تحسين موازنة الحمل في الشبكات المعرَّفة برمجياً

م. نزيه احمد حرفوش[1]

د.م محمد مازن محايري[2] د.م رؤوف حمدان[2]

الملخص

أصبح تصميم وإدارة شبكات الحاسوب أكثر إبداعاً خلال السنوات القليلة الماضية بمساعدة الشبكات المعرّفة برمجيًا (SDN)، على الرغم من أنَّ هذه التكنولوجيا ظهرت فجأةً لكنها في الواقع جزء من تاريخ طويل لمحاولة جعل شبكات الحاسوب أكثر قابلية للبرمجة، وتعتبر الشبكات المعرَّفة برمجيّاً (Software Defined Networks SDN) مهيأة لتغيير التعقيد في تصميم وإدارة الشبكات من خلال تقديم واجهة نظيفة ومفتوحة بين كل من الأجهزة التي تعمل على الشبكة والبرامج التي تتحكم فيها، حيث توصَف بأنها طريقةُ ربطٍ شبكيٌ يكون فيها مستوى التحكم مفصولاً عن العتاد الصّلب، على خلاف التجهيزات الشبكية التقليدية حيث يتواجد مستوي التحكم ومستوى البيانات في نفس الجهاز، وتُمنح مسؤولية التحكم في الشبكات المعرفة برمجياً إلى تطبيق برمجي يسمَّى (المتحكِّم) (Controller) والذي يتم اعتباره كنظام تشغيل الشبكة (Network Operating System) أي أنه تم نقل التحكم من تحكم موزع إلى تحكم مركزي. يملك المتحكم رؤية كاملة عن طوبولوجيا الشبكة التي يديرها وبالتالي يكون هو المسؤول عن توجيه حركة المرور وموازنة الحمل في الشبكة ككُل.

أظهرت الشبكات المعرَّفة برمجيّاً (SDN) فوائد عديدة في نواح كثيرة مقارنة بالشبكات التقليدية. ومع ذلك، فإنَّ التوجيه غير الكفوء لحركة تدفق البيانات في الشبكات المعرَّفة برمجيّاً (SDN) يؤثر على الكفاءة ويسبب في بعض الأحيان حالات اختناق عنق الزجاجة، وبالتالي يؤثر توزيع الحمل غير المتكافئ في شبكات (SDN) بشكل كبير على أداء الشبكة، لذلك لجأ العديد من الباحثين إلى تطبيق تقنيات ساكنة أو ديناميكية لتحقيق موازنة الحمل (Load Balancing) بهدف تحسين كفاءة شبكات (SDN). نقدِّم في هذه الورقة آلية لتحسين موازنة الحمل على حركة تدفق البيانات واختيار الطرق الأفضل لتوجيه الحزم ضمن الشبكة، وتعتمد الآلية المُقترحة على تصميم وتنفيذ آلية توجيه خفيفة الوزن وفعالة من حيث الكلفة، تم استخدام المحاكي (Mininet) لتنفيذ الشبكة وربطها مع المتحكم (HPE VAN) وتم تنفيذ الخوارزمية باستخدام (Python) وإجراء التجارب باستخدام تعليمات الأداة (iperf).

الكلمات المفتاحية: الشبكات المعرفة برمجيا، موازنة الحمل، متحكم (SDN)، OpenFlow، Mininet، Miniedit،Iperf .

1 طالب دكتوراه في هندسة الحواسيب وشبكاتها – كلية الهمك – جامعة دمشق

2 مدرس في قسم الحواسيب والأتمتة – كلية الهمك – جامعة دمشق.

Improving the Load Balancing in Software Defined Networks

Eng. Nazeeh Harfoush[(1)]
Dr. Mohammad Mazen Mahayri[(2)] **Dr. Raouf Hamdan[(3)]**

Abstract

Computer networks design and management has become more innovative throughout the past few years with the aid of Software Defined Networks (SDN). Although this technology had a sudden outbreak, it is actually part of a long history in attempt to make computer networks programmable. SDN for short, are ready to de-complicate computer network design and management through presenting a clean and open interface between devices which are run on networks and the programs controlling these devices. The best describe of SDN is a network connection method in which the control plane is separated from the hardware, unlike traditional network equipment in which control and data planes co-exist in the same device. The control plane was abstracted to a software application, called the Controller, which is considered as network operating system. Subsequently, control plane has been centralized. Furthermore, the Controller has a comprehensive view of full topology of the network that it runs; thus, it is responsible for traffic routing and load balancing of entire network.

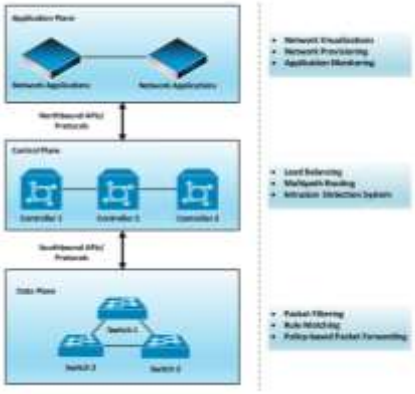
SDN have shown great advantages in many respects in contrast to traditional networks. However, the distribution of traffic routing in SDNs negatively affects efficiency and offers many challenges. For example, unequal load distribution gravely affects network performance, so many load balancing techniques have been used to improve SDN efficiency. this paper is a presentation of a mechanism for load balancing in traffic routing inside networks. This mechanism is an application of light and cost-efficient traffic routing. The Mininet emulator was used for execution of the network and connecting it to the Controller HPE VAN. The algorithm was applied using python, experiments were done using commands of iperf tool.

Keywords: Software defined networks, load balancing, Controller, Mininet, Miniedit, Iperf.

[(1)] PhD student in Department of Computer and Automation Engineering, Faculty of Mechanical and Electrical Engineering, Damascus University.
[(2)]Professor in Department of Computer and Automation Engineering, Faculty of Mechanical and Electrical Engineering, Damascus University.
[(3)] Professor in Department of Computer and Automation Engineering, Faculty of Mechanical and Electrical Engineering, Damascus University.

المقدمة:

تعمل شبكات (SDN) على تغيير طريقة تصميم الشبكات وإدارتها وذلك عن طريق سمتان محددتان، أولاً: فصل مستوى التحكم (Control plane) عن مستوى البيانات (Data plane) حيث إن مستوى التحكم هو المسؤول عن اتخاذ قرارات توجيه حركة المرور بينما يقوم مستوى البيانات بإعادة توجيه حركة المرور وفقًا للقرارات التي يتخذها مستوى التحكم، ثانياً: بعد أن كان مستوى التحكم موزعاً على كافة أجهزة التوجيه في الشبكات التقليدية تم دمج مستوى التحكم في شبكات (SDN)، بحيث يتحكم نظام تحكم مركزي بعدة عناصر شبكية تعمل في مستوى البيانات. ويقوم مستوى التحكم في شبكة (SDN) بالتحكم مباشرةً بعناصر مستوى البيانات (أي أجهزة التوجيه والمبدلات والصناديق الوسطى الأخرى) عبر واجهة برمجة التطبيقات الجنوبية (Southbound API) المعرَّفة بشكل جيد كما هو موضح في الشكل (1)، كما ويعتبر بروتوكول (OpenFlow) مثالاً بارزاً على واجهة برمجة التطبيقات الجنوبية (Southbound API).



الشكل (1) بنية الشبكة المعرفة برمجياً (SDN Architecture)

يوضح الشكل 1 بنية الشبكات المعرفة برمجيا حيث تتكون هذه البنية من ثلاث طبقات (من الأعلى للأسفل): طبقة التطبيقات (Application plane) وهي المسؤولة عن استعراض حالة الشبكة وإعطاء تقارير عنها، وطبقة التحكم(Control Plane) المسؤولة عن تحديد مسار الحزم (packets) وتطبيق القيود عليها، ويتم التحكم بها بواسطة مهندس الشبكة، وطبقة البيانات (Data plane) والتي تحتوي على طبقة التنفيذ وأجهزة الشبكة المرتبطة بطبقة التنفيذ.

حين يطلب المستخدم الاتصال بجهة ما، يقوم جهاز الشبكة بالتواصل مع متحكم الشبكة لتحديد المسار المخصص لحزم البيانات وتطبيق القيود عليها، ثم تتم عملية الاتصال.

الفصل بين طبقتي التحكم والبيانات استدعى وجود بروتوكول ينظم التواصل بين الطبقتين، ولأن الشبكات المعرفة برمجياً تم اعتمادها من قبل منظمة (Open Networking Forum (ONF)) والتي تتكون من مجموعة من الشركات العالمية، كان لزاماً أن يتم الاتفاق على بروتوكول لإدارة التعامل بين طبقة التحكم وطبقة البيانات (البنية التحتية).

يقوم بروتوكول (Open Flow) بتحديد مسار الرزم بناء على قواعد محددة مسبقاً بواسطة مهندس الشبكة. إضافة إلى ذلك، يحدد البروتوكول الوظيفة المناسبة (Action) كأن يقوم المُبَدّل (Switch) بتمرير رزمة البيانات (Forward)، أو تجاهلها (Drop).

بمقارنة الشبكات المعرفة برمجياً بالشبكات التقليدية: يمكن أن يتم التفريق بين النوعين في أن الشبكات المعرفة برمجيًا يمكن تهيئتها أثناء عمل الشبكة بشكل أسهل، إضافة إلى ذلك فإن ميزة التحكم المركزي (والتي تعتبر من مقومات نجاح هذا النوع من الشبكات) تساعد على تقليل تكلفة تشغيل وإدارة الشبكة. بالإضافة إلى ذلك عند تصميم شبكة (SDN) يجب التركيز على الخدمات التي يقدمها مركز التحكم بالشبكة، بصرف النظر عن أنواع ومصنّعي

المبـدلات المسـتخدمة فـي الشـبكة، وذلـك لأن تبـادل معلومـات حالـة الشـبكة واتخـاذ قـرارات التوجيـه سـيكون مركـزي ويقـع علـى عـاتق الواجهـة الجنوبيـة تسـليم هـذه القرارات لكل أجهزة الشبكة.

أما الشبكات التقليدية: يصعب تهيئتها في وقت عملها، حيث تتطلب إيقاف الشبكة وقت التهيئة، أيضاً على صعيد الـتحكم، فالشـبكات التقليديـة يجـب تهيئـة كـل جهـاز علـى حدا، ومن ناحية اكتشاف الأخطاء، فهذه معضلة الشبكات التقليدية حيث إن اكتشاف الأخطاء يتطلب وقتاً طويلاً.

الدراسات السابقة

إنَّ إدارة الشـبكة المعرفـة برمجيـاً عمليـة بسـيطة وفـي نفس الوقت تحقق الابتكار والتطوير في شبكات الحاسوب إلـى جانـب ذلك، فإن حقيقة أن المتحكم يملك رؤية شاملة ولحظيـة لمـوارد الشـبكة إلـى جانـب المعرفـة بمتطلبـات تطبيقات موازنـة الحمل كل ذلك يجعل الشبكات المعرفة برمجيا بيئـة مناسبة لتحقيـق موازنـة الحمل. نتيجـة لـذلك، كانت تقنيات موازنـة الحمل حاسمة لشبكات (SDN) لتحسين أدائها في العديد من مناهج توجيه حزم البيانات [1]. كما تم توظيفها لتخصـيص مـوارد الشـبكة بشـكل فعـال مـن أجـل التحسـين الشـامل لكـل مـن أداء الشـبكة وجـودة الخدمـة (QoS) [2]. لـذلك، يمكـن تحسين الأداء العام من خـلال اسـتخدام تقنيـات موازنـة الحمـل [3،4،5]. يتـيح مـوازن الحمل القائم على شبكات (SDN) التحكم في العديد من أجهزة الحاسوب، بالتالي ضمان إمكانية وجود شبكة أكثر مرونـة كما وتضـمن القدرة علـى التكوين المباشـر والإدارة المحسّنة للشبكة من أجل تقديم خدمات للتطبيقات بشكل أكثـر مرونـة وفعاليـة [6]. علـى الـرغم مـن أن الحوسـبة والتخزين قد شهدا تقدمًا في المحاكاة الافتراضية والأتمتة، إلا أن الشبكات الحاسوبية تخلَّفت عن هذا الرُّكب لحين ابتكـار نمـوذج شـبكات (SDN) حيـث إنَّ اسـتخدام نمـوذج شبكات (SDN) يُمكّن الشبكة الحاسوبية أن تعمل بنفس مبدأ عمل نماذج الحوسبة السحابية والتخزين الافتراضي بالإضـافة إلـى ذلـك تسـاعد موازنـة الحمـل فـي شـبكات (SDN) علـى استكشـاف أفضـل طريـق وأفضـل تطبيـق لتسليم الطلبات بشكل أسرع [7].

تم تقديم أول تطبيق لموازنة الحمل في شبكات (SDN) على أساس مبدل (OpenFlow switch) في الورقة [8]. في هذه البنية، يتم استخدام عنوان إنترنت (IP) ثابت في كل خـادم جنبًا إلـى جنـب مـع محـاكي خـادم الويـب الـذي يعمـل علـى منفـذ معـين، ويقـوم المـتحكم (NOX) بـإدارة قائمة الخوادم نظرًا لأنه يتم تقديم طلب جديد من العميل، يقوم مبدل (OpenFlow switch) بفحص جدول التدفق (flow table) بحثًا عن إدخالات مطابقة.

كما تم تطوير وحدة موازنة حمل مكتوبة بلغة (C++) على وحدة التحكم (NOX) لتوفير خدمات موازنة الحمل ومع ذلك، تسمح هذه الدراسة لوحدة تحكم واحدة فقط بإدارة عمليات التبديل (forwarding). تم تقديم خوارزميتين من خوارزميات موازنة الحمل – الثابتة والديناميكية – للجدولة، المصـممة علـى بنيـة شـبكات (SDN) فـي الورقـة [9]، أظهرت نتـائج التجـارب أن كفـاءة الخوارزميـة الديناميكيـة أعلى من كفاءة الخوارزمية الثابتة. أما في الورقة [10]، قـام المؤلفـون بتنفيـذ ومقارنـة إسـتراتيجية موازنـة الحمـل الدائري مع إستراتيجية عشوائية تم تنفيذها بالفعل باستخدام مبدل (Open Flow switch) متصل بوحدة تحكم (POX).

عبد اللطيف وآخرون [5] اقترحوا خدمة موازنة الحمل لخادم السحابة (SBLB) التي تعمل على شبكات (SDN) لتحسين كفـاءة المـوارد وتقليـل وقـت الاسـتجابة، مكونـات التقنيـة المقترحـة هـي تطبيـق يعمـل علـى مـتحكم (SDN) ومجموعـات مـن الخـوادم التـي تسـتخدم مبـدلات (OpenFlow switches)، يتكـون إطـار العمـل

(Framework) المقترح من قبلهم من وحدة إدارة الأجهزة، ووحدة موازنة الحمل الديناميكية، ووحدة التحكم (المتحكم)، ويتعامل المتحكم مع جميع الاتصالات، ويتحكم في تجمعات المضيفين، ويحافظ على شحن المضيف في الوقت الفعلي وتؤكد النتائج التجريبية كفاءة المخطط المقترح. كما قامت ساره [11] وآخرون بتطبيق تقنية موازنة الحمل وجدار الحماية على متحكم (Floodlight) الذي يتحكم بمركز بيانات يعمل وفق الطوبولوجيا (fat tree) وأظهرت النتائج تحسناً في معدل (round trip time) بمقدار 9%.

في هذه الورقة سيتم تصميم آلية لموازنة حمل حركة تدفق الحزم في مراكز البيانات عن طريق التحكم في التوجيه لاستغلال وإشغال الوصلات التي تملك أحمال خفيفة بغية تخفيف الحمل على الوصلات ذات الأحمال المرتفعة وتطبيق الآلية المقترحة على الشبكات التي تستخدم طوبولوجيا (fat tree) والشبكات التي تستخدم طوبولوجيا (full-mesh) وتسجيل ومقارنة النتائج ضمن كل طوبولوجيا وإظهار تأثير الطوبولوجيا المستخدمة على كل من حركة تدفق الحزم وموازن الحمل المُقترح وذلك باستخدام المتحكم (HPE VAN) لإدارة شبكة مركز البيانات، تم التحكم في توجيه الحزم عن طريق تطبيق مكتوب بلغة (python) يعمل على طبقة التطبيقات ويقوم بالاتصال بشكل مباشر مع متحكم الشبكة من خلال الواجهة الشمالية (Northbound API).

بروتوكول (OpenFlow) والمتحكم (HPE VAN) والمحاكي (Mininet)

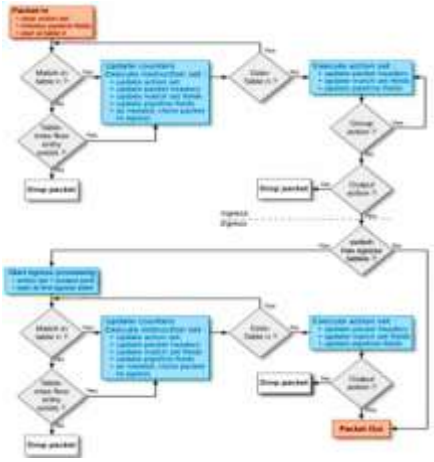
بروتوكول (OpenFlow) هو بروتوكول اتصالات يتيح عملية التواصل ما بين مستوى البيانات ومستوى التحكم في الشبكات المعرفة بالبرمجيات. يسهّل (OpenFlow) اتخاذ قرارات التوجيه وإعادة توجيه القواعد (rules) مثل موازنة الحمل والأمن وجودة الخدمة (QoS) من خلال المتحكم ليتم دفعها إلى الأجهزة في مستوى البيانات. ويعتبر بروتوكول (OpenFlow) هو البروتوكول المعتمد بشكل شائع للاتصال بين أجهزة الشبكة ومتحكم (SDN) ويمثل واجهة برمجة التطبيقات الجنوبية [12]. تم إصدار النسخة المعتمدة لبروتوكول (OpenFlow) لأول مرة من قبل مؤسسة الشبكات المفتوحة (Open Network Foundation (ONF)، ومنذ ذلك الحين تم بدأ استخدام هذا البروتوكول من قبل شركات تكنولوجيا المعلومات والاتصالات (ICT) مثل (Google HP, Cisco,) [13] [14]. يتيح بروتوكول (OpenFlow) الوصول المباشر والتكوين لأجهزة الشبكة مثل أجهزة التوجيه والمبدلات، كمثال: يتيح الوصول إلى جدول التدفق، وتمرير التعليمات إلى المبدلات حول كيفية إعادة توجيه تدفقات الشبكة (Flows). بالإضافة إلى ذلك فان قابلية برمجة متحكم (SDN) يضمن إمكانية تنفيذ التغييرات على الشبكة بأقل تأخير.

هناك نوعان رئيسيان من المبدلات القائمة على (OpenFlow) وهي: (OpenFlow-hybrid) و(OpenFlow-only) [15]. تتيح مبدلات (OpenFlow-hybrid) تنفيذ كل من عمليات (OpenFlow) وعمليات التبديل العادية (normal switching) مثل عمليات تمرير الطبقة الثانية التقليدية (L2 forwarding) وعزل شبكات المنطقة المحلية الافتراضية (VLAN's) وعمليات توجيه الطبقة الثالثة (L3 Routing) (توجيه IPv4 وتوجيه IPv6) وإضافة وتعديل قائمة التحكم في الوصول (ACL) ومعالجة جودة الخدمة. بينما تسمح مبدلات (Open Flow-only) بعمليات (Open Flow) فقط مما يعني أن جميع الحزم تتم معالجتها بواسطة خط الأنابيب

(Open Flow pipeline)، ولا يمكن معالجتها بطريقة أخرى.

يحتوي المبدل الذي يدعم بروتوكول (Open Flow) على المكونات الثلاثة التالية: أولاً، جدول التدفق والذي تمت برمجته بواسطة متحكم (SDN) لتوجيه عمل المبدلات من أجل معالجة كل حزمة. ثانياً، القناة الآمنة والتي تُستخدم لتوفير الاتصال بين مستوى التحكم الذي تم فصله وباقي تجهيزات مستوى البيانات، والمكون الثالث هو برتوكول أمن طبقة النقل (TLS) الذي يتم استخدامه كقناة آمنة [16].

الفكرة الرئيسية التي يحقق إدارتها بروتوكول (Open Flow) هي مطابقة التدفقات الجديدة مع مُدخلات جداول التدفق [17] والتي كانت تمثل الحزم (Packets) وجداول التوجيه (Routing Tables) في الشبكات التقليدية، حيث يسمح بتوصيل أجهزة البائعين المختلفة مثل مبدلات (cisco, juniper, Huawei) بمتحكم مركزي واحد منطقياً [18]. ويعرض الشكل 2 العمليات الأساسية لبروتوكول (Open Flow) حيث البحث عن مطابقة حقول التدفق الذي تتم معالجته حالياً ضمن المبدل مع جداول التدفق الخاصة بنفس المبدل لتحديد كيفية تنفيذ الإجراء المرتبط بهذا التدفق في حال حدثت المطابقة.



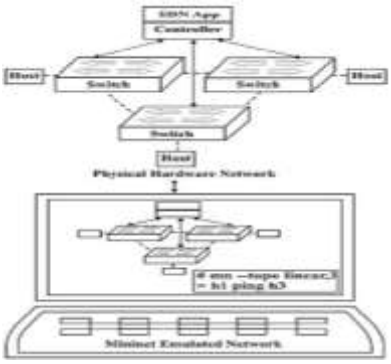
الشكل (2) مخطط العمليات الأساسية لبروتوكول (OpenFlow)

كما ويوفر متحكم (HPE Virtual Application Networks (VAN) نقطة تحكم موحدة في شبكة تعمل على نموذج (SDN)، مما يبسط الإدارة والتزويد والتزامن. يتيح ذلك تقديم جيل جديد من خدمات الشبكة القائمة على التطبيقات. كما أنه يوفر واجهات برمجة التطبيقات مفتوحة المصدر للسماح للمطورين بإنشاء حلول مبتكرة لربط متطلبات العمل ديناميكيًا بالبنية التحتية للشبكة عبر برمجيات مخصصة أو واجهات تحكم (RESTful) للأغراض العامة. تم تصميم (HPE VAN) للعمل في الحرم الجامعي أو مراكز البيانات أو بيئات مزودي الخدمة.

أما بالنسبة إلى المحاكي (Mininet) فهو المحاكي المُستخدم لنشر الشبكات الكبيرة على الموارد المحدودة والمؤلفة من جهاز حاسوب واحد بسيط أو على جهاز افتراضي، وقد أُنشئَ هذا المحاكي لتمكين البحوث في كل من الشبكات المعرفة برمجياً وبروتوكول (Open Flow) 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 العتاديَّة

(Hardware) السريعة والدقيقة ولكنها مكلفة للغاية، وبدلاً عن ذلك تم استخدام الخيار الآخر وهو المحاكاة (simulation) التي تكون رخيصة جداً ولكن في بعض الأحيان بطيئة، وتتطلب تعديل التعليمات البرمجية، وباستخدام (Mininet) تم ضمان سهولة الاستخدام والدقة في الأداء وقابلية التوسع.

الشكل 3 يوضح لنا محاكاة شبكة حقيقية على منصة (Mininet) التي تسمح بإنشاء طوبولوجيا ذات حجم كبير جداً تصل إلى آلاف العقد (Nodes) وإجراء الاختبار عليها بسهولة جداً [19]، ولديها أدوات بسيطة لتنفيذ الأوامر السطريَّة (command line) كما تتضمن واجهة برمجة تطبيقات (API)، وتسمح للمستخدم إنشاء وتخصيص وتبادل واختبار الشبكات المعرفة برمجياً بسهولة.



الشكل (3) محاكاة شبكة حقيقية على منصة Mininet

يمكن بسهولة إنشاء طوبولوجيا مخصصة باستخدام (Mininet) [20]. على سبيل المثال إنشاء طوبولوجيا مخصصة تحتوي على مبدل عدد/2/ وخمسة مضيفين يحتاج فقط إلى كتابة بضع أسطر من كود بايثون (Python)، وبالمثل يمكن بسهولة إنشاء شبكة معقدة جداً وتكون مرنة وقوية، كما ويمكن تكوين هذه الطوبولوجيا استناداً إلى البارامترات التي ستمرر إليها، وإعادة استخدام هذه الطوبولوجيا في تجارب متعددة.

التصميم

تم إنشاء تصميمين مختلفين للشبكة من حيث الطوبولوجيا، ومن ثمَّ إجراء التجارب وتسجيل نتائج التجارب على كل طوبولوجيا قبل إجراء أي تعديلات على عمل المتحكم (المتحكم يعمل بالحالة الافتراضية) وتمَّ إعادة نفس السيناريو على كل طوبولوجيا بعد تطبيق الآلية المقترحة من أجل تسجيل ومقارنة النتائج، تمَّ إنشاء الطوبولوجيا المخصصة لكل تصميم باستخدام الأداة (Miniedit) المرفقة مع (Mininet) وهي أداة مكتوبة باستخدام واجهة برمجة تطبيقات بايثون (Python API)، وتساعد هذه الأداة على إنشاء طوبولوجيا معقدة بشكل بسيط ومرئي.

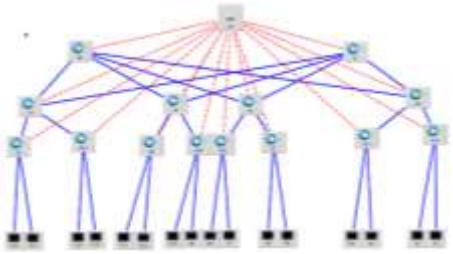
تم استخدام جهاز حاسب واحد لتشغيل التصميمين المقترحين وتنفيذ التجارب، الحاسب المُستَخدم يعمل بوحدة معالجة (Intel Core i5-4210U CPU @ 1.70GHz 2.40 GHz) وذواكر وصول عشوائي (16.0 GB).

ملاحظة: تمَّ تحويل نمط جميع المبدلات عن طريق المتحكم من (hybrid-mode) إلى (pure OpenFlow mode)، كما تمَّ استخدام وصلات من النوع (TC) لوصل جميع الأجهزة، وتقليل عرض نطاق الوصلات المستخدمة من اجل عرض المشكلة بشكل أوضح والتي كانت ستحدث مع وصلات ذات عرض نطاق أعلى في بيئة عمل حقيقية.

1.4 الطوبولوجيا fat-tree:

في هذا التصميم تم استخدام طوبولوجيا (fat-tree) حيث تُعتبر هذه الطوبولوجيا مناسبة للاستخدام في الحوسبة السحابية ومراكز البيانات بسبب قابليتها للتوسع بشكل سهل وذلك بتكرار طوبولوجيا الشجرة من ثلاث طبقات وضمها الى الطوبولوجيا المستخدمة دون الحاجة الى أعادة عنونة الأجهزة بشكل كامل بالإضافة الى ذلك

تتميز هذه الطوبولوجيا بإمكانية تحقيق التنصيف الكامل لعرض الحزمة على كامل العقد الموجودة ضمن المجموعة الواحدة (مبدلات التوزيع والحافة المتصلة مع بعضها بشكل مباشر). الشكل 4 يوضِّح التصميم الأول للشبكة المقترحة والتي تتألف من مبدلين رئيسيين (core switches) من نوع (OpenVswitch) الخاص بشبكات (SDN) يشكلان طبقة النواة (core)، وتليها طبقة التوزيع (Distribution) التي تتألف من 4 مبدلات (Open Vswitch)، كل منها موصول مع مبدلي طبقة النواة، ومن ثم طبقة الحافة (edge) التي تتألف من 8 مبدلات (Open Vswitch) المسؤولة عن تقديم خدمات التوصيل لكل من مضيفي ومُخدِّمات الشبكة وفي السيناريو المُقتَرَح تمَّ إنشاء 16 جهاز موصولين على مبدلات طبقة الحافة وتمَّت تسمية كل منهم بالحرف h مع الرقم التسلسلي له. ويتم التحكم بكامل الطوبولوجيا المقترحة عن طريق المتحكم C0 وهو من نوع (HPE VAN).

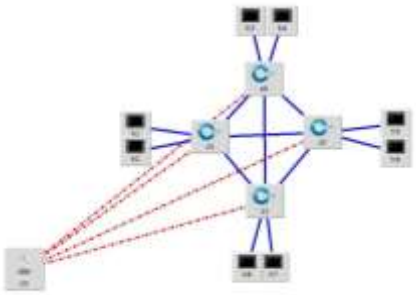


الشكل (4) الطوبولوجيا fat-tree المستخدمة في التصميم الأول

في هذا السيناريو يُمثِّل المضيفان (h16) و(h15) مُخدِّمين لنقل الملفات والويب من نوع (FTP) و(HTTP) على الترتيب أما باقي الأجهزة باستثناء (h1) و(h7) تقوم بزيادة الحمل على الشبكة بشكل طبيعي عبر اتصالها بمخدِّمَي نقل الملفات والويب. ويُمَثِل كل من (h7) و(h1) جهازي هاتف يعملان بتقنية نقل الصوت عبر برتوكول الإنترنت VOIP (Voice over IP). تمَّ تنفيذ السيناريو السابق باستخدام تعليمات الأداة (Iperf) التي تدعمها الآلة الافتراضية التي يعمل عليها (Mininet).

2.4 الطوبولوجيا Mesh:

في هذا التصميم تم استخدام طوبولوجيا (full-Mesh) حيث تُعتبر هذه الطوبولوجيا مفضلة من حيث الموثوقية وتم إجراء التجارب على هذه الطوبولوجيا لمقارنة عمل متحكم (SDN) والآلية المقترحة سواء كانت الشبكة ضخمة وتعمل على مستوى مركز بيانات أو كانت الشبكة صغيرة تعمل على مستوى مؤسسة صغيرة. الشكل 5 يوضِّح التصميم الثاني للشبكة المقترحة والتي تتألف من 4 مبدلات من نوع (Open Vswitch) الخاص بشبكات (SDN) وفي السيناريو المُقتَرَح تمَّ إنشاء 8 أجهزة موصولين على المبدلات وتمَّت تسمية كل منهم بالحرف h مع الرقم التسلسلي له. ويتم التحكم بكامل الطوبولوجيا المقترحة عن طريق المتحكم C0 وهو من نوع (HPE VAN).



الشكل (5) الطوبولوجيا mesh المستخدمة في التصميم الثاني

التنفيذ والنتائج

تمَّ استخدام الأداة (iperf) وهي أداة مكتوبة بلغة بايثون تقوم بإرسال حزم (TCP) و(UDP) بين الأجهزة، قمنا بتشغيل تعليمات (iperf) على جميع الأجهزة الموصولة في كل تصميم من التصميمين وذلك من اجل زيادة حركة المرور ضمن الشبكة ولضمان توليد حمل كافٍ لإيصال الوصلات لحالة الاختناق، الجدول 1

يوضح التعليمات التي تم تشغيلها على كل جهاز من أجهزة الشبكة.

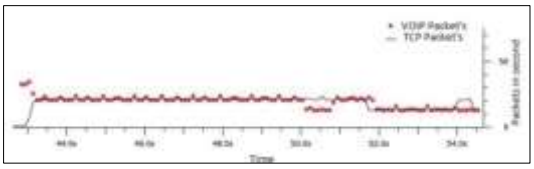
كما هو موضح في الجدول 1، تقوم الأجهزة (h2, h3, h4, h5, h6, h8, h9, h10, h11, h12, h13, h14) بسحب ملفات مختلفة من المُخدِّم (h15) وطلب صفحات ويب من المُخدِّم (h16) مسببةً بذلك زيادة الحمل على الشبكة من خلال زيادة حركة مرور الحزم على الوصلات التي يتم اختيارها تلقائيا من قِبَل المتحكم، وفي نفس الوقت يُجري كل من (h1,h7) مكالمة صوتية باستخدام تقنية (VOIP).

وتمَّ تطبيق السيناريو السابق على التصميم الثاني باستخدام وتنفيذ نفس التعليمات على جميع أجهزة التصميم الثاني.

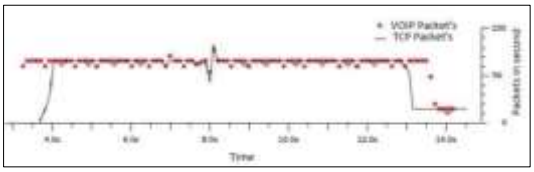
الجدول (1) يوضح تعليمات iperf المطبقة على كل جهاز ضمن طوبولوجيا fat-tree

اسم الجهاز	التعليمة المستخدمة	السبب
h15, h16	**Iperf -s**	يعمل الجهازين كمُخدِّمين من نوع **FTP** و **HTTP**
h2, h3, h4, h5, h6, h8, h9, h10, h11, h12, h13, h14	**Iperf -c 10.0.0.15, Iperf -c 10.0.0.16**	تعمل هذه الأجهزة كمضيفين ويسببون الحمل على الشبكة
h7	**iperf -s -u --len 300 -tos 184 -fk**	محاكاة جهاز هاتف يعمل بتقنية **VOIP**
h1	**iperf -c 10.0.0.7 -u -c 10.0.0.7 --len 300 --bandwidth 67000 --dualtest --tradeoff --tos 184 -fk**	محاكاة جهاز هاتف يعمل بتقنية **VOIP**

المدة الافتراضية لتنفيذ السيناريو السابق على كل طوبولوجيا باستخدام (Iperf) هي 10 ثواني وهي مدة المقترحة لخنق جميع وصلات الشبكة في كلا التصميمين وإظهار حالة عنق الزجاجة طيلة العشر ثواني المقترحة، تمَّ القيام بالتجربة بشكل متكرر، الشكل 6 والشكل 7 يوضحان عدد الحزم التي تم نقلها خلال زمن تنفيذ التجارب على التصميم الأول والتصميم الثاني بالتسلسل حيث يمثل المخطط باللون الأسود عدد الحزم المنقولة باستخدام بروتوكولي (FTP, HTTP) بينما النقاط باللون الأحمر تمثل عدد الحزم المنقولة للمكالمة الصوتية التي تجري بين الأجهزة.

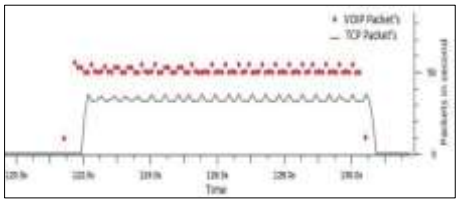


الشكل(6) مخطط نقل الحزم ضمن fat-tree قبل تعديل التوجيه

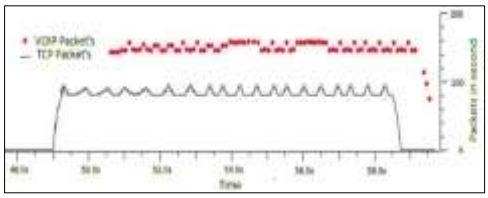


الشكل(7) مخطط نقل الحزم ضمن full-mesh قبل تعديل التوجيه

بعد تسجيل النتائج السابقة تمَّ تكرار التجربة ولكن بتطبيق آلية موازنة الحمل المقترحة من خلال تغيير الوصلات التي سيتم اختيارها وإعطاء الأولوية للتدفقات (flows) التي تحمل حزم بروتوكول (VOIP) وتغيير المسارات (Path's) للتدفقات الأخرى، وذلك عن طريق برنامج مكتوب بلغة بايثون يعمل على طبقة التطبيقات ويحدد للمتحكم المسارات التي يتم اختيارها لتوجيه التدفقات. الشكل 8 والشكل 9 يوضحان عدد الحزم التي يتم نقلها عبر الوصلات وذلك بعد تطبيق آلية موازنة الحمل على التصميم الأول والتصميم الثاني بالتسلسل.

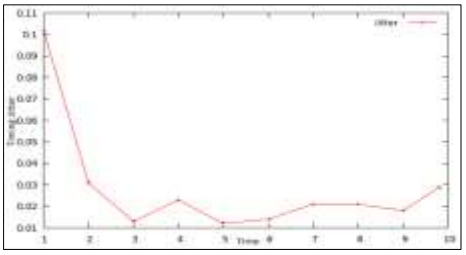


الشكل(8) مخطط نقل الحزم ضمن fat-tree بعد تعديل التوجيه

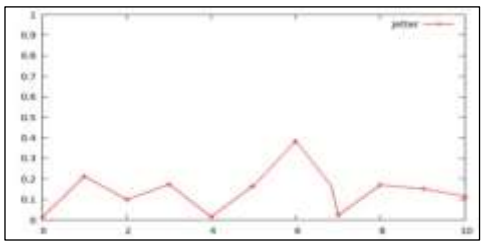


الشكل (9) مخطط نقل الحزم ضمن full-mesh بعد تعديل التوجيه

كما تمَّت مراقبة وتسجيل قيَم مُعامِل إضافي وهو التقلقل (الارتعاش) (Jitter) لتحديد جودة الخدمة على المكالمة الصوتية (VOIP)، الشكل 10 والشكل 11 يوضحان لنا مخطط الارتعاش الذي تتعرض له الحزم المنقولة طيلة مدة المكالمة خلال سلوك توجيه التدفقات المعدَّل للمتحكم ضمن التصميم الأول والتصميم الثاني على التوالي.

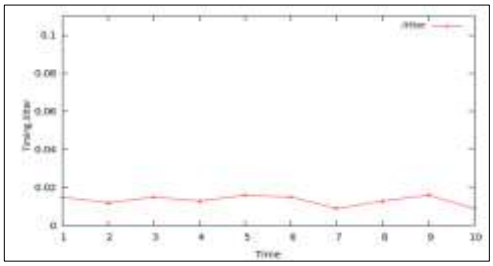


الشكل (10) مخطط الارتعاش (Jitter) في طوبولوجيا fat-tree قبل تعديل التوجيه (Routing)

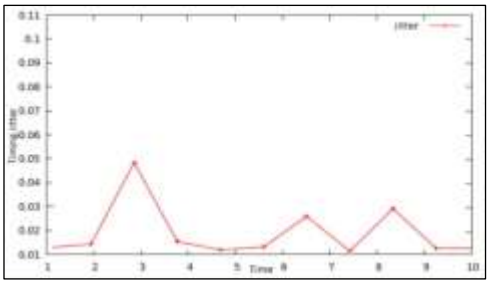


الشكل (11) الارتعاش (Jitter) في طوبولوجيا full-mesh قبل تعديل التوجيه (Routing)

بينما يوضح لنا الشكل 12 والشكل 13 مخطط الارتعاش الذي تتعرض له الحزم المنقولة طيلة مدة المكالمة خلال سلوك توجيه التدفقات المعدَّل للمتحكم ضمن التصميم الأول والتصميم الثاني على التوالي.



الشكل (12) مخطط الارتعاش (Jitter) في طوبولوجيا fat-tree بعد تعديل التوجيه (Routing)



الشكل(13) مخطط الارتعاش (Jitter) في طوبولوجيا full-mesh بعد تعديل التوجيه (Routing)

التحليل ومقارنة النتائج

عند مراقبة مدخلات جداول الدفق (flow tables) في مبدلات الشبكة في طوبولوجيا fat-tree ومقارنة المخططات الظاهرة في الشكل 6 والشكل 8 نلاحظ بأن السلوك الذي يتبعه المتحكم في توجيه التدفقات يسبب تأخير في وصول الحزم نتيجة الضغط الحاصل على

بعض الوصلات التي تم اختيارها بشكل افتراضي وهذه هي الحالة الافتراضية في شبكات SDN التي يتحكم في توجيه الحزم فيها المتحكم (HPE Van)، في التجربة الأولى قبل تعديل سلوك التوجيه المدة الكاملة هي 10 ثواني نلاحظ انخفض عدد الحزم الواصلة تقريبا إلى 20 حزمة في الثانية الواحدة بالنسبة لحزم بروتوكولي (FTP, HTTP) و في أفضل الحالات إلى 20 حزمة بالنسبة لحزم بروتوكول (VOIP) أي ما يعادل بالمجموع 40 حزمة في الثانية وهذا بسبب زيادة التحميل الحاصل على الوصلات المستخدمة، بالإضافة إلى ذلك فان مدة التجربة 10 ثواني نلاحظ التأخير الحاصل ولوصول كافة الحزم ليكون تقريباً 20 ثانية كما هو موضح في الشكل 6.

أمّا في التصميم الثاني الذي يستخدم طوبولوجيا full-mesh عند مراقبة مدخلات جداول الدفق (flow tables) في مبدلات الشبكة ومقارنة المخططات الظاهرة في الشكل 7 والشكل 9 نلاحظ أنَّ عدد الحزم الواصلة يتراوح بين 60 الى 70 حزمة في الثانية الواحدة بالنسبة لحزم بروتوكولي (FTP, HTTP) ومثلها بالنسبة لحزم بروتوكول (VOIP) أي ما يعادل بالمجموع 140 حزمة في الثانية الواحدة.

يقوم المتحكم في الحالة الافتراضية باختيار أول طريق يتم اكتشافه من قبل تطبيقين موجودين ضمن نظام تشغيل المتحكم وهما (Path Diagnostics, Path Daemon) حيث يتم قراءة رأس الحزمة واستخراج مصدر ووجهة الحزمة ومن ثم اكتشاف الطريق الواصل بينهما وذلك بعد وصول حدث (packet-in) من أول مبدل لا يجد أي حقل يطابق حقول رأس الحزمة مع مدخلات جدول التدفق الخاص بهذا المبدل ومن ثم إرسال حدث (packet-out) إلى جميع المبدلات المتواجدة على الطريق المكتشف بتوقيت كافي لتمرير الحزمة على الطريق المطلوب عن طريق بروتوكول (OpenFlow)، وتقوم المبدلات باستلام حدث (packet-out) وتحديث جداول التدفق لديها ومن ثم تمرير الحزمة إلى الهدف وعند انتهاء الوقت الممنوح من قبل المتحكم لهذه التدفقات (Flows) يقوم المبدل بحذفها من جداوله.

عندما استخدم المتحكم بالحالة الافتراضية هذه الإجرائية تجاهل استخدام الوصلات الأخف حمولة مسببا بذلك ضغط على الوصلات الموجودة على الطرق التي حددت من قبله وبالتالي نشأت هذه المشكلة.

أما بعد تطبيق آلية التوجيه المقترحة لموازنة الحمل و تخفيف الحمل عن الوصلات الأشد حمولة بتغيير المسارات إلى وصلات اقل حمولة نلاحظ أن زمن وصول الحزم بشكل كامل قد انخفض إلى عشر ثواني وعدد الحزم الواصلة في الثانية الواحدة ارتفع إلى 40 حزمة بالنسبة لحزم بروتوكولي (FTP, HTTP) بينما في بروتوكول (VOIP) ارتفع عدد الحزم الواصلة إلى 50 حزمة في الثانية الواحدة بالنسبة للتصميم الأول (fat-tree)، أمّا بالنسبة للتصميم الثاني (full-mesh) فإنَّ عدد الحزم الواصلة في الثانية الواحدة ارتفع إلى 100 حزمة تقريباً بالنسبة لحزم بروتوكولي (FTP, HTTP) بينما في بروتوكول (VOIP) ارتفع عدد الحزم الواصلة إلى 170 حزمة في الثانية الواحدة، وبمقارنة هذه النتائج نلاحظ التحسن الواضح في تقليل التأخير و زيادة الإنتاجية (throughput) من خلال توزيع الأحمال بشكل عادل و استغلال الوصلات الأقل حمولة كما يمكن ملاحظة الفرق الحاصل في من تغيير الطوبولوجيا سواء كان المتحكم يعمل بالطريقة الافتراضية أو بعد تطبيق الآلية المقترحة حيث تثبت التجربة أن أداء المتحكم يتغير بتغير طوبولوجيا الشبكة وفي كلا التصميمين زاد عدد الحزم المنقولة في الثانية الواحدة إلى الضعف تقريباً أمّا من ناحية جودة الخدمة نلاحظ التحسن الكبير في أداء

المكالمة الصوتية (VOIP) وذلك من خلال مقارنة الشكل 10 مع الشكل 12 والشكل 11 مع الشكل 13 حيث يُظهر الشكل 10 التقلقل في الإرسال من مرتبة 0.01 ميلي ثانية بينما ينخفض هذا التقلقل ليصبح من مرتبة 0.001 ميلي ثانية في الشكل 12 في نفس الطوبولوجيا (fat-tree) بعد تطبيق الآلية المقترحة لموازنة الحمل، كما ويُظهر الشكل 11 التقلقل في الإرسال من مرتبة 0.1 ميلي ثانية بينما ينخفض هذا التقلقل ليصبح من مرتبة 0.01 ميلي ثانية في الشكل 13 في نفس الطوبولوجيا (full-mesh) بعد تطبيق الآلية المقترحة لموازنة الحمل بالتالي نلاحظ أن مرتبة التقلقل افضل في طوبولوجيا fat-tree في حالتي العمل الافتراضية والمقترحة كما ونلاحظ انخفاض مرتبة التقلقل في التصميمين بعد تعديل آلية توجيه الحزم الافتراضية التي تعمل على المتحكم.

الخاتمة

الفكرة الرئيسية من هذا العمل هي تطبيق آلية للتدخل في التوجيه وتغيير نمط التوجيه الافتراضي لمتحكم الشبكات المعرفة برمجياً (HPE VAN SDN Controller) عن طريق قراءة بعض معاملات الشبكة عبر واجهة برمجة التطبيقات الشمالية والتدخل في قرارات المتحكم عبر الواجهة نفسها وذلك بغرض تخفيف الحمل عن الوصلات الأشد حمولة وتوزيع الحمل بشكل عادل بين جميع التجهيزات الشبكية التي تعمل في مركز البيانات، بعد تطبيق الآلية المقترحة نلاحظ التحسن الملحوظ في معدل نقل البيانات وانخفاض الارتعاش في مكالمات (VOIP)، وبالتالي موازنة الحمل في الشبكات المعرفة برمجيا موازنة ثابتة (Static Load Balancing) ولتحقيق الموازنة الديناميكية لا بد من تطبيق احدى خوارزميات الأمثلة على المتحكم بشكل داخلي (Internal).

Reference

[1] Kumari, P., Thakur, D., 2017. Load balancing in software defined network, in: International Journal of Computer Sciences and Engineering. volume 5, pp. (2017) 227–232

[2] Li, L., Xu, Q., 2017. Load balancing researches in SDN: A survey, in: Electronics Information and Emergency Communication (ICEIEC), 2017 7th IEEE International Conference on, IEEE. pp. 403–408.

[3] Zhou, W., Yang, S., Fang, J., Niu, X., Song, H., 2010. Vmctune: A load balancing scheme for virtual machine cluster using dynamic resource allocation, in: Grid and Cooperative Computing (GCC), 2010 9th International Conference on, IEEE. pp. 81–86.

[4] Salman, M.A., Bertelle, C., Sanlaville, E., 2014. The behavior of load balancing strategies with regard to the network structure in distributed computing systems, in: Signal-Image Technology and Internet-Based Systems (SITIS), 2014 Tenth International Conference on, IEEE. pp. 432–439.

[5] Akbar Neghabi, A., Jafari Navimipour, N., Hosseinzadeh, M., Rezaee, A., 2019. Nature-inspired meta-heuristic algorithms for solving the load balancing problem in the software-defined network. International Journal of Communication Systems 32, e3875.

[6] Kaur, P., Chahal, J.K., Bhandari, A., 2018. Load balancing in software defined networking: A review. Asian Journal of Computer Science and Technology 7, 1–5.

[7] Neghabi, A.A., Navimipour, N.J., Hosseinzadeh, M., Rezaee, A., 2018. Load balancing mechanisms in the software defined networks: a systematic and comprehensive review of the literature. IEEE Access 6, (2018) 14159–14178.

[8] Uppal, H., Brandon, D., 2010. OpenFlow based load balancing. CSE561: Networking Project Report, University of Washington, 2010.

[9] Zhang, H., Guo, X., 2014. SDN-based load balancing strategy for server cluster, in: Cloud Computing and Intelligence Systems (CCIS), 2014 IEEE 3rd International Conference on, IEEE. pp. 662– 667.

[10] Kaur, S., Kumar, K., Singh, J., Ghumman, N.S., 2015. Round-robin based load balancing in software defined networking, in: Computing for Sustainable Global Development (INDIACom), 015 2nd International Conference on, IEEE. pp. 2136–2139.

[11] Mohammed, Sarah & Jasim, Ammar. (2019). Evaluation of Firewall and Load balance in Fat-Tree Topology Based on Floodlight Controller. Indonesian Journal of Electrical Engineering and Computer Science. 17. 10.11591/ijeecs. v17. i3. pp 1157-1164.

[12] Lara, A.; Kolasani, A.; Ramamurthy, B. Network innovation using OpenFlow: A survey. IEEE Commun. Surv. Tutor. 2013, 16, 493–512.

[13] Wickboldt, J.A.; De Jesus, W.P.; Isolani, P.H.; Both, C.B.; Rochol, J.; Granville, L.Z. Software-defined networking: Management requirements and challenges. IEEE Commun. Mag. 2015, 53, 278–285.

[14] Jain, S.; Kumar, A.; Mandal, S.; Ong, J.; Poutievski, L.; Singh, A.; Venkata, S.; Wanderer, J.; Zhou, J.; Zhu, M.; et al. B4: Experience with a globally-deployed software defined WAN. ACM SIGCOMM Comput. Commun. Rev. 2013,43,3–14.

[15] Jammal, M.; Singh, T.; Shami, A.; Asal, R.; Li, Y. Software defined networking: State of the art and research challenges. Comput. Networks 2014, 72, 74–98.

[16] D. Kreutz, F. Ramos, P. Verissimo, C. Esteve, S. Azodolmolky and S. Uhlig, Software-Defined Networking: A Comprehensive Survey, IEEE, Vol. 103, No. 1, pp. 14-76, 2015.

[17] R. Peng, and Lei Ding, Research on Virtual Network Load Balancing based on OpenFlow, AIP, pp. 020014-1–020014, 2017.

[18] T. Assegie, A review on software defined network security risks and challenges,

ELKOMNIKA (Telecommunication, Computing, Electronics and Control)، Vol.17, No. 6, 2019.

[19] 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.

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