تحسين الأداء في وحدة تحكم الشبكات المعرَّفة برمجياً HPE VAN باستخدام معدات وظائف الشبكات الحاسوبية

د. محمد مازن محايري*

د. رؤوف حمدان**

نزيه احمد حرفوش***

(تاريخ الإيداع 11 / 10 / 2020. قُبِل للنشر في 21/ 1 / 2021)

□ ملخّص □

تعتمد شبكات الكمبيوتر اليوم على مجموعة واسعة من الصناديق الوسطى لتحسين أمان الشبكة وأدائها وذلك من خلال توفير الخدمات مثل جدار الحماية وترجمة عناوين الشبكة (NAT) وكذلك موازنة الحمل. تلجأ هذه الشبكات في إدارتها إلى الشبكات المعرفة برمجياً ((SDN) Software Defined Networks) التي يمكن أن توفر بيئة مركزية قابلة للبرمجة تُستَخدم لتوجيه حركة المرور من خلال تحقيق سلسلة الخدمة المطلوبة بحيث تنفَّذ الصناديق الوسطى كبرامج في الأجهزة الافتراضية في الشبكات المعرفة برمجياً [1] (SDN) ومع ذلك، حتى لو كانت وحدة التحكم في(SDN) قادرة على تحديد أفضل مسار لصندوق وسيط معين (معدات وظائف الشبكة)، فإن عدم المعرفة المسبقة بسياسة هذه المعدات قد يؤدي إلى اختناقات مرورية، مما يؤدي إلى انخفاض أداء الشبكة بشكل عام. من خلال الأخذ بالاعتبار خصائص المعدات المستخدمة وخصائص الشبكات المعرفة برمجياً (SDN)، سيتم تصميم طريقة توجيه خفيفة الوزن وفعالة من حيث التكلفة واعية لسياسة معدات وظائف الشبكة لمواجهة هذا التحدي. أي تتم إضافة وحدة ترجمة على شكل صندوق وسيط (middle box) إلى بنية (SDN) لمساعدة هذه المعدات على إعطاء "تلميحات" لتوجيه وحدة التحكم في تنسيق واجهة برمجة التطبيقات (REST API) حتى يمكن اتخاذ قرارات توجيه أفضل.

الكلمـــات المفتاحيـــة: الشـبكات المعرفــة برمجيــاً، برتوكـول openflow، أداء الشـبكة، الصــناديق الوســطى، وحدة التحكم HPE VAN.

* أستاذ –كلية الهمك– جامعة دمشق– دمشق– سورية.

**أستاذ –كلية الهمك– جامعة دمشق– دمشق– سورية.

***طالب دكتوراه – هندسة الحواسيب وشبكاتها – قسم هندسة الحواسيب والأتمتة – كلية الهمك – جامعة دمشق– دمشق– سورية.

Improve Performance in Software Defined Network Controller's (HPE VAN) Using Computer Network Functionality Equipment

Dr. Mohammad Mazen Mehayri*
Dr. Raouf Hamdan**
Nazeeh Harfoush***

(Received 11 / 10 / 2020. Accepted 21 / 1 / 2021)

□ ABSTRACT □

Computer networks today rely on a wide range of Middleboxes specialized to improve network security and performance by providing services such as firewall and network address translation (NAT) as well as load balancing. In their management, these networks resort to Software Defined Networks (SDN), which can provide a central programmable environment that is used to direct traffic by achieving the required service chain so that the middleboxes are implemented as programs in virtual machines in Software Defined Networks (SDN) [1]. However, even if the SDN controller is able to determine the best path for a specific middlebox (network function equipment), lack of prior knowledge of the policy of this equipment may lead to traffic jams, which leads to a significant decrease in network performance in general. By considering the characteristics of the used equipment and the characteristics of software defined networks (SDN), a lightweight and cost-effective routing method that is aware of the network functionality equipment policy will be designed to address this challenge. That is, a middle box is added to the SDN structure to help this equipment give "hints" to guide the REST API in order to make better routing decisions.

Keywords: Software Defined Networks, OpenFlow, Network Performance, Middleboxes, HPE-VAN.

* **Professor- Faculty of Electrical and Mechanical- Damascus University- Damascus- Syria.**

** **Professor- Faculty of Electrical and Mechanical- Damascus University- Damascus- Syria.**

*** **Postgraduate Student- Department of Computers and Automatics- Faculty of Electrical and Mechanical- Damascus University- Damascus- Syria.**

مقدمة:

الصناديق الوسطى (أو معدات وظائف الشبكة) عبارة عن تجهيزات شبكية تتوضع على المسار بين المصدر والوجهة وتنفِّذ وظائف متعددة من ضمنها توجيه الحزم والترشيح وهندسة حركة سير الدفق إلخ. تقوم بنية (SDN) بإنشاء شبكة مركزية التحكم قابلة للبرمجة وذلك من خلال فصل طبقة التحكم في التجهيزات الشبكية عن طبقة توجيه البيانات. تتصل الطبقتان مع بعضهما البعض من خلال بروتوكول معين مثل (OpenFlow) الذي يمكّن وحدات التحكم في الشبكة من تحديد مسارات حزم البيانات عبر شبكة من المبدلات. استُبدلت جداول التوجيه بجداول تتناسب مع البروتوكول الجديد (OpenFlow) تدعى هذه الجداول بجداول التدفق (Flow Tables) حيث تُخزَّن ضمن المبدل، ومدخلات هذه الجداول تدعى قواعد التدفق (Flow Roles)، يمكن لوحدة التحكم أن تقرر تعديل قواعد التدفق الحالية أو نشر قواعد تدفق جديدة عن طريق إضافة أو تعديل أو إزالة القواعد والإجراءات من جداول التدفق بالإضافة إلى ذلك يوفر بروتوكول (OpenFlow) مجموعة واسعة من حقول المطابقة مثل (Ethernet) و (VLAN) وعناوين (IP) وأرقام المنافذ إلخ، وتشتمل أنواع الإجراءات التي يدعمها على مجموعة متنوعة من إجراءات إعادة التوجيه مثل إسقاط الحزم وتطبيق علامات (VLAN) وإعادة توجيه الحزم إلى منافذ مبدلات (Switches) معينة أو إعادة توجيها إلى وحدات التحكم، بمعنى آخر يمكن من خلال بروتوكول (OpenFlow) التحكم في الشبكة باستخدام خوارزميات إعادة التوجيه المخصصة.

نلاحظ أنه حتى في بيئة الشبكات المعرفة برمجياً والقابلة للبرمجة بدرجة كبيرة والتي تحافظ على النظرة العامة للشبكة، فقد نشهد بعض اختناقات الأداء بسبب السياسات غير المعروفة لمعدات وظائف الشبكة المستخدمة ضمن هذه الشبكة، ولمعالجة هذا الأمر نقوم بتصميم طريقة توجيه خفيفة الوزن وفعالة من حيث التكلفة ومدركة لسياسة الصندوق الوسيط من خلال النظر في خصائص معدات وظائف الشبكة (الصناديق الوسطى) و (SDN).

أُخِذَ بالاعتبار أيضاً محدودية سعة ذاكرة المبدل (switch memory capacity) من خلال الاستفادة من إحصائيات تدفق (OpenFlow) لتحديد القاعدة التي يجب إضافتها أو حذفها عندما تنفد المبدلات من الذاكرة. من خلال دراسة الأداء والقائمة على المحاكاة (باستخدام Mininet)، يظهر أنه مقارنةً بالطرائق الحالية فإن تصميم صندوق المعدات الوسيطة المقترح يؤدي إلى أداء أفضل في الأنظمة التي تدعم (SDN) وذلك عن طريق تقليل استخدام عرض النطاق الترددي للشبكة في أنظمة جدار الحماية، أو تقليل وقت الاستجابة الكلي وكذلك زيادة الإنتاجية في أنظمة موازنة التحميل.

أهمية البحث وأهدافه:

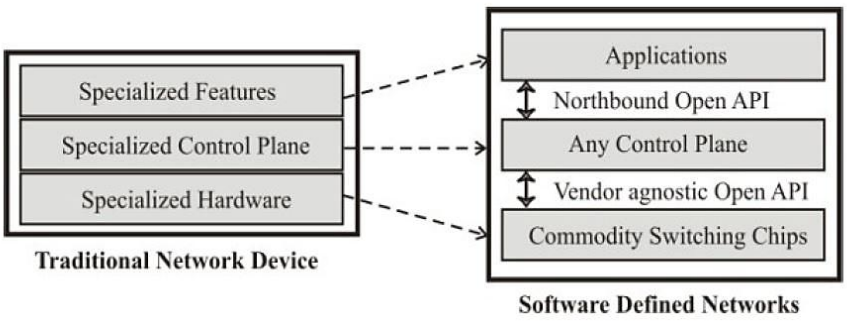
الهدف من هذا البحث هو تحسين الأداء الكلي للشبكات المعرفة برمجياً من خلال تطبيق وتنفيذ صندوق وسطي ضمن الشبكة. سيُنَفَذ الصندوق الوسطي المقترح ضمن المتحكم، لمحاولة تحقيق أداء أفضل واستقرار المتحكم ومَنِع التحميل الزائد. علاوة على ذلك، استخدام وتحديد مقاييس للتوعية أثناء وضع واستخدام طوبولوجيا الشبكة، وكل ذلك سيؤدي إلى زيادة الأداء في البيئات التي تعتمد على عدد كبير من المبدلات. سيُحقَق ذلك بإنشاء تدفقات ضمن مسارات أُنشِأَت ديناميكياً.

طرائق البحث ومواده:

يبدأ هذا البحث بإجراء دراسة مرجعية عن الشبكات المعرفة برمجياً، وكذلك دراسة أداء هذا النوع من الشبكات وإضافة صندوق وسطي الى بنية الشبكة المستخدمة في الدراسة، ومن ثم تسجيل النتائج ومقارنتها مع نتائج الأداء دون استخدام الصندوق الوسطي. ولتقييم التصميم المقترح استُخدِمَ (Mininet v2.2) لبناء سلسلة من طوبولوجيا مبدلات (SDN) والمضيفات على شكل معدات وظائف الشبكة و (OpenvSwitch OVS) لمضاهاة مبدلات SDN ولمعدات وظائف الشبكة التقليدية والبرمجية نعمل على (Linux: iptables) و (HAProxy) أما بالنسبة لوحدة تحكم الشبكة نستخدم (HPE VAN Controller).

الشبكات المعرفة برمجياً (SDN) وبروتوكول (Openflow):

يقترن مستوى البيانات ومستوى التحكم في الشبكات التقليدية بإحكام على نفس الجهاز الشبكي (الشكل 1) [2]. لذلك في الشبكات التقليدية يكون تطوير تطبيقات جديدة والتعديل في سلوك الأجهزة الموجودة مهمة صعبة للغاية. تتغلب الشبكات المعرفة برمجياً [3] على هذه المشاكل عن طريق نقل منطق التحكم من الأجهزة الشبكية إلى موقع مركزي. يسمى منطق التحكم المنقول متحكم الشبكة المعرفة برمجياً (SDN Controller) أو نظام تشغيل الشبكة (Network Operating System (NOS)). ويكون لدى المتحكم رؤية عامة للشبكة بالكامل لذلك باستخدام الشبكات المعرفة برمجياً يمكن إدارة وظائف الشبكة بطريقة فعالة جداً.



الشكل 1 فصل مستوى التحكم عن مستوى البيانات في الشبكات المعرفة برمجياً

يستخدم بروتوكول (OpenFlow) والذي هو بروتوكول قياسي لتوفير اتصال بين المتحكم وجهاز الشبكة [4] حيث يسمى المتحكم مستوى التحكم بينما يطلق على الأجهزة الشبكية تسمية مستوى البيانات. ومتحكم (Openflow) هو المسؤول عن تحديد أي إجراء يُنَفذ بواسطة المبدل حيث يكون نهج القرار إما تفاعلي (Reactive) أو استباقي (Proactive).

حسب النهج التفاعلي عندما تصل الحزمة إلى المبدل، في حال كان المبدل لا يعرف كيفية التعامل مع تلك الحزمة، سيُرسل الحزمة إلى المتحكم المسؤول عن إدراج مدخل التدفق (flow–entry) في جدول تدفق المبدل باستخدام بروتوكول (OpenFlow).

إنَّ العيب الرئيسي لهذا النهج أنَّ المبدل يعتمد كلياً على قرار المتحكم بالنتيجة عندما يفقد المبدل الاتصال مع المتحكم فإنه لا يمكن التعامل مع تلك الحزمة.

بينما في النهج الاستباقي، يبدأ المتحكم بملء مدخلات الدفق في جداول التدفق لكل مبدل، وفي هذا النهج يتغلب المبدل على القِصر الموجود في النهج التفاعلي حيث أنَّه في حال فقدان الاتصال مع المتحكم فإن ذلك لا يعطل حركة المرور.

المزايا الرئيسية للشبكات المعرفة برمجياً التي تتميز فيها عن النهج التقليدي للشبكات هو أن الشبكات المعرفة برمجياً تسمح باختبار سرعة ونشر تطبيقات جديدة في شبكة حقيقية، والحد من رأس المال ونفقات التشغيل وتسمح بالإدارة المركزية لكل المبدل.

بنية النظام

تتمثل الفكرة الأساسية في تضمين وحدة تسمى "مترجم السياسة الوسطى" لترجمة سياسات معدات وظائف الشبكة (الصناديق الوسطى) إلى نموذج (REST API) بحيث يمكن تثبيتها في مبدلات (OpenFlow) بواسطة وحدة التحكم لتحديد السياسات الأمامية المستندة إلى كليهما. لشرح كيفية قيام فكرتنا بتحسين أداء النظام بشكل فعال، اعتبرنا وجود نوعين من الصناديق الوسطى في الشبكة: جدار الحماية وموازنات التحميل.

1.2 جدار الحماية

جدار الحماية هو نظام أمن شبكة يستند إلى البرامج أو الأجهزة، وعادة ما يكون موجوداً بين شبكة داخلية موثوق بها وشبكة أخرى (خارجية) تتحكم في حركة مرور الشبكة الواردة والصادرة استناداً إلى مجموعة من القواعد. على سبيل المثال في شبكة مؤسسة ما، قد يقوم مركز الشبكات بتصفية كل حركة المرور الواردة والصادرة لضمان الأمن. افترض الآن أن هناك من يريد إرسال حزمة تعد غير آمنة إلى الإنترنت. ستمر هذه الحزمة ببعض أجهزة التوجيه ثم تصل إلى جدار الحماية وتتطابق مع قاعدة في مجموعة قواعد جدار الحماية، ثم تسقط في النهاية.

ومع ذلك في بيئة الشبكات المعرفة برمجياً إذا تمكنت جدران الحماية من تقديم المعلومات مسبقاً فيمكن إسقاط الحزم مسبقاً وبالنتيجة توفير النطاق الترددي للشبكة ومواردها. أحد الأساليب الممكنة هو "ترحيل جدار الحماية" [5]، ولكن هذا مكلف من حيث الوقت والموارد. فكرتنا هي الاستفادة من حقيقة أن مبدل (OpenFlow) يمكن أن يحافظ على حالة تدفق حركة التطبيق.

النهج المقترح هو تقديم تطبيق لترجمة قواعد جدار الحماية وتثبيتها في مبدلات (OpenFlow)، تتكون قواعد جدار الحماية من 7 أجزاء: البروتوكول، عنوان المصدر، منفذ المصدر، عنوان الوجهة، منفذ الوجهة، عنوان MAC، وإجراء الحزم.

يوضح الجدول 1 مثالًا عن كيفية ترجمة قواعد (iptables) إلى واجهة برمجة التطبيقات (REST API). يأخذ هذا العمل في الاعتبار فقط جدران الحماية عديمة الجنسية، والتي تقيد أو تحظر الحزم بناءً على عناوين المصدر والوجهة أو القيم الثابتة الأخرى. أُهملت حالة تشغيل اتصالات الشبكة بمعنى آخر جدران الحماية عديمة الجنسية ليست على معرفة بأنماط حركة المرور أو البيانات المنخفضة ويمكن لجدار الحماية ذي الحالة من ناحية أخرى مراقبة حركة مرور الشبكة من طرف إلى آخر بتتبع حالة اتصالات الشبكة (مثل تدفقات TCP أو اتصالات UDP) وحفظ سمات مهمة لكل اتصال في الذاكرة. تُترك هذه الأنواع من القواعد لتُعالج بواسطة جدار الحماية الأصلي ويمكن أن تكون جزء من الدراسات المستقبلية.

جدول 1 مثال عن ترجمة قواعد جدار الحماية الى لغة واجهة برمجة التطبيقات

	أوامر جدول IP	حقول واجهة برمجة التطبيقات
البروتوكول	-p	Nw-proto
عنوان المصدر	-s	Src-ip

منفذ المصدر	-sport	Tp-src
عنوان الوجهة	-d	Dst-i
منفذ الوجهة	-dport	Tp-dst
عنوان MAC	-MAC-source	Src-mac
الإجراء	–j	action

1.3 موازن الحمل

موازن الحمل هو جهاز متخصص يحتوي عادةً على معلومات حول الخوادم التي ترسل إليها عند وضع موازن الحمل في نقاط الاختناق تكون مسؤولة عن مراقبة صحة الخادم وتوزيع الحمل بالتساوي عبر مجموعة من الخوادم الخلفية (المعروفة أيضًا باسم مزرعة الخوادم أو تجمع الخوادم)، قد يعتمد التوجيه على الحمل الذي تم الإبلاغ عنه للخوادم أو أقصر أوقات الاستجابة أو موقع المحتوى أو الجولة المستديرة أو ببساطة اختيار عشوائي ولكن عادةً لا تأخذ موازنات الحمل الحالية حالة الشبكة. يستخدم النهج المقترح الشبكات المعرفة برمجياً للحصول على معلومات الشبكة وجدولة الطلبات للخوادم بناءً على كل من تحميل الشبكة والخادم. ومع ذلك فإن هذا الأمر أكثر تعقيدًا نظرًا لأن موازنة الحمل عادة ترتبط ارتباطًا وثيقًا بهندسة المرور وأُقتُرِحَ برنامج (Plug–n–Serve) (المعروف الآن باسم Aster*x) كتطبيق هندسة المرور لـ (SDN/OpenFlow) عند وصول الطلب ، تقرر وحدة التحكم الخادم الذي سيتم التوجيه إليه والمسار المقابل. ويُتَخذ القرار بناءً على طوبولوجيا الشبكة وازدحام الشبكة والتحميل على الخوادم. يحتوي (Aster*x) على ثلاث وحدات وظيفية رئيسية: مدير التدفق ومدير الشبكة ومدير المضيف. يدير مدير التدفق مسارات التدفق باستخدام خوارزمية جدولة موازنة الحمل المختارة، بينما يحتفظ مدير الشبكة بمعلومات هيكل الشبكة واستعلامات التبديل بشكل دوري للحصول على استخدام الارتباط، ويتتبع مدير المضيف حالة جميع الخوادم والأحمال في النظام، ويرسل التقارير إلى مدير التدفق. نلاحظ أن هذا يتطلب مديري المضيف للخوادم الفردية، سيقوم النهج المقترح ببساطة بإنشاء وحدة لترجمة سياسة موازن الحمل للحصول على حالة الخادم من موازنات الحمل الحالية[6]. لذلك، يمكن لوحدة التحكم تحديد الخوادم المناسبة للطلب الحالي ثم اختيار خادم للطلب بناءً على تحميل الشبكة الحالي، ويقدم الجدول 2 المعلومات المستخرجة من (HAProxy) وترجمتها إلى لغة واجهة برمجة التطبيقات. حيث يمكن تحديد عناوين IP الظاهرية وتجمعات الخوادم وإرسالها إلى وحدة التحكم باستخدام واجهة برمجة التطبيقات (REST API). ومع ذلك، فإن خوارزمية الجدولة تتطلب بذل جهد إضافي لتنفيذ خوارزمية موازنة الحمل في وحدات التحكم بناءً على تحميل الخادم والشبكة، بما في ذلك نوع الطلب واستخدام الارتباط وازدحام الشبكة المحتمل.

جدول 2 مثال عن ترجمة سياسة موازن الحمل

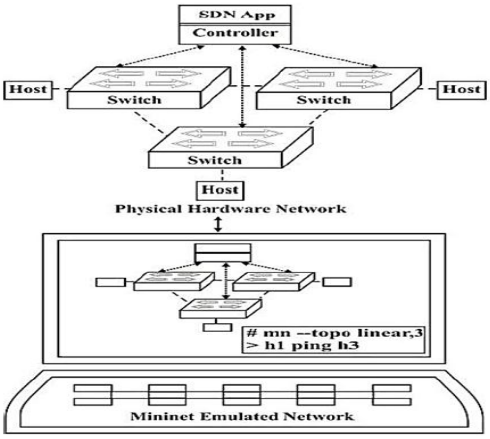
	HAproxy	حقول واجهة برمجة التطبيقات
عنوان IP لموازن الحمل	Stats uri	Address (Vip)
عنوان IP حقيقي للخادم	server	Address (member)
خوارزمية الجدولة المستخدمة	balance	Implem. at HPE–VAN controller

النتائج العلمية

1.4 المحاكي Mininet

(Mininet) هو المحاكي المستخدم لنشر الشبكات الكبيرة على الموارد المحدودة والمؤلفة من جهاز كمبيوتر واحد بسيط أو على جهاز افتراضي، وقد أنشئ هذا المحاكي لتمكين البحوث في كل من الشبكات المعرفة برمجياً وبروتوكول (OpenFlow) ويسمح المحاكي (Mininet) بتشغيل الكود غير المعدل بشكل تفاعلي على أجهزة افتراضية ضمن جهاز حاسب بسيط، ويوفر الملائمة والواقعية بتكلفة منخفضة جداً. والبديل عن منصة (Mininet) هو أسرة الاختبار العتادية (Hardware) السريعة والدقيقة ولكنها مكلفة للغاية، وبدلاً عن ذلك استخدمنا الخيار الآخر وهو المحاكاة (simulator) التي تكون رخيصة جداً ولكن في بعض الأحيان بطيئة، وتتطلب تعديل التعليمات البرمجية، وباستخدام (Mininet) تم ضمان سهولة الاستخدام والدقة في الأداء وقابلية التوسع.

تسمح منصة (Mininet) بخلق طوبولوجيا ذات حجم كبير جداً تصل إلى آلاف العقد وإجراء الاختبار عليها بسهولة جداً [7]. ولديها أدوات لتنفيذ الأوامر السطرية (command line) بسيطة جداً كما تتضمن واجهة برمجة تطبيقات (API). ويسمح (Mininet) للمستخدم خلق وتخصيص وتبادل واختبار الشبكات المعرفة برمجياً بسهولة (الشكل 2).



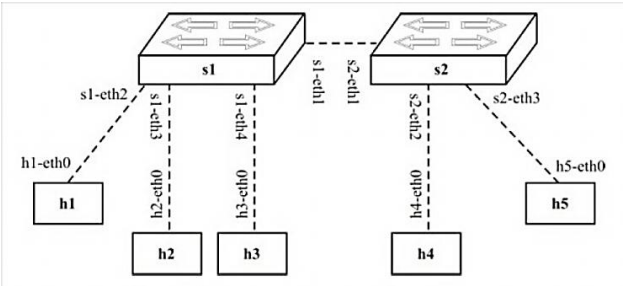
الشكل 2 محاكاة شبكة حقيقية على منصة Mininet

استخدمنا جهاز حاسب يحمل المواصفات الموضحة الشكل 3 لتطبيق التجارب العملية.

الشكل 3 المواصفات الفنية لجهاز الحاسب المُستخدم في تطبيق التجارب العملية

6.1.1 إنشاء طوبولوجيا مخصصة

يمكن بسهولة إنشاء طوبولوجيا مخصصة باستخدام (Mininet) [8]. على سبيل المثال خلق طوبولوجيا مخصصة تحوي مبدل عدد/2/ وخمسة مضيفين (الشكل 4) يحتاج فقط كتابة بضعة أسطر من كود بايثون [9]، وبالمثل يمكن بسهولة إنشاء شبكة معقدة جداً مرنة وقوية. ويمكن أيضاً تكوين هذه الطوبولوجيا استناداً إلى البارامترات التي ستمرر إليها، وإعادة استخدام تلك الطوبولوجيا في تجارب متعددة.



الشكل 4 إنشاء طبولوجيا مخصصة في Mininet

1.5 جدار الحماية

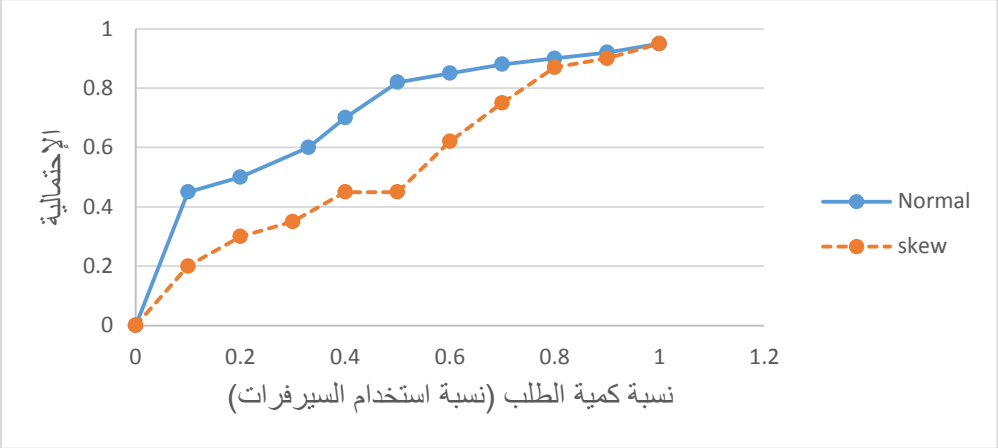
يتم النظر في طوبولوجيا شبكة الإنترنت ذات الـ 14 عقدة و14 امتداد في كل طوبولوجيا، تحتوي كل عقدة على مضيفين متصلين بها. لقد عُيّنَ عرض النطاق الترددي وحُمّلَ كل رابط إلى القيم الافتراضية نظراً لأن مقدار الحركة ضئيل في هذا السيناريو. أُرسلت حزم اختبار باستخدام بروتوكولات مختلفة، عنوان الشبكة للمصدر، منفذ المصدر، عنوان الشبكة للوجهة، منفذ الوجهة، وما إلى ذلك ونرى ما إذا كنا نستطيع الحصول على النتيجة المرجوة أم لا عندما تتطابق مع قواعد معينة لجدار الحماية. ثم حُددت قواعد جدار الحماية باستخدام جداول عناوين الشبكة.

باستخدام مجموعة قواعد جدار الحماية المحددة، أُنشئت حركة مرور بشكل عشوائي وقيسَ وقت الاستجابة (بين إرسال حزمة وتلقي استجابة أو الحزمة المُسقطة). نختبر طريقتنا باستخدام مجموعتين من قواعد جدار حماية مختلفة لكل طوبولوجيا للحصول على متوسط وقت الاستجابة، بشرط متوسط الاحتمال 0.8 لرزمة لتمرير جدار الحماية. يمكن تجاهل الحزم المطابقة لقاعدة "Dropped" في جدار الحماية على الفور بدلاً من إسقاطها بعد وصولها إلى جدار الحماية. بدلاً من ذلك، يمكننا أيضاً مراعاة محدودية سعة الذاكرة لجدول التدفق. عندما يكون جدول التدفق ممتلئاً، فإننا نستخلص إحصائيات التدفق لتحديد عدد التطابقات لكل قاعدة، ثم نختار القواعد المراد استبدالها وفقًا لذلك. يوضح الشكل 5 أنه حتى عندما لا يمكن تضمين 50٪ من القواعد في جدول التدفق، فإن طريقتنا لا تزال تحقق أداء أفضل مما لو كانت وحدة التحكم لا تعرف شيئاً عن جدار الحماية.

نُظهر في الشكل 5 نتائج دراسة الحالتين الآتيتين:

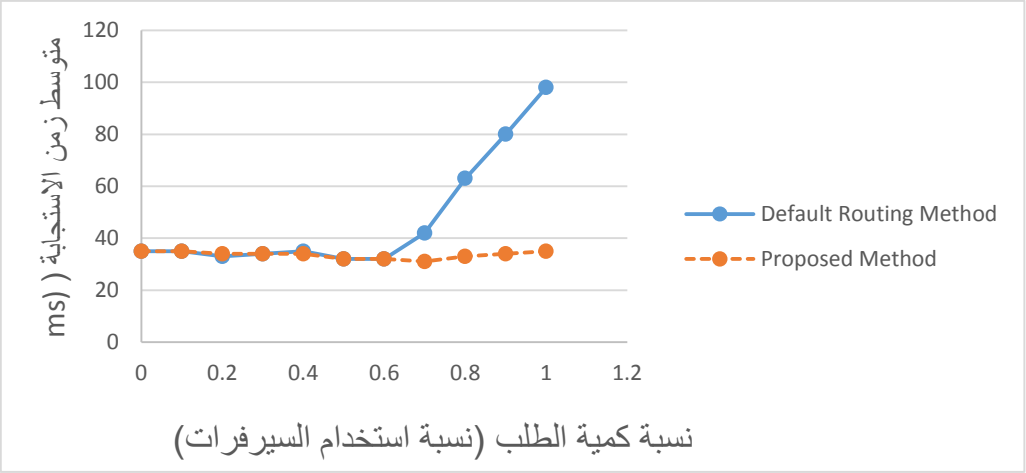
- الحالة العادية "Normal" استخدمنا فيها توزيعاً موحداً بمتوسط 0.8 وتباين 0.03 لاستخدام الوصلة.
- حالة الانحراف "Skew"، استخدمنا توزيع (Zipf) مع متوسط 0.8، يشير نوع الانحراف "skew" إلى وجود بعض الخوادم تحت حمل أخف، وبالتالي فإن هذه الخوادم مفضلة للغاية.

يعمل النظام المقترح بشكل أفضل تحت نوع تحميل الخادم "Normal" لأن جميع الخوادم تحت عبء ثقيل، لذلك تسيطر حالة الشبكة على الاختيار.



الشكل 5 توضيح احتمالية تقديم خدمة أفضل في خوادم عالية الاستخدام

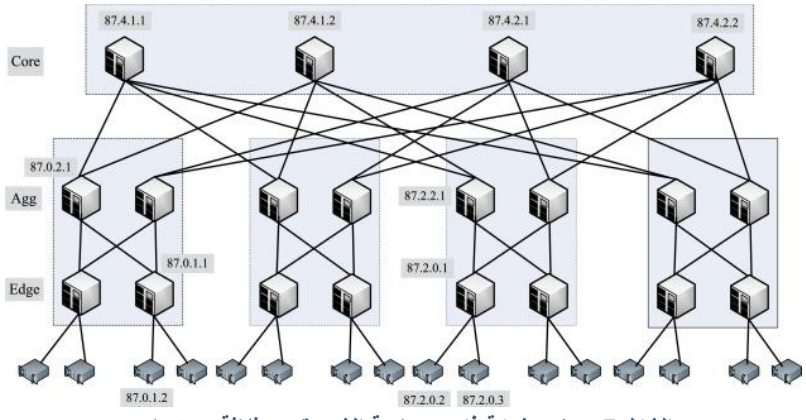
بينما يوضح الشكل 6 متوسط زمن الاستجابة تحت أنواع مختلفة من الطلبات في نفس معدل وصول الحزمة. عندما يزداد الطلب على عرض النطاق الترددي، أي احتمال زيادة الازدحام، يظل أداء النظام المقترح جيداً في حين أن موازنات الحمل الأصلية ذات مسارات التوجيه الافتراضية تكون ذات أداء ضعيف.



الشكل 6 يوضح مقارنة بين أنواع طلبات مختلفة بالنسبة لزمن الاستجابة

1.6 موازن الحمل

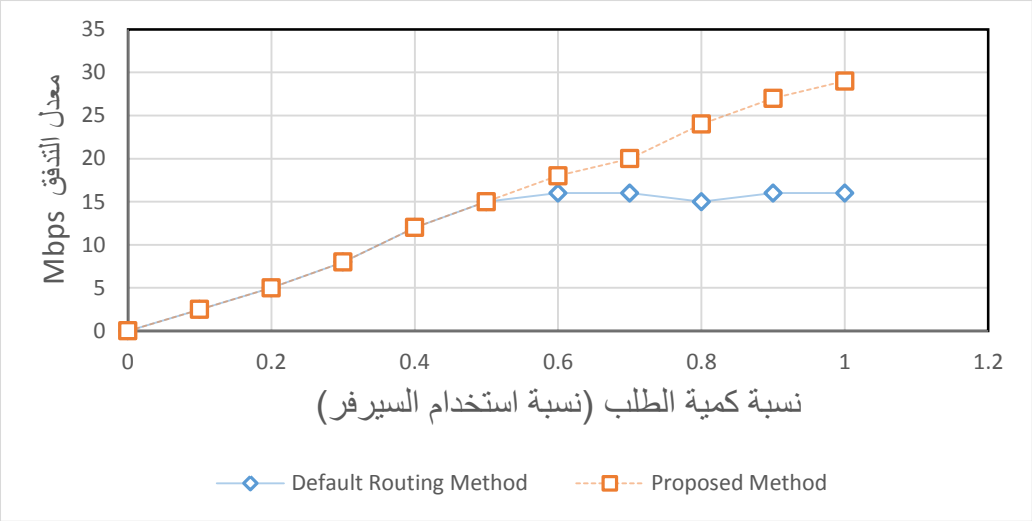
أنشأنا شجرة من ثلاثة مستويات باستخدام (Mininet) تتكون من 20 مبدلاً و16 مضيفاً موضحة في الشكل 7، ولكن الإصدار 2.2 من (Mininet) يدعم فقط سعة تصل إلى 1 جيجابيت في الثانية لذلك ببساطة خفضنا عرض النطاق الترددي لكل وصلة. لقد حددنا النطاق الترددي من المبدل الأساسي إلى المبدل التجميعي إلى 200 Mbps، ومن مبدل التجميع إلى الحافة إلى 100 Mbps وضبط النطاق الترددي 50 Mbps بين مبدلات الحافة والمضيفين. ونظراً لتقليل سعة الوصلة، صُممَ معدل خدمة الخوادم بحيث يكون 1500 طلب في الثانية، لضمان عدم سيطرة حالة الشبكة على زمن الاستجابة. شغلنا (iPerf) على كل مضيف لتوليد عبء العمل في الخلفية بشكل عشوائي.



الشكل 7 يوضح شبكة ذات معمارية الشجرة من ثلاثة مستويات

وركزنا في هذه التجربة على ثلاثة مقاييس أساسية: احتمال العثور على خادم أفضل لخدمة الطلب ومتوسط وقت الاستجابة والإنتاجية. هناك العديد من العوامل التي يمكن أن تؤثر على النتائج. تم اختيار ثلاثة أنواع رئيسية لتقييمنا: نوع تحميل الخادم (الذي يشير إلى ما إذا كان تحميل الخادم زائداً)، ومعدل وصول الحزمة (معدل الطلب)، ونوع الطلب (الذي يمكن أن يكون عرض النطاق الترددي مرتفعاً أو مكثفًا في الحساب أو مختلطًا). قارنا الطريقة المقترحة بالطريقة الأصلية المستخدمة في وحدة التحكم.

يوضح الشكل 8 معدل التدفق تحت أنواع مختلفة من الطلبات وتواجه طريقة موازنة الحمل الأصلية عنق الزجاجة عند معدل التدفق 15 ميغابت في الثانية حيث إنها تحدد الخادم فقط ولا تختار مسار التوجيه، وفي الوقت نفسه فإن النهج المقترح يؤدي إلى سلوك ثابت على المدى المقاس.



الشكل 8 مقارنة أنواع طلبات مختلفة بالنسبة لمعدل التدفق

References:

[1] Huang M, Liang W, Xu Z, Guo S. *Efficient Algorithms for Throughput Maximization in Software-Defined Networks With Consolidated Middleboxes*. IEEE Transactions on Network and Service Management. 2017;14(3):631-645.

[2] Feamster N, Rexford J, Zegura E. *The road to SDN*. ACM SIGCOMM Computer Communication Review. 2014;44(2):87-98.

[3] Nunes B, Mendonca M, Nguyen X, Obraczka K, Turletti T. *A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks*. IEEE Communications Surveys & Tutorials. 2014;16(3):1617-1634.

[4] McKeown N, Anderson T, Balakrishnan H, Parulkar G, Peterson L, Rexford J et al. *OpenFlow: enabling innovation in campus networks*. ACM SIGCOMM Computer Communication Review. 2008;38(2):69-74.

[5] Ferro G. ◎ SDN Use Case: Firewall Migration in the Enterprise [Internet]. EtherealMind. 2021 [cited 10 July 2020]. Available from: https://etherealmind.com/sdn-use-case-firewall-migration-in-the-enterprise

[6] Shrivastava G, Kaushik P, K.Pateriya R. *Load balancing strategies in software defined networks*. International Journal of Engineering & Technology. 2018;7(3):1854..

[7] HANDIGOL, N.; BRANDON H.; VIMAL K.; JEYA K.; BOB L.; AND NICK M. "*Reproducible network experiments using container-based emulation.*" In Proceedings of the 8th international conference on Emerging networking. 2012;8:253–264

[8] mininet/mininet [Internet]. GitHub. 2021 [cited 5 December 2019]. Available from: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

[9] Python Release Python 3.9.0 [Internet]. Python.org. 2021 [cited 20 December 2019]. Available from: https://www.python.org/downloads/release/python-390/