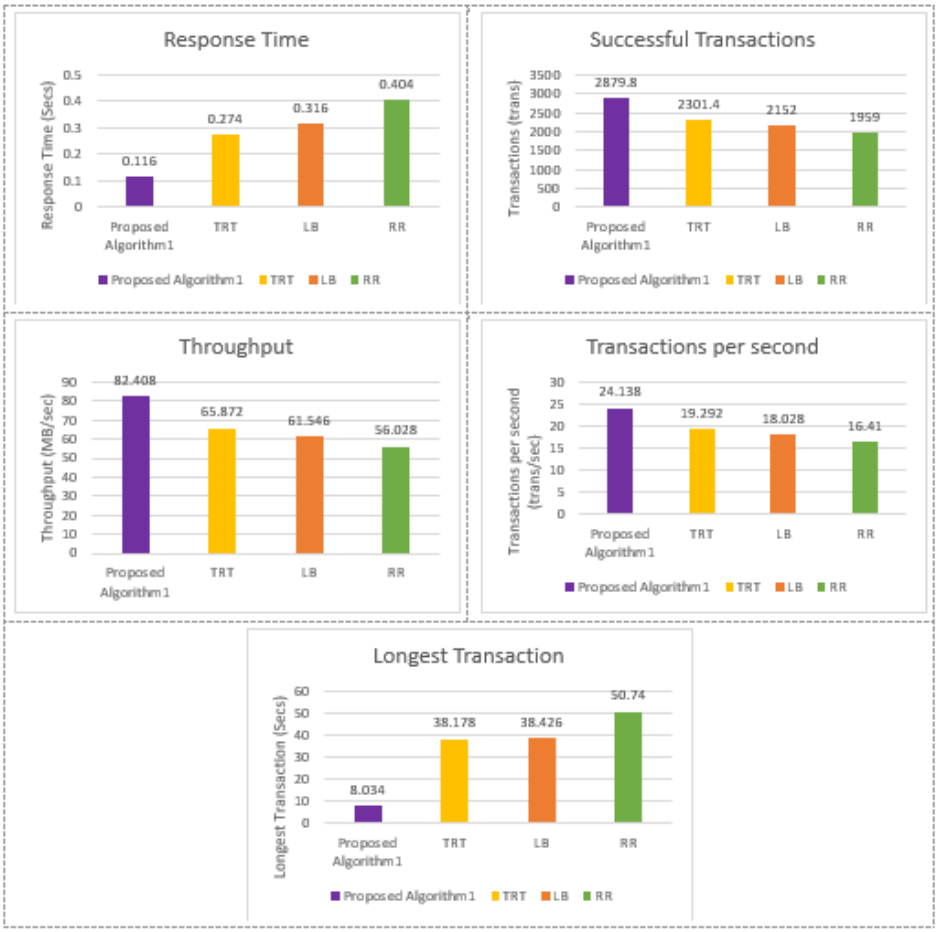
8-3 التحقيق العملي والتجارب

كما ذُكر سابقاً، تم تنفيذ عدد من السيناريوهات في ظروف شبكة متغيرة، كما تم قياس مجموعة من المعاملات واعتمادها في جميع التجارب وهي المعاملات الناتجة عن أداة Siege حيث شملت عدد المناقلات الناجحة، وقت الاستجابة، عدد المناقلات لكل ثانية، الإنتاجية، ووقت أطول مناقلة، وقد تم تنفيذ هذه السيناريوهات على النحو التالي:

8-3-1 السيناريو الأول

تم استخدام ثلاثة خوادم ويب متماثلة في المواصفات، وتم توليد حمل HTTP عبر أداة Siege إلى موازن الحمل لمدة دقيقتين بـ 15 مستخدمًا متزامنًا، مع توجيه حمل إضافي إلى الخادم الثاني فقط (ملف كبير الحجم بعدد مستخدمين متزامنين 15 لعكس الحالة التي يكون فيها الضغط على الخادم ناتج بشكل رئيسي عن حجم البيانات المنقولة أكثر من عدد الاتصالات) لخلق بيئة ديناميكية تختبر قدرة الخوارزميات على التكيف مع التغيرات. كما تم توليد حركة مرور شبكية متنوعة (غير HTTP) باستخدام أداة D-ITG تحاكي أنماط FTP و Video Streaming عبر TCP، لمحاكاة بيئة شبكة واقعية. أُعيد تنفيذ السيناريو خمس مرات من أجل كل خوارزمية، وحُسِب متوسط النتائج ورُسمت المخططات الموافقة.

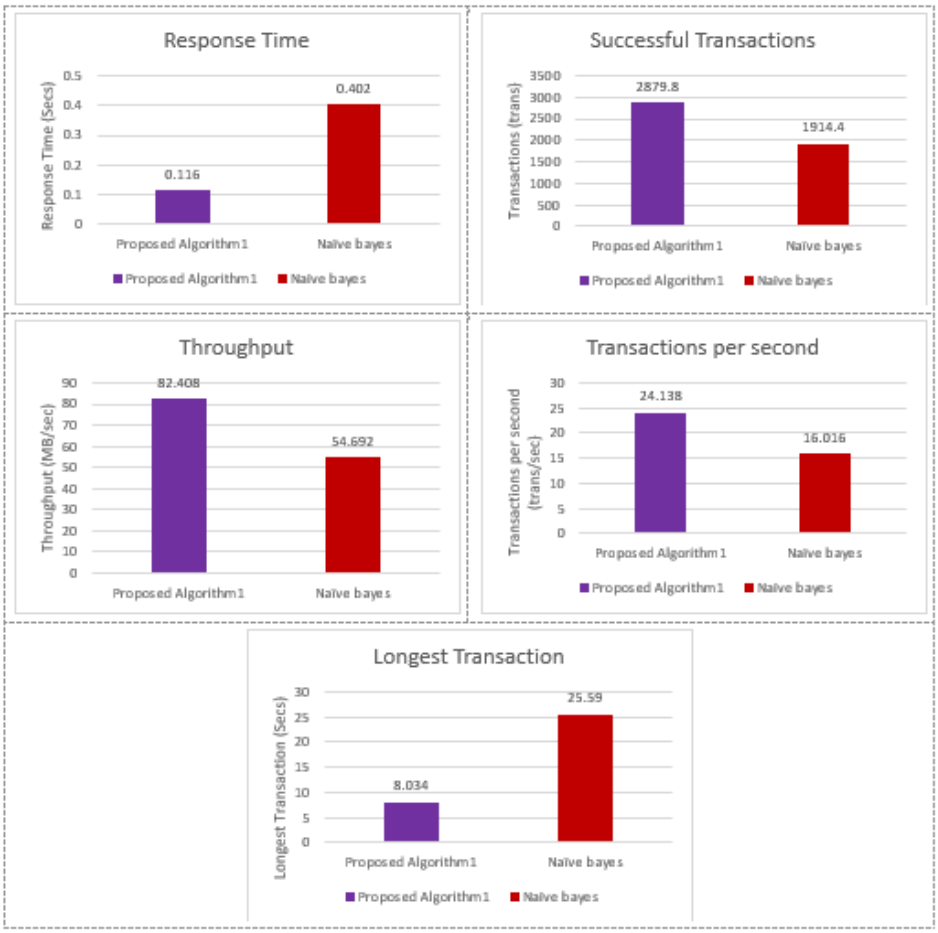
أ- المقارنة مع الخوارزميات التقليدية



الشكل (7) المقارنة مع الخوارزميات التقليدية في السيناريو الأول.

من النتائج السابقة نلاحظ تفوق أداء الخوارزمية المقترحة على الخوارزميات الأخرى من حيث كل البارامترات المقاسة.

ب- المقارنة مع خوارزمية Naïve Bayes



الشكل (8) المقارنة مع خوارزمية Naïve Bayes في السيناريو الأول.

من النتائج السابقة نلاحظ أيضاً تفوق الخوارزمية المقترحة على خوارزمية Naïve Bayes من حيث كل البارامترات المقاسة.

8-3-2 السيناريو الثاني

تم تطبيق نفس السيناريو الأول تماما مع تغيير طبيعة الحمل الإضافي الموجه إلى المخدم الثاني حيث تم استخدام ملف بحجم أصغر ولكن مع عدد اتصالات أكبر بلغ 100 اتصال. وعلى الرغم من أنّ

حجم الملف في هذا السيناريو أصغر، إلا أنّ زيادة عدد الاتصالات إلى 100 اتصال أدت إلى زيادة الحمل على المخدم الثاني بحوالي 46%، فبزيادة عدد الاتصالات زاد إجمالي البيانات المنقولة وعدد الطلبات التي يجب على الخادم معالجتها في الوقت نفسه وهذا ما يؤدي إلى استهلاك أكبر لموارد الخادم نتيجة إدارة عدد كبير من الجلسات المتزامنة، لذلك يعتبر هذا السيناريو أكثر ضغطاً على النظام والشبكة ككل بسبب زيادة حجم المرور المار عبر السويتش وبالتالي زيادة استهلاك موارد الشبكة وحدوث ضغط على البنية التحتية. نُفِّذ السيناريو خمس مرات لكل خوارزمية، وحُسِب متوسط النتائج ورُسمت المخططات الموافقة.

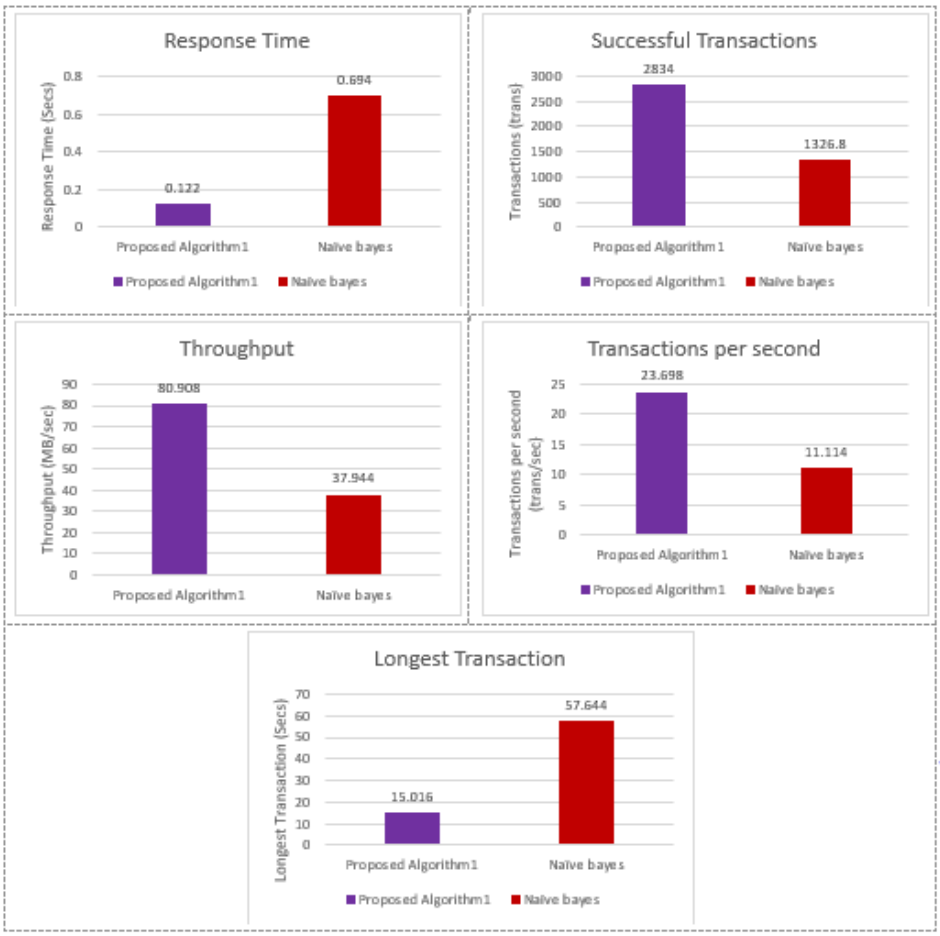
المقارنة مع الخوارزميات التقليدية



الشكل (9) المقارنة مع الخوارزميات التقليدية في السيناريو الثاني.

في هذا السيناريو أيضاً نلاحظ تفوق الخوارزمية المقترحة على الخوارزميات الثلاث الأخرى من حيث كل البارامترات المقاسة.

أ- المقارنة مع خوارزمية Naïve Bayes



الشكل (10) المقارنة مع خوارزمية Naïve Bayes في السيناريو الثاني.

من النتائج السابقة نلاحظ أيضاً هنا تفوق الخوارزمية المقترحة على خوارزمية Naïve Bayes من حيث كل البارامترات المقاسة.

❖ **تحليل وخلاصة السيناريوهين الأول والثاني**

للتأكد من جاهزية الخوارزمية للعمل في بيئات شبكية ديناميكية ومتعددة الاستخدامات، تم اختبارها في ظروف غير مثالية تحتوي مصادر متعددة للمرور. أظهرت النتائج تفوقها على الخوارزميات التقليدية الأخرى بفضل مراعاتها لعدة متغيرات رئيسية بدلاً من التركيز على عامل واحد، حيث استطاعت باستخدام التعلم الآلي التعامل مع العلاقات غير الخطية والمعقدة بين الموارد، مما أدى إلى توقعات دقيقة لزمن الاستجابة وقرارات أكثر كفاءة. بالمقابل، تفوقت أيضاً على خوارزمية Naïve Bayes التي اكتفت بتحليل حالة الخادم من خلال مؤشرات داخلية محدودة (استخدام المعالج والذاكرة والقرص) دون الأخذ بعين الاعتبار العوامل الشبكية؛ وهذا ما جعل الخوارزمية المقترحة تتفوق عليها بفضل شمولية المدخلات ومرونة النموذج وقدرته على التقاط العلاقات غير الخطية بين المتغيرات.

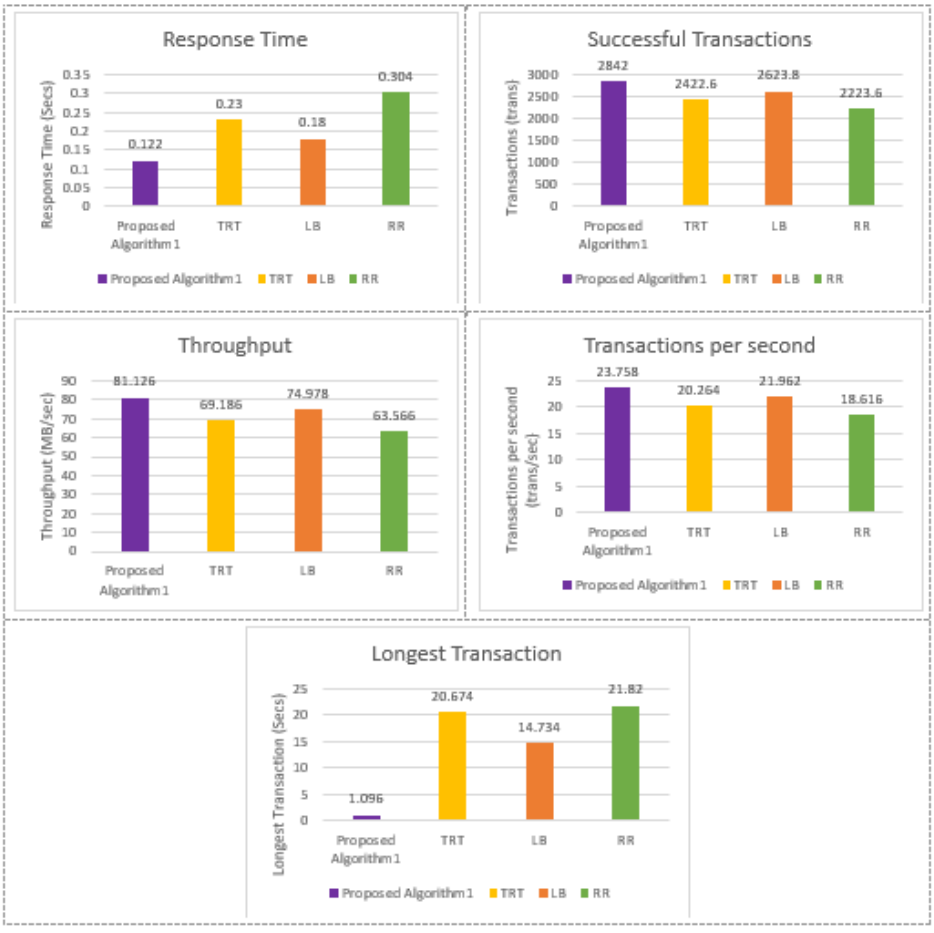
❖ **توسيع نطاق التجارب للتحقق من فعالية النموذج**

في البداية تم إجراء التجارب على ثلاث مخدمات فقط، مما أعطى تصوراً أولياً عن أداء الخوارزمية، لكن لضمان موثوقية النتائج وتقييم قدرتها على التكيف مع بيئات تشغيل مختلفة، تم توسيع نطاق الاختبار ليشمل خمس مخدمات. هذا التوسع يتيح لنا التأكد فيما إذا كان أداء الخوارزمية يتمتع بالثبات والدقة عبر أنظمة متعددة وليس مرتبطاً ببيئة محددة لتعزيز قابليتها للتطبيق العملي في سيناريوهات واقعية.

8-3-3 السيناريو الثالث

تم تطبيق نفس السيناريو الأول تماماً لكن باستخدام خمس مخدمات ويب.

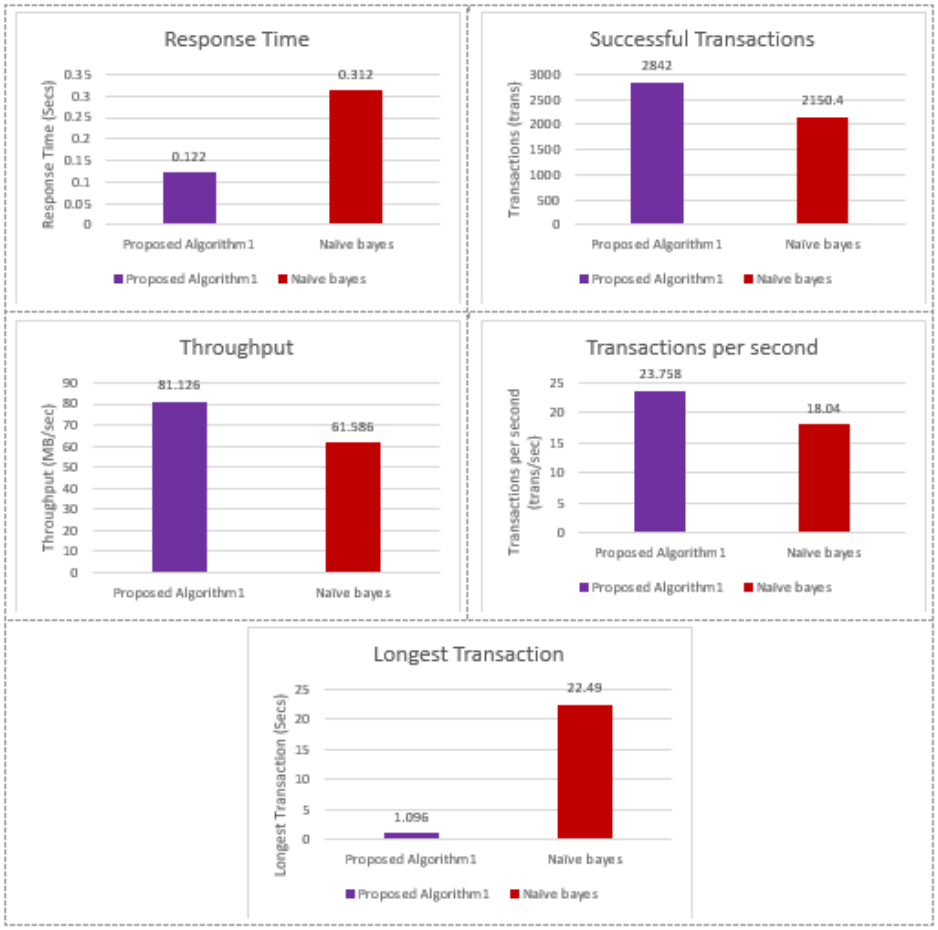
أ- المقارنة مع الخوارزميات التقليدية



الشكل (9) المقارنة مع الخوارزميات التقليدية في السيناريو الثالث.

مما سبق نلاحظ أنه بزيادة عدد المخدمات إلى خمسة بقيت الخوارزمية المقترحة تقدم نتائج أفضل من الخوارزميات التقليدية المُقارن معها.

ب- المقارنة مع خوارزمية Naïve Bayes



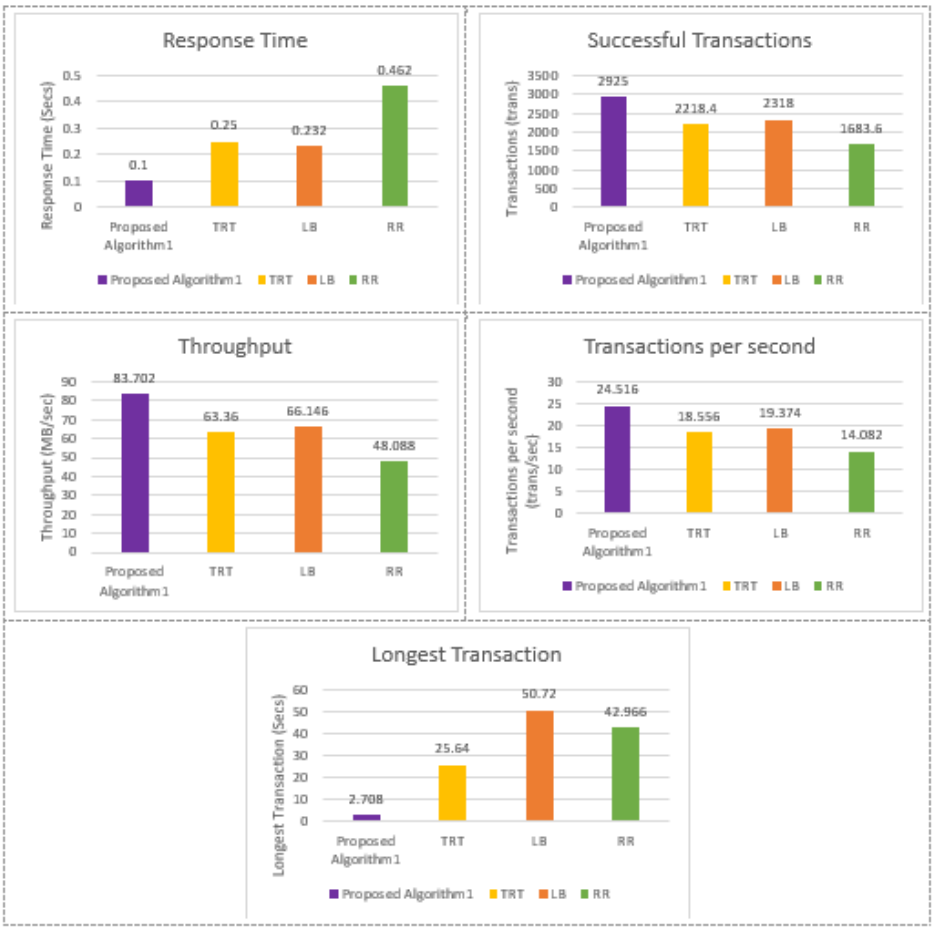
الشكل (10) المقارنة مع خوارزمية Naïve Bayes في السيناريو الثالث.

مما سبق نلاحظ أيضاً أن الخوارزمية المقترحة حافظت على تفوقها على خوارزمية Naïve Bayes بزيادة عدد المخدمات إلى خمسة.

8-3-4 السيناريو الرابع

تم تطبيق نفس السيناريو الثاني تماماً لكن باستخدام خمس مخدمات ويب.

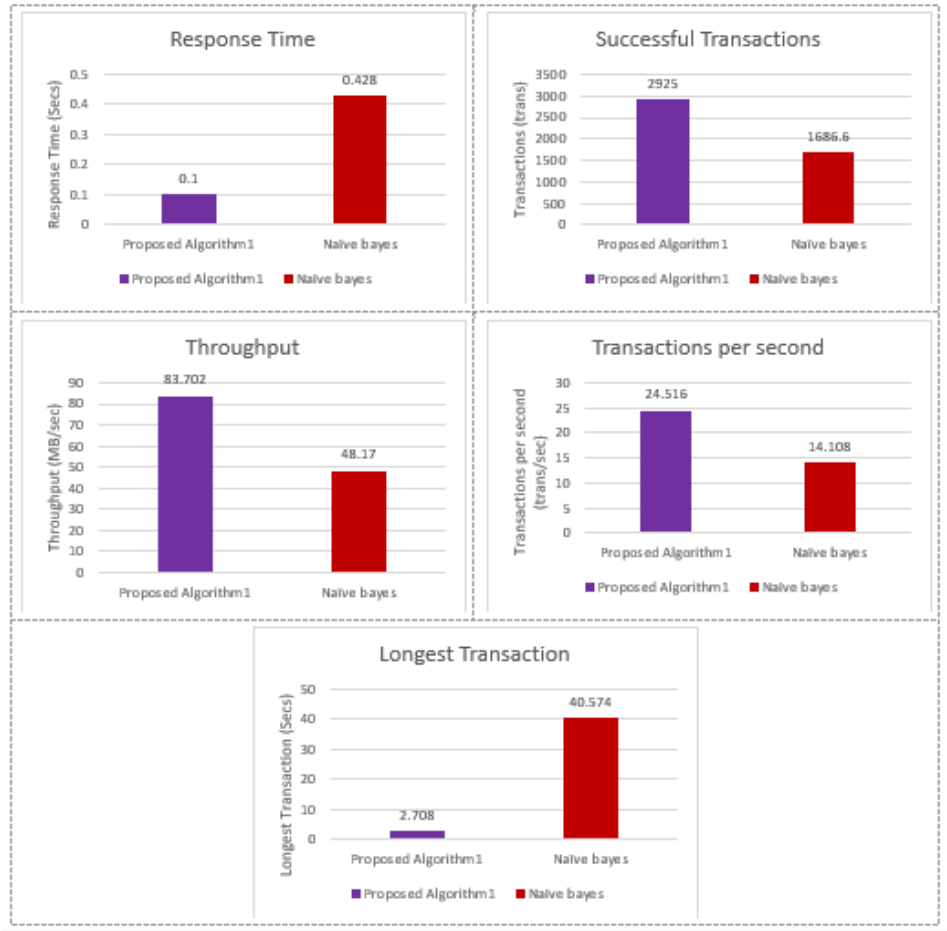
أ- المقارنة مع الخوارزميات التقليدية



الشكل (11) المقارنة مع الخوارزميات التقليدية في السيناريو الرابع.

في هذا السيناريو أيضاً نلاحظ تفوق الخوارزمية المقترحة على بقية الخوارزميات التقليدية بزيادة عدد المخدمات إلى خمسة.

ب- المقارنة مع Naïve Bayes



الشكل (12) المقارنة مع خوارزمية Naïve Bayes في السيناريو الرابع.

أيضاً في هذا السيناريو تفوقت الخوارزمية المقترحة على خوارزمية Naïve Bayes.

❖ تحليل وخلاصة السيناريوهين الثالث والرابع

تعتمد الخوارزمية المقترحة على آلية تنبؤية ذات تكلفة حسابية ثابتة وجمع لإحصائيات المخدمات، مما يجعلها عرضة لتأثير الحمل الناتج عن التوسع. في السيناريو الثالث، لم يكن الحمل الشبكي مرتفعاً، فكانت الفائدة من زيادة المخدمات إلى خمسة محدودة وأثر التكلفة أكثر بروزاً، لكنها بقيت الأفضل. أما

في السيناريو الرابع، فإن الشبكة كانت تحت ضغط عالٍ، فتجاوزت فائدة التوزيع الذكي تكلفة القرار وتحسن الأداء بزيادة المخدمات. أظهرت الخوارزميات التقليدية تحسناً ملحوظاً عند الانتقال من ثلاث إلى خمس مخدمات بفضل حساباتها المحلية، لكن بقيت الخوارزمية المقترحة هي الأفضل في إدارة الأحمال. بالمقارنة مع Naïve Bayes، تحسن أداء Naïve Bayes بشكل ملحوظ على خمسة خوادم بفضل بساطتها الحسابية وتكيفها مع البيئة الموسعة دون التأثر بحمل الإحصائيات الإضافي. مع ذلك، حافظت الخوارزمية المقترحة على تفوقها عليها بفضل قدرة الغابة العشوائية على التعامل مع البيانات متعددة الأبعاد ودمجها بطريقة فعالة عبر مجموعة من الأشجار التنبؤية، مما يتيح نتائج أكثر استقراراً وواقعية حتى في ظل بيئة تشغيل معقدة.

9. التحليل الإحصائي للنتائج السابقة

لإثبات تفوق الخوارزمية المقترحة بشكل علمي موثوق، تم إجراء تحليل إحصائي شامل على النتائج التجريبية المستخرجة من تنفيذ السيناريوهات الأربعة. تم اختيار مؤشري الأداء الأكثر أهمية في تقييم أنظمة موازنة الحمل، وهما: زمن الاستجابة (Response Time) والإنتاجية (Throughput)، للمقارنة الإحصائية بين الخوارزمية المقترحة والخوارزميات الأربعة الأخرى.

أظهر تحليل التباين الثنائي (Two-way ANOVA) [33] وجود تأثير رئيسي معنوي للخوارزمية على زمن الاستجابة ($F(4,80) = 178.86$، $p < 0.001$) وعلى الإنتاجية ($F(4,80) = 294.76$، $p < 0.001$)، مع وجود تأثير تفاعلي معنوي بين الخوارزمية والسيناريو في كلا المؤشرين ($p < 0.001$). وأكد التحليل اللاحق بطريقة توكي للمقارنات المتعددة (Tukey HSD) [34] والاختبار التائي المستقل (Independent Samples t-test) [35] أن الخوارزمية المقترحة حققت أداءً متفوقاً بشكل إحصائي معنوي على جميع الخوارزميات المقارنة في كلا المؤشرين. في زمن الاستجابة، سجلت الخوارزمية المقترحة أقل متوسط بلغ 0.1150 ± 0.0132 ثانية، متفوقةً على ثاني أفضل خوارزمية TRT بنسبة 59%، وكانت جميع الفروقات معنوية ($p < 0.001$). وفي الإنتاجية، سجلت الخوارزمية المقترحة أعلى متوسط بلغ 82.04 ± 1.81، متفوقةً على ثاني أفضل خوارزمية LB بنسبة 29%،

وكانت جميع الفروقات معنوية ($p < 0.001$). تفصيل الإحصائيات الكاملة موضح في الجداول التالية.

الجدول (1) مقارنة زمن الاستجابة

الخوارزمية	المتوسط ± الانحراف المعياري	الترتيب
Proposed Algorithm	0.0132 ± 0.1150	1
Traditional Response Time	0.0720 ± 0.2800	2
Least Bandwidth	0.0961 ± 0.2825	3
Naive Bayes	0.1542 ± 0.4590	4
Round Robin	0.1764 ± 0.4770	5

الجدول (2) مقارنة الإنتاجية

الخوارزمية	المتوسط ± الانحراف المعياري	الترتيب
Proposed Algorithm	1.81 ± 82.04	1
Least Bandwidth	9.01 ± 63.69	2
Traditional Response Time	7.77 ± 62.91	3
Round Robin	10.74 ± 50.93	4
Naive Bayes	9.31 ± 50.60	5

الجدول (3) الإحصائيات التفصيلية ـــ زمن الاستجابة

المقارنة	Tukey HSD (p-adj)	t-test (t)	t-test (p)	فرق المتوسطات	CI95%
PA vs TRT	0.0002	10.08−	0.001 >	0.1650−	[−0.197, −0.133]
PA vs LB	0.0002	7.72−	0.001 >	0.1675−	[−0.210, −0.125]
PA vs NB	0.001 >	9.94−	0.001 >	0.3440−	[−0.412, −0.276]
PA vs RR	0.001 >	9.15−	0.001 >	0.3620−	[−0.440, −0.285]

الجدول (4) الإحصائيات التفصيلية ـــ الإنتاجية

المقارنة	Tukey HSD (p-adj)	t-test (t)	t-test (p)	فرق المتوسطات	CI95%
PA vs LB	0.001 >	8.93	0.001 >	18.34	[14.32, 22.37]
PA vs TRT	0.001 >	10.72	0.001 >	19.12	[15.63, 22.62]
PA vs RR	0.001 >	12.78	0.001 >	31.11	[26.34, 35.88]
PA vs NB	0.001 >	14.83	0.001 >	31.44	[27.28, 35.59]

تُظهر الجداول 1 و2 متوسطات مؤشري الأداء (زمن الاستجابة والإنتاجية) مع الانحراف المعياري لكل خوارزمية، حيث القيمة الأقل في زمن الاستجابة والأعلى في الإنتاجية تشيران إلى الأداء الأمثل. أما الجداول 3 و4 فيُفصّلان النتائج الإحصائية للمقارنات الثنائية بين الخوارزمية المقترحة والخوارزميات الأخرى باستخدام الاختبار التائي المستقل (t–test) وطريقة توكي (Tukey HSD)، مع فترات الثقة 95%(CI95%) . جميع القيم $p < 0.001$ وفترات الثقة لا تشمل الصفر، مما يؤكد أن تفوق الخوارزمية المقترحة على جميع الخوارزميات إحصائياً معنوي ومؤكد في كلا المؤشرين.

10.الخاتمة

أظهرت هذه الدراسة فعالية استخدام تقنيات التعلم الآلي في تحسين أداء موازنة حمل الخوادم ضمن بيئة الشبكات المعرفة بالبرمجيات، من خلال التنبؤ بزمن استجابة الخوادم بناءً على مؤشرات الأداء الفعلية، فقد أثبتت الخوارزمية المقترحة قدرتها على تحقيق توزيع أكثر فعالية للحمل مقارنةً بالخوارزميات الأخرى. تفتح هذه النتائج آفاقاً واعدة لتطوير حلول ذكية وقابلة للتكيف في إدارة موارد الشبكات خاصة في البيئات الديناميكية والمعقدة.

1. المراجع

[1]. Diel, G., Miers, C. C., Pillon, M. A., & Koslovski, G. P. (2023). RSCAT: Towards zero touch congestion control based on actor–critic reinforcement learning and software-defined networking. *Journal of Network and Computer Applications*, 215, 103639.

[2]. Beissenova, G., Zhidebayeva, A., Kopzhassarova, Z., Kozhabekova, P., Myrzakhmetova, B., Kerimbekov, M., Ussipbekova, D., & Yeshenkozhaev, N. (2024). Load balancing in DCN servers through software defined network machine learning. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 15(2), 509-519.

[3]. Muhammad, T. (2022). A Comprehensive Study on Software-Defined Load Balancers: Architectural Flexibility & Application Service Delivery in On-Premises Ecosystems. *INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY*, 6(1), 1-24.

[4]. Maugendre, M. (2015). Development of a performance measurement tool for SDN (Master's thesis). Polytechnic University of Catalonia.

[5]. Ghazi, Y., & Mustafa, A. (2025). Improving single-controller software-defined networks architecture using named data networks architecture. *Homs University Journal*,*47*(2),49-76.

[6]. InterServer. (2019). *How to perform HTTP load testing using Siege*. https://www.interserver.net/tips/kb/http-load-testing-siege/.

[7]. Rodola, G. (2009–2025). *psutil documentation*. https://psutil.readthedocs.io/en/latest.

[8]. Botta, A., Donato, W., Dainotti, A., Avallone, S., & Pescapé, A. (2013). *D-ITG 2.8.1 manual*. COMICS Group, University of Napoli Federico II.

[9]. Alssaheli, O. M. A., Abidin, Z. Z., Zakaria, N. A., & Abas, Z. A. (2022). Software defined network based load balancing for network performance evaluation. *International Journal of Advanced Computer Science and Applications*, *13*(4), 117–124.

[10]. Linn, A. S., Win, S. H., & Win, S. T. (2019). Server load balancing in software defined networking. *National Journal of Parallel and Soft Computing*, *1*(1), 261–265.

[11]. Hamed, M. I., ElHalawany, B. M., Fouda, M. M., & Tag Eldien, A. S. (2017). Performance analysis of applying load balancing strategies on different SDN environments. *Benha Journal of Applied Sciences*, *2*(1), 91–97.

[12]. ElHalawany, B. M., Hamed, M. I., Fouda, M. M., & Tag Eldien, A. S. (2017). A new approach for server-based load balancing using software-defined networking. In *Proceedings of the 8th International Conference on Intelligent Computing and Information Systems (ICICIS 2017)* (pp. 155–160). IEEE.

[13]. Zhong, H., Fang, Y., & Cui, J. (2018). Reprint of: LBBSRT: An efficient SDN load balancing scheme based on server response time. *Future Generation Computer Systems*, *80*, 409–416.

[14]. Kumar, N., & Venkatesha Prasad, R. (2019). A comparative study on load balancing algorithms in software defined networking. In N. Kumar & R. Venkatesha Prasad (Eds.), *Ubiquitous communications and network computing (UBICNET 2019). Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering (LNICST)* (Vol. 276, pp. 142–150). Springer Nature.

[15]. Al-Bahadili, R. J., & Al-Saadi, A. (2020). An investigation of using traffic load in SDN based load balancing. *Iraqi Journal of Computer Communication Control and System Engineering*, *20*(2), 38–44.

[16]. Vani, K. A., & RamaMohanBabu, K. N. (2023). An intelligent server load balancing based on multi-criteria decision-making in SDN. *International Journal of Advanced Computer Science and Applications*, *14*(4), 888–896.

[17]. Soleimanzadeh, K., Ahmadi, M., & Nassiri, M. (2019). SD-WLB: An SDN-aided mechanism for web load balancing based on server statistics. *ETRI Journal*, *41*(2), 197–206.

[18]. Li, G., Gao, T., Zhang, Z., & Chen, Y. (2018). Fuzzy logic load-balancing strategy based on software-defined networking. *IEEE Access*, *6*, 2915–2924.

[19]. Malbašić, T., Bojović, P. D., Bojović, Ž., Šuh, J., & Vujošević, D. (2022). Hybrid SDN networks: A multi-parameter server load balancing scheme. *Journal of Network and Systems Management*, *30*(2), Article 32.

[20]. Krishna, P. M., Rao, S. S., Bhurat, S. S., Venkatesh, T., & Vani, K. A. (2021). Machine learning to balance web applications using SDN. *International Journal of Innovative Research in Management, Engineering and Computer Science (IJIRMPS)*, *9*(3), 201-207.

[21]. Hamdan, A. A., Trisnawan, P. H., & Bakhtiar, F. A. (2022). Implementation of load balancing using linear regression method on software defined network. *Journal of Information Technology and Computer Science Development*, *6*(7), 3560–3569.

[22]. Utomo, A. A., Trisnawan, P. H., & Primananda, R. (2020). Server load balancing mechanism using naive Bayes method with psutils agent on software defined network. *Journal of Information Technology and Computer Science Development*, *4*(10), 3961–3968.

[23]. Aldaher, M., Abbas, M., & Abboud, M. (2026). Review of server load balancing in software defined networks. *Homs University Journal*.

[24]. Xia, W., Wen, Y., Foh, C. H., Niyato, D., & Xie, H. (2015). A survey on software-defined networking. *IEEE Communications Surveys & Tutorials*, *17*(1), 27–51.

[25]. Ongaro, F. (2014). *Enhancing quality of service in software-defined networks* (Master's thesis). University of Bologna.

[26]. Wilk, P., & Nawrocki, P. (2014). Network management services based on the OpenFlow environment. *Computer Science*, *15*(4), 45–60.

[27]. المحلل العربي. (2017). *تطبيع البيانات: مهارة على كل محلل إتقانها*. https://www.arabiananalyst.com/2017/11/17/normalizing-data-an-analytical-trick-you-must-know-ar/

[28]. Muraina, I. O. (2021). Ideal dataset splitting ratios in machine learning algorithms: General concerns for data scientists and data analysts. In *Proceedings of the 7th International Mardin Artuklu Scientific Research Conference* (pp. 496–504). Mardin, Turkey.

[29]. Hamdan, A. A., Trisnawan, P. H., & Bakhtiar, F. A. (2022). Implementation of load balancing using linear regression method on software defined network. *Journal of Information Technology and Computer Science Development*, *6*(7), 3560–3569.

[30]. Aguilar, L. C. F. P., & Batista, D. M. (2019). Effectiveness of implementing load balancing via SDN. In *Proceedings of the Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2019)* (pp. 535–548).

[31]. Padilha, L. (n.d.). *pox* [Computer software]. GitHub. https://github.com/lcfpadilha/pox

[32]. علوم24. (n.d.). *فهم خوارزمية Naïve Bayes مع مثال عملي في Python*. https://sciences24.com/%d8%ae%d9%88%d8%a7%d8%b1%d8%b2%d9%85%d9%8a%d8%a9-naive-bayes/

[33]. Seltman, H. W. (n.d.). Chapter 11: Two-way ANOVA. In *Experimental design and analysis*. Department of Statistics, Carnegie Mellon University.

[34]. Tukey, J. W. (1949). Comparing individual means in the analysis of variance. *Biometrics*, *5*(2), 99–114. https://doi.org/10.2307/3001913

[35]. Student. (1908). The probable error of a mean. *Biometrika*, *6*(1), 1–25. https://doi.org/10.1093/biomet/6.1.1

[36].