تقييم أداء المتحكمات المختلفة في الشبكات المعرفة برمجياً

د. أحمد صقر أحمد*

حازم ديب**

(تاريخ الإيداع 17 / 6 / 2020. قُبِل للنشر في 19/ 4 / 2021)

□ ملخّص □

الشبكات المعرفة برمجياً Software Defined Network (SDN) هي مفهوم شبكي حديث وهي أحد أكثر المجالات التي يعمل عليها الباحثون في أنحاء العالم في السنوات الأخيرة. يعتمد هذا النموذج بشكل رئيسي على الفصل بين طبقة التحكم (Control Plane) وطبقة البيانات (Data Plane)، فقد اقترحت الشبكات المعرفة برمجياً نقل الذكاء الخاص بالشبكة أي القرارات الواجب اتخاذها من أجهزة التبديل والتوجيه الشبكية إلى متحكم مركزي. تتصرف المتحكمات كنقطة تحكم للشبكة وتنظم عملية تدفق البيانات بين طبقة التطبيقات وطبقة البيانات من خلال واجهة التخاطب الشمالية (Northbound Interface) وواجهة التخاطب الجنوبية (Southbound Interface) وذلك لإنشاء شبكة أكثر ديناميكية وقدرة على التكيف. يتم اتخاذ قرارات التوجيه في المتحكم المركزي ويتم نقلها بعد ذلك إلى المبدلات والتي تقوم بتنفيذ القرار المنطقي المتخذ في المتحكم بشكل مباشر، ومن ذلك نجد أن المتحكمات شديدة الأهمية في نجاح الشبكات المعرفة برمجياً.

ناقشنا في هذا البحث الوظائف الأساسية لمتحكمات الشبكات المعرفة برمجياً، وخصائص هذه المتحكمات، كما قمنا بتقييم أداء عدد من هذه المتحكمات وفقاً لعاملين هما: الإنتاجية وزمن الانتظار، وتم حسابهما على أحجام مختلفة من الشبكات بزيادة عدد المبدلات والنياسب في كل شبكة.

الكلمات المفتاحية: الشبكات المعرفة برمجياً، المتحكم، الاستجابة، زمن الانتظار.

*أستاذ – قسم النظم والشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة تشرين – اللاذقية –سورية.

Email: ahmad.s.ahmad@tishreen.edu.sy

**طالب دكتوراه – قسم النظم والشبكات الحاسوبية – كلية الهندسة المعلوماتية – جامعة تشرين – اللاذقية – سورية.

Email: hazem.deeb@tishreen.edu.sy

Performance Evaluation of Different Controllers in SDN Networks

Dr. Ahmad Saker Ahmad*
Hazem Deeb**

(Received 17 / 6 / 2020. Accepted 19 / 4 / 2021)

□ ABSTRACT □

Software Defined Networking (SDN) is a new networking concept and is also the most discussed topic of networking technology in recent years. This approach is also characterized by the separation of the data and control planes. The SDN proposed an approach which involves in moving the network's intelligence out from the packet switching devices and putting it into a centralized logical controller. Controllers act as the control point for networks to manage flow between the Application layer and the Data layer through the Southbound API's and the Northbound API's to create a more flexible network. The forwarding decisions are primarily done in the controller and then moves down to the switches where it directly executes the logical decisions. Therefore, controllers are critical to the success of SDN networks.

In this paper, the basic functions and features of SDN controllers have been discussed, also the performance of multiple controllers has been evaluated on two metrics: Throughput and Latency which had been calculated on varying network sizes by increasing the number of switches and threads in each network.

Keywords: SDN, Controller, Throughput, Latency.

* **Professor, Department of Computer Networks & Systems, Faculty of Informatics Engineering Faculty, Tishreen University, Lattakia, Syria. Email: ahmad.s.ahmad@tishreen.edu.sy**

** **PhD Student, Department of Computer Networks & Systems, Faculty of Informatics Engineering Faculty, Tishreen University, Lattakia, Syria. Email: hazem.deeb@tishreen.edu.sy**

مقدمة:

إن المتحكم (في بعض الحالات ممكن أن يشار إلى المتحكم كنظام تشغيل شبكي) هو العنصر الرئيسي الذي يحدد نجاح أو فشل الشبكات المعرفة برمجياً لذلك كان من الضروري دراسة ومقارنة الأنواع المختلفة من المتحكمات الموجودة حالياً.

في الوقت الحالي نحن بعيدون عن متحكم قادر على فصل العتاد الصلب الخاص به عن لغة البرمجة التي يستخدمها، فالمتحكمات اليوم تعمل كمجموعة من التطبيقات المتجانسة وهي مرتبطة بشكل وثيق مع لغات البرمجة المكتوبة بها.

وبالنظر إلى أهمية المتحكم في هيكلية الشبكات المعرفة برمجياً كان لابد من المقارنة بين الأنواع المختلفة من المتحكمات وفقاً لعدد من العوامل التي سندرسها لاحقاً.

أهمية البحث وأهدافه :

تتطور الشبكات المعرفة برمجياً يوماً بعد يوم وتواجه عدداً من التحديات لعل أهمها القدرة على دعم التقنيات الجديدة التي تدخل بشكل متسارع إلى عالم الشبكات مثل انترنت الأشياء IoT (Internet of Things)، وقدرتها على دعم قابلية التوسع دون الحاجة إلى إجراء تعديلات كبيرة على بنية الشبكة والإعدادات الخاصة بها. وإلى يومنا هذا لم يتم تطوير متحكم مثالي لاستخدامه في جميع أنواع الشبكات المعرفة برمجياً، وتعد عملية اختيار المتحكم الأنسب عملية صعبة وتتبع لعدة عوامل أهمها طبيعة العمل المطلوب وبنية الشبكة، لذلك كان الهدف الرئيسي من هذا البحث هو المقارنة بين مجموعة من المتحكمات وفقاً لعوامل متعددة بغية الوصول إلى المتحكم الأفضل الذي نستطيع استخدامه في بحثنا لاحقاً.

طرائق البحث ومواده:

تم في هذا البحث التعرف على الوظائف الأساسية للمتحكم وعلى الخصائص المختلفة للمتحكمات في الشبكات المعرفة برمجياً، كما تم التعرف على عدد من أشهر المتحكمات الموجودة وتقييم أداءها وفقاً لعوامل مختلفة بواسطة المحاكي Mininet الذي يعمل على نظام تشغيل Linux والأداة Cbench، وتم التطرق إلى التجارب التي تم إجراؤها والنتائج التي تم الحصول عليها.

1- الوظائف الأساسية للمتحكم:

بما أن الشبكات المعرفة برمجياً تقوم بفصل طبقة التحكم عن طبقة البيانات فقد تم نقل ذكاء الشبكة بالكامل إلى المتحكم، حيث تتم هنالك كل عمليات الحوسبة واتخاذ القرارات، كما يمكن إضافة العديد من التطبيقات والمزايا في حال الحاجة إليها. يوجد العديد من الوحدات في الشبكات المعرفة برمجياً مثل: وحدة اكتشاف الوصلات، وحدة بنية الشبكة، وحدة التخزين، وحدة صنع القرار، وحدة جدول التدفق ووحدة التحكم بالبيانات، حيث تعد هذه الوحدات هي الوحدات الرئيسية في الشبكات المعرفة برمجيا ً[1]. تعد وحدتان من هذه الوحدات مسؤولة عن توفير خدمات التوجيه والتي هي وحدة اكتشاف الوصلات ووحدة بنية الشبكة. تقوم الوحدة الأولى بجمع المعلومات عن حالة كل وصلة في الشبكة، حيث يوجد لدينا نوعين من الوصلات في الشبكة: وصلات بين مخدمين ووصلات بين مستخدم طرفي ومخدم. النوع الأول يستخدم بروتوكول اكتشاف الوصلات Link Layer Discovery Protocol (LLDP) . تستخدم المعلومات التي تقوم بتوفيرها وحدة اكتشاف الوصلات لبناء قاعدة بيانات للعقد الجارة على مستوى المتحكم، وتستخدم قاعدة

البيانات من قبل الوحدة الثانية أي وحدة بنية الشبكة لبناء قاعدة البيانات العامة للشبكة ككل والتي تعتمد على إيجاد أقصر طريق رئيسي وأقصر طريق احتياطي لأي من العقد في الشبكة سواء كانت مبدلات أو أجهزة مضيفين. وإن أي تغيير في حالة أحد الوصلات يتم تعقبه من خلال وحدة اكتشاف الوصلات، لذلك يجب على وحدة بنية الشبكة أن تقوم بإعادة حساب كلفة الطرق بعد كل تعديل على قواعد بيانات الجيران المحلية.[2]

2- خصائص متحكمات الشبكات المعرفة برمجياً:

I. لغات البرمجة :

تتم برمجة المتحكمات باستخدام مجموعة من لغات البرمجة مثل Java و C و C++ و Python و Ruby ، في بعض الحالات يبرمج المتحكم بلغة برمجة واحدة فقط، بينما في حالات أخرى تستخدم عدة لغات لبرمجة المتحكم الواحد بحيث تمنحه أداءً أعلى ضمن ظروف عمل معينة. [3]

II. عابرة للمنصات:

إن التشغيل على عدة منصات يجعل عملية تعدد النياسب أسهل للفهم، حيث يعد الوصول السريع والإدارة الجيدة للذاكرة من الخصائص الأساسية للغات البرمجة. وعندما نريد اختيار متحكم محدد يجب أن نأخذ هذه العوامل بالحسبان لأنها تؤثر على أداء المتحكم وعلى سرعة التطوير أيضاً. تعد لغات البرمجة C++، Python وJava هي اللغات الأكثر استخداماً في برمجة المتحكمات. بشكل عام المتحكمات التي تستخدم لغة البرمجة Java تمتلك الخصائص التي تجعلها عابرة للمنصات وتمنحها مرونة كبيرة، بينما المتحكمات التي تستخدم لغة البرمجة C++ توفر أداءاً جيداً ولكنها أقل مرونة، بينما تقدم المتحكمات التي تستخدم لغة Python إدارة جيدة للذاكرة وواجهة تحكم رسومية جيدة للمستخدم ولكنها تعتبر ضعيفة في معالجة عملية تعدد النياسب. [4]

III. واجهات التخاطب الجنوبية:

توفر واجهات التخاطب الجنوبية بروتوكول اتصال بين طبقة البيانات وطبقة التحكم، وتستخدم من قبل المتحكم لإرسال إعدادات التهيئة ومداخل التدفق وتقوم بالتغييرات على قواعد التوجيه المستخدمة في أجهزة طبقة البيانات، وتقوم بنقل وظائف الشبكة إلى طبقة التحكم. يعد عدم التجانس بين المصنعين من أهم التحديات التي تواجه الواجهة الجنوبية، حيث يملك كل مصنع معماريته الخاصة مما يخلق مشاكل عدم تجانس والتي تقوم الواجهة الجنوبية بحلها من خلال توحيد البروتوكولات المستخدمة. يعد بروتوكول OpenFlow هو البروتوكول الأكثر استخداماً في هذه الواجهات، ولكن هذا البروتوكول هو ليس الوحيد المتاح والذي يتم العمل على تطويره. [5]

IV. واجهات التخاطب الشمالية:

تستخدم هذه الواجهات من قبل طبقة التطبيقات للتواصل مع المتحكم، وتعد الجزء الأكثر حساسية في بنية الشبكات المعرفة برمجياً. تأتي الأهمية العظمى من الشبكات المعرفة برمجياً من قدرتها على دعم وتشغيل مجموعة واسعة من التطبيقات المختلفة. تلعب الواجهة الشمالية دوراً أساسياً لمطوري التطبيقات، ويوفر واجهة تخاطب مشتركة بين المتحكم وطبقة الإدارة، كما يساعد في توفير المعلومات للأجهزة المستخدمة في تطوير التطبيقات بالشكل الذي يجعل الشبكات المعرفة برمجياً ديناميكية وسهلة التحكم. مؤخراً بدأت منظمة الشبكات المفتوحة ONF بنقل تركيزها على الواجهات الشمالية بعد أن أنهت عملها في توحيد معايير الواجهات الجنوبية عن طريق البروتوكول OpenFlow. فقد أنشأت مجموعة عمل للمطورين الذين يقومون بالعمل على برمجة الواجهات الشمالية وتطوير نماذج مبدئية وإنشاء معايير

موحدة. يعتبر بروتوكول نقل الحالة التمثيلية Representational State Transfer (REST) protocol حالياً هو الأكثر استخداماً في هذه الواجهات. [6]

V. دعم البروتوكول OpenFlow:

يعد البروتوكول OpenFlow هو البروتوكول الرئيسي في الشبكات المعرفة برمجياً، وقد كان أول الواجهات الجنوبية الموحدة التي تسمح بالتحكم بشكل مباشر بعمليات التوجيه التي تقوم بها طبقة البيانات في المبدلات الشبكية. يحدد البروتوكول OpenFlow عملية الاتصال بين المتحكم والمبدل، ويتألف من مجموعة من الرسائل التي يتم إرسالها من المتحكم إلى المبدل ومجموعة من رسائل الاستجابة بالاتجاه المعاكس بالشكل الذي يسمح للمتحكم ببرمجة المبدل لتوجيه تدفق البيانات الخاص بالمستخدم. عندما نريد اختيار المتحكم OpenFlow يجب علينا فهم وظيفة البروتوكول OpenFlow التي يدعمها المتحكم وخاصةً أننا في الطريق لتطوير نسخ أحدث من هذا البروتوكول. على سبيل المثال إن أحد الأسباب التي تدعونا لمعرفة هذه الوظائف أن النسخة 1.0 من البروتوكول OpenFlow لا تدعم IPv6 بينما تدعمها النسخة 1.3 والنسخ التي تليها. [7]

VI. قابلية برمجة الشبكات:

تعد قابلية برمجة الشبكات من أهم الفوائد التي نحصل عليها من شبكات SDN. ظهرت هذه الخاصية لمواكبة التطورات المتسارعة في مجال تكنولوجيا المعلومات، ويتم ذلك من خلال توفير قدر أكبر من التسيق والتنظيم بين عناصر الشبكة عن طريق السماح ببرمجة المتحكمات. تسمح هذه الخاصية بالتعامل مع التعقيدات غير المتوقعة التي تصادفنا في إدارة الشبكة خاصةً مع الازدياد الهائل في عدد الأجهزة المتصلة وتطوير خدمات جديدة. الطريقة التقليدية القديمة في إدارة الشبكات تستهلك كثيراً من الوقت وهي عرضة للعديد من الأخطاء وتقود إلى عدد كبير من التناقضات. يأتي النموذج الجديد ليساهم في إزالة هذه الصعوبات مقدماً الإمكانية الديناميكية في إدارة الشبكة. [8]

VII. الكفاءة :

كفاءة المتحكم هو مصطلح يستخدم للإشارة إلى المتحولات المتخلفة التي تستخدم في تقييم المتحكمات كالأداء والوثوقية وقابلية التوسع والأمان. يوجد عدة معايير لتقييم الأداء في المتحكم أهمها الإنتاجية وزمن الانتظار، حيث يلعبان دوراً أساسياً في نجاح أو فشل المتحكم. وبشكل مشابه يوجد معايير مختلفة تحدد الوثوقية وقابلية التوسع والأمان. معظم الأبحاث التي تقوم بمقارنة المتحكمات تعتمد بشكل رئيسي على معيار الأداء. وإضافةً إلى ذلك فإن وجود متحكم مركزي وحيد في شبكات SDN يعتبر تحدياً كبيراً من ناحيتي الوثوقية والأداء، لذلك بدأ الاتجاه إلى المفهوم الموزع للمتحكمات بدلاً من وجود المتحكم المركزي. [9]

3- أنواع المتحكمات المختلفة:

I. المتحكم Open Daylight :

إن المتحكم OpenDaylight هو مشروع مجتمعي مفتوح المصدر مكتوب بلغة Java هدفه الرئيسي تطوير شبكات SDN من خلال توفير مزايا ودعم بروتوكولات جديدة في مجال صناعة المتحكمات. يسمى هذا المتحكم في الوقت الراهن بمنصة OpenDaylight، وهو مفتوح المصدر بالنسبة للمستخدمين والزبائن موفراً بذلك منصة عمل مشتركة لأولئك الذين لديهم دافع واهتمام في تطوير شبكات SDN، ويتيح لهم إمكانية العمل معاً من أجل إيجاد حلول مبتكرة. يدعم المتحكم OpenDaylight البروتوكول OpenFlow كما يمكن أن يدعم بعض المعايير الأخرى الخاصة بشبكات SDN. يعتبر البروتوكول OpenFlow المعيار الأول في شبكات SDN والذي يحدد إمكانية الاتصال

المفتوح الذي يسمح للمتحكمات بالعمل مع طبقة البيانات ويسمح بإجراء تغييرات على بنية الشبكة. يسمح هذا البروتوكول لشركات الأعمال بأن تملك تحكماً كاملاً على الشبكات الخاصة بها ويعطي القدرة على التكيف مع التغييرات التي تحتاجها هذه الشركات بشكل جيد. يستخدم المتحكم OpenDaylight بشكل واسع في مجموعات متنوعة من البيئات فهو يوفر الدعم للعديد من المعايير والبروتوكولات المستخدمة في شبكات SDN، حيث أن التطبيقات المستخدمة في المتحكم OpenDaylight تستطيع أن تجمع المعلومات من الشبكة وتقوم بتحليلها بواسطة مجموعة من الخوارزميات التي يستخدمها هذا المتحكم وتقوم بإنشاء قواعد جديدة إلى الشبكة المستخدمة. [10]

II. المتحكم NOX:

يعتبر المتحكم NOX المكتوب بلغة C++ جزءاً من نظام شبكات SDN، والذي يوفر منصة صريحة لبناء تطبيقات التحكم في الشبكة. إن البروتوكول OpenFlow كان الأول من بين تقنيات SDN الذي يتيح عملية التعرف على الاسم الصريح وقد تم تطويره بشكل رئيسي في شركة NiciraNetworks لتقديم منصة للتحكم في الشبكات تؤمن واجهة برمجة عالية المستوى لإيجاد الحلول لمشاكل إدارة الشبكة بواسطة تطبيقات تحكم جديدة مما ساهم بتحويل مشاكل الشبكات إلى قضايا برمجية. منذ أن تبرعت شركة Nicira بمتحكم NOX لمجتمع الباحثين في العام 2008 فقد أصبح هذا المتحكم هو الأساس للكثير من الباحثين عندما بدؤوا بعمليات الاكتشاف الأولية لشبكات SDN. يهدف متحكم NOX إلى توفير منصة عمل للمطورين والباحثين لإعطائهم القدرة على تطوير تطبيقات جديدة والابتكار في مجال الشبكات الصناعية والتجارية. تحدد تطبيقات NOX بشكل رئيسي عملية توجيه التدفقات ضمن الشبكة كما تسهم في تقديم حلول أفضل لإدارة الشبكة. [11]

III. المتحكم POX:

إن المتحكم POX هو متحكم مفتوح المصدر مبرمج بلغة Python من أجل تطوير تطبيقات تحكم في شبكات SDN. في الفترة الأخيرة أصبح POX أكثر استخداماً من NOX، والذي يعتبر مشروعاً مماثلاً فهو يسمح بسرعة أكبر في التطوير وإيجاد النماذج المبدئية لشبكات SDN. يستخدم المتحكم POX البروتوكول OpenFlow أو البروتوكول OVSDB لتوفير بيئة عمل تسمح بالاتصال مع مبدلات شبكات SDN. يستطيع المبرمجون باستخدام لغة البرمجة Python استخدام المتحكم POX لإنشاء متحكم شبكات SDN، حيث يعتبر المتحكم POX أداة تستخدم لتعريف الناس بشبكات SDN كما يستخدم أيضاً لأغراض بحثية وبناء تطبيقات متعلقة بالشبكات. يمكن استدعاء مكونات POX بشكل مباشر من خلال سطر الأوامر حيث يتم تحقيق الوظيفة المطلوبة من الشبكة من خلال استخدام هذه المكونات. يمكن استخدام المتحكم POX كمتحكم رئيسي في شبكات SDN وذلك لأنه يتيح بسهولة إمكانية تحميل مكونات الشبكة. يستطيع المطورون إنشاء متحكمات أكثر تعقيداً من خلال استخدام مكونات جديدة أو من الممكن أن يقوموا بكتابة تطبيقات تستهدف واجهات برمجة التطبيقات بشكل مباشر. [12]

IV. المتحكم Ryu :

المتحكم Ryu هو متحكم SDN مفتوح المصدر مكتوب بلغة Python يستخدم لزيادة مرونة الشبكات من خلال تسهيل معالجة المهمات. يوفر متحكم Ryu مجموعة مختلفة من المكونات مع واجهة برمجية كاملة تسمح للمطورين بإنشاء طرق جديدة في إدارة الشبكة، وبناء تطبيقات تحكم مع إمكانية أسهل في الاتصال. تستخدم هذه المكونات أيضاً في عمليات التطوير التي تقوم بها المنظمات لتلائم احتياجاتها بالشكل الأمثل، فالمكونات الحالية على سبيل المثال يمكن تحقيقها وتعديلها بسهولة في الشبكات الحالية لتلائم الاحتياجات المتغيرة في التطبيقات المختلفة. يقدم متحكم

Ryu مزايا متوسطة مما يجعله خياراً مناسباً للأعمال الصغيرة والأغراض البحثية. وبما أن هذا المتحكم مكتوب بلغة Python فهو يقدم إمكانيات عالية لتطوير التطبيقات والوحدات الموجودة في شبكات SDN. ومع ذلك فإن افتقارها للمرونة وعدم قدرتها على العمل على منصات مختلفة تحد من استخدامها في السوق الحقيقية. سوف نحاول المقارنة بين أنواع المتحكمات هذه بالرغم من أنه لن يكون المعيار الوحيد الذي سوف يحدد عملية اختيار المتحكم بل إن الاستجابة العالية للطلبات مع أقل زمن انتظار تعتبر المفتاح الرئيسي عند اختيار أي متحكم. [13]

V. المتحكم Beacon:

المتحكم Beacon هو متحكم مفتوح المصدر مكتوب بلغة Java يدعم تقنية تعدد النياسب ومعالجة الأحداث وأعداداً كبيرة من منصات العمل بالإضافة إلى أنه يعتبر متحكم سريع نسبياً. تم بدء العمل على تطويره في بداية العام2010 وقد تم استخدامه في عدد من المشاريع والأبحاث. يملك القدرة على إدارة مركز بيانات لأشهر عديدة دون أن يتعرض للفشل. إن هذا المتحكم يعمل تحت معيار GPLv2 و Foss Exception v1.0 مما يسمح له بتنصيب الحزم وهو قيد التشغيل دون أن تتأثر باقي الحزم بذلك. يدعم المتحكم Beacon واجهات المستخدم ويمكن أن يتضمن مخدم ويب في خدماته. [14]

VI. المتحكم Floodlight :

هو متحكم مفتوح المصدر مرخص من قبل Apache يستخدم لغة البرمجة Java وهو مطور من قبل شركة Big Switch Networks. يستخدم هذا المتحكم بروتوكول OpenFlow لإدارة تدفق البيانات في بيئة شبكات SDN. متحكم Floodlight بسيط وسهل الاستخدام عند بدء التشغيل والعمل والصيانة، كما يستطيع أن يعمل مع كافة أنواع المبدلات سواء كانت فيزيائية أو افتراضية طالما تدعم هذه المبدلات البروتوكول OpenFlow. يعد المتحكم Beacon هو فرع من المتحكم Floodlight. [15]

VII. المتحكم MUL :

هو متحكم مكتوب بلغة C يدعم تقنية تعدد النياسب ويملك عدة مستويات من الواجهات الشمالية من أجل تشغيل عدد متنوع من التطبيقات، ويهدف هذا المتحكم حالياً إلى دعم النسخ الأحدث من الواجهات الجنوبية مثل البروتوكول OpenFlow بنسختيه 1.3 و 1.4. تم تصميم هذا المتحكم مع مراعاة معياري الوثوقية والأداء، ويعتبر هذا المتحكم سهل التعلم والتحقيق مما يجعله أكثر مرونة. [16]

VIII. المتحكم Maestro :

هو متحكم مكتوب بلغة Java لتحقيق تطبيقات التحكم في الشبكة يدعم العديد من الواجهات لتحقيق هذه التطبيقات التي تقوم بالتحكم بحالة الشبكة وإدارة الاتصالات بين الأجهزة المختلفة في الشبكة، كما يسمح بتحسين إنتاجية وأداء الشبكة من خلال دعم التفرع في شبكات SDN، ولكن هذا المتحكم يحتاج إلى قليل من الجهد لإدارة التفرع في الشبكة بما أن ذلك يتضمن القيام بالعمل المعقد في الشبكة كإدارة الحمل وجدولة العمليات والنياسب. [17]

IX. المتحكم ONOS :

هو متحكم مفتوح المصدر مكتوب بلغة Java عملت على تطويره شركة ON.LAB، تم تصميمه بشكل مخصص من أجل أن يتأقلم مع البيئات القابلة للتوسع، حيث يملك إتاحية عالية وقابلية للتوسع ويعمل كباقي المتحكمات في شبكات SDN على فصل طبقة التحكم عن طبقة البيانات. [18]

X. المتحكم IRIS:

هو متحكم SDN مكتوب بلغة Java يهتم بمعالجة مشكلات التوسع والإتاحية حيث يوفر بنية للتوسع بشكل أفقي مما يسمح بإضافة عدد من المخدمات بشكل ديناميكي بالشكل الذي يسمح بتحسين الأداء، كما يوفر إمكانية معالجة هرمية للشبكة مما يخفف من مشاكل الإدارة وكلفتها. [19]

XI. المتحكم Libfluid :

هو متحكم SDN مكتوب بلغة C++، ظهر أول مرة في مسابقة Open Networking Foundation عام 2014 وقد فاز بهذه المسابقة متفوقاً على 9 أنواع أخرى من المتحكمات وفقاً لعدد من المعايير التي تم اعتمادها في المسابقة. كان الهدف من تصميم هذا المتحكم أن يصبح المتحكم الافتراضي في بيئات تطوير شبكات SDN من خلال تصميم متحكم ذو أداء عالي وسهل الاستخدام من قبل المطورين. يوجد عدة أنواع من متحكمات Libfluid ولعل أهمها متحكمي Libfluid-Raw وLibfluid-Msg الذين يختلفان في طريقة معالجة الرسائل، حيث يقوم النوع msg بمعالجتها ضمن المتحكم نفسه، بينما يقوم المتحكم من النوع raw بمعالجة هذه الرسائل مباشرة في مكتبات لغة C++، ومن الملاحظ ان المتحكم raw يعطي نتائج أفضل من المتحكم msg وذلك بسب الفجوة بينهما وخاصةً عند الحاجة إلى توليد أغراض بلغة C++. [20]

النتائج والمناقشة:

1- السيناريو المقترح:

تعتبر Cbench هي الأداة الأكثر استخداماً عند مقارنة المتحكمات المختلفة في شبكات SDN. تعمل هذه الأداة بنمطي عمل بشكل رئيسي هما الإنتاجية Throughput وزمن الانتظار Latency وتستخدم الأداة Cbench هذين المعيارين لتقييم أداء المتحكمات المختلفة. في نمط الإنتاجية يتم إرسال أكبر عدد من الحزم التي يستطيع المتحكم معالجتها، أما في نمط زمن الانتظار تقوم الأداة Cbench بإرسال حزمة واحدة فقط وتنتظر الاستجابة لحساب الزمن الذي يستغرقه المتحكم لمعالجة حزمة واحدة. يمكن أن نقوم بتشغيل أداة Cbench والمتحكم المراد دراسته على جهاز واحد أو يمكن تشغيلهما على جهازين منفصلين. لقد اخترنا القيام بتشغيل الأداة والمتحكم على نفس الجهاز وذلك بسبب محدودية الوصلات الموجودة والتي تملك عرض حزمة أعظمي 10 Gbps. الأداة التي قمنا باختبار التجارب عليها تمتلك معالج Intel® Core™ i5-2430M CPU @ 2.40 GHZ. وذواكر بسعة 8 GB DDR3 بنظام تشغيل Windows 10 مع أداة افتراضية على برنامج Virtual Box تم تنصيب عليها نظام تشغيل Linux 18.01 – 64 bit . وتم تشغيل عملية التجربة بواسطة الأمر التالي :

./cbench –c localhost –p 6633 –l 14 –m 10000 –M 1000 –s 8 –t

حيث:

c- : المتحكم عن طريق اسمه أو عنوان ال IP الخاص به (Controller).

p-: رقم منفذ المتحكم (Port).

l- : عدد مرات التجربة (Loops per test).

m- : زمن الاختبار بالثانية .

M- : عدد العناوين الفيزيائية في كل مبدل.

s-: عدد المبدلات.

t- : نمط الإنتاجية .

2- الدراسات المشابهة:

هناك العديد من الدراسات التي قيمت أداء هذه المتحكمات، وقد حاولنا في بداية هذا البحث إجراء مقارنة لدراستنا مع بعض الدراسات الأخرى، ولكن واجهنا صعوبات تتعلق بأن جميع الدراسات الأخرى لم تدرس هذا العدد من المتحكمات واكتفت بالمقارنة بين 6 أنواع مختلفة من المتحكمات على الأكثر، وعند محاولة دمج دراستين أو أكثر لمقارنتهما مع بحثنا واجهنا صعوبات تتعلق باختلاف السيناريوهات والعوامل المستخدمة في كل دراسة، لذا اكتفينا بهذا العمل.

3- بارمترات المحاكاة:

I. الإنتاجية (Throughput):

تعرف الإنتاجية عادةً بأنها معدل معالجة طلبات التدفق من قبل المتحكم خلال واحدة الزمن، الشيء المهم والواجب ملاحظته هو أن عملية الحساب هذه لا تتعلق بمفهوم الوقت الذي تستغرقه الرسالة للوصول، ولكنها تتعلق بعدد الحزم في الرسائل المرسلة وما يقابلها من حزم في رسائل الاستجابة المستقبلة، أي بشكل أبسط، يمكن تعريف الإنتاجية بأنها عدد الاستجابات في واحدة الزمن، حيث تقوم المبدلات بإرسال أكبر عدد ممكن من الحزم دفعةً واحدةً بدون انتظار استجابة، وبعدها تتم عملية حساب عدد الاستجابات التي تتلقاها خلال واحدة الزمن المختارة، وغالباً ماتكون واحدة الإنتاجية هي: استجابة/ ثانية.[21]

II. زمن الانتظار (Latency):

هو الزمن الفاصل بين لحظة إرسال حزمة ما من المبدل إلى المتحكم وبين لحظة استقبال الاستجابة لهذه الواردة من المتحكم إلى المبدل. يمكن لعدد من العوامل أن تؤثر على زمن الانتظار ومنها وقت المعالجة الذي يستغرقه المتحكم في معالجة هذه الحزمة، بالإضافة إلى زمن التأخير في الوصلات بين المبدل والمتحكم. يقوم المبدل بإرسال حزمة واحدة فقط إلى المتحكم وينتظر الاستجابة لهذه الحزمة، بعدها يتم حساب الزمن الفاصل بين لحظة إرسال الحزمة ولحظة استقبال الاستجابة الخاصة بها، وغالباً تقدر واحدة زمن الانتظار بأجزاء من الثانية.[22]

4- تحليل النتائج:

تم حساب نتائج الاختبار لكل من المتحكمات المذكورة عن طريق الأداة Cbench ماعدا المتحكم OpenDaylight الذي يعاني من بعض المشاكل مع هذه الأداة حيث تم إجراء التجربة عليه باستخدام الأداة Wbench المكتوبة بلغة Python والمشابهة لـ Cbench.

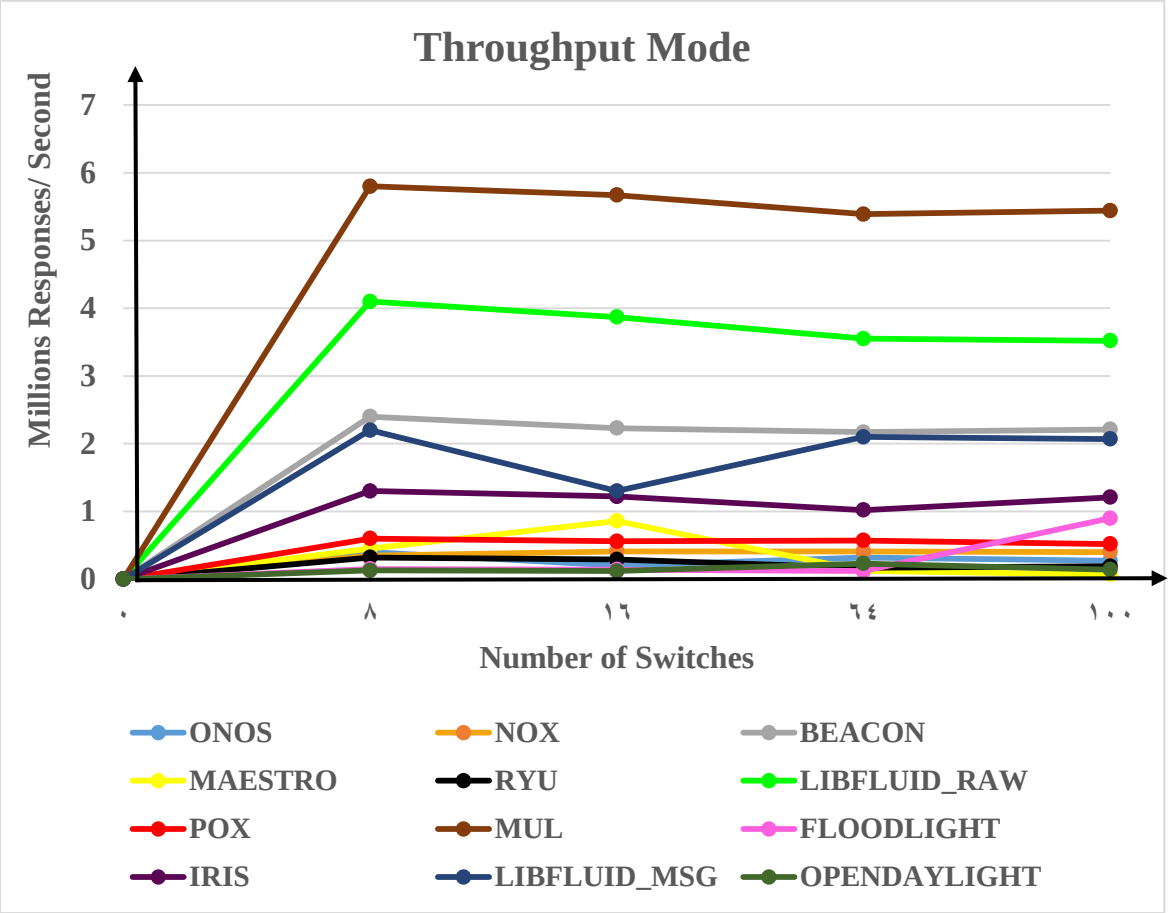
تم إجراء التجارب مع زيادة عدد المبدلات في كل من نمطي الإنتاجية وزمن الانتظار، كما تم إجراء تجربة ثالثة مع زيادة عدد النياسب Threads في كل متحكم وذلك في نمط الإنتاجية، وقد حصلنا على النتائج الموضحة بالأشكال 1-2-3.

	Programming language	Northbound API	Southbound API	Supported Platform	Modularity	Multithreading
Open Day Light	JAVA	REST, RESTCONF, XMPP, NETCONF.	Open Flow 1.0, 1.3	Linux, Mac OS, Windows.	High	Yes
Nox	C++	AD-Hoc	Open Flow 1.0	Linux	Low	Yes
Pox	Python	AD-Hoc	Open Flow 1.0	Linux, Mac OS, Windows.	Low	No
Ryu	Python	REST	Open Flow 1.0 – 1.5	Linux, Mac OS.	Fair	Yes
Beacon	Java	AD-Hoc	Open Flow 1.0	Linux, Mac OS, Windows.	Fair	Yes
Floodlight	Java	REST, Java, RPC, Quantum.	Open Flow 1.0,1.3	Linux, Mac OS, Windows.	Fair	Yes
Mul	C	REST	Open Flow 1.0, 1.3, 1.4	Linux	High	Yes
Maestro	Java	AD-Hoc	Open Flow 1.0	Linux, Mac OS, Windows.	Fair	Yes
ONOS	Java	REST, Neutron	Open Flow 1.0, 1.3.	Linux, Mac OS, Windows.	High	Yes
IRIS	Java	REST	Open Flow 1.0-1.3.	Linux	Fair	Yes
Libfluid	C, C++		Open Flow 1.0, 1.3.	Linux	Fair	Yes

الجدول (1): خصائص المتحكمات المختلفة في شبكات SDN

4-1- التجربة 1: زيادة عدد المبدلات في نمط الإنتاجية:

في التجربة 1 قمنا بزيادة عدد المبدلات من 1 وحتى 100 مبدل وقمنا بدراسة النتائج التي أظهرتها الأداة Cbench في نمط الإنتاجية كما هو موضح في الشكل (1).

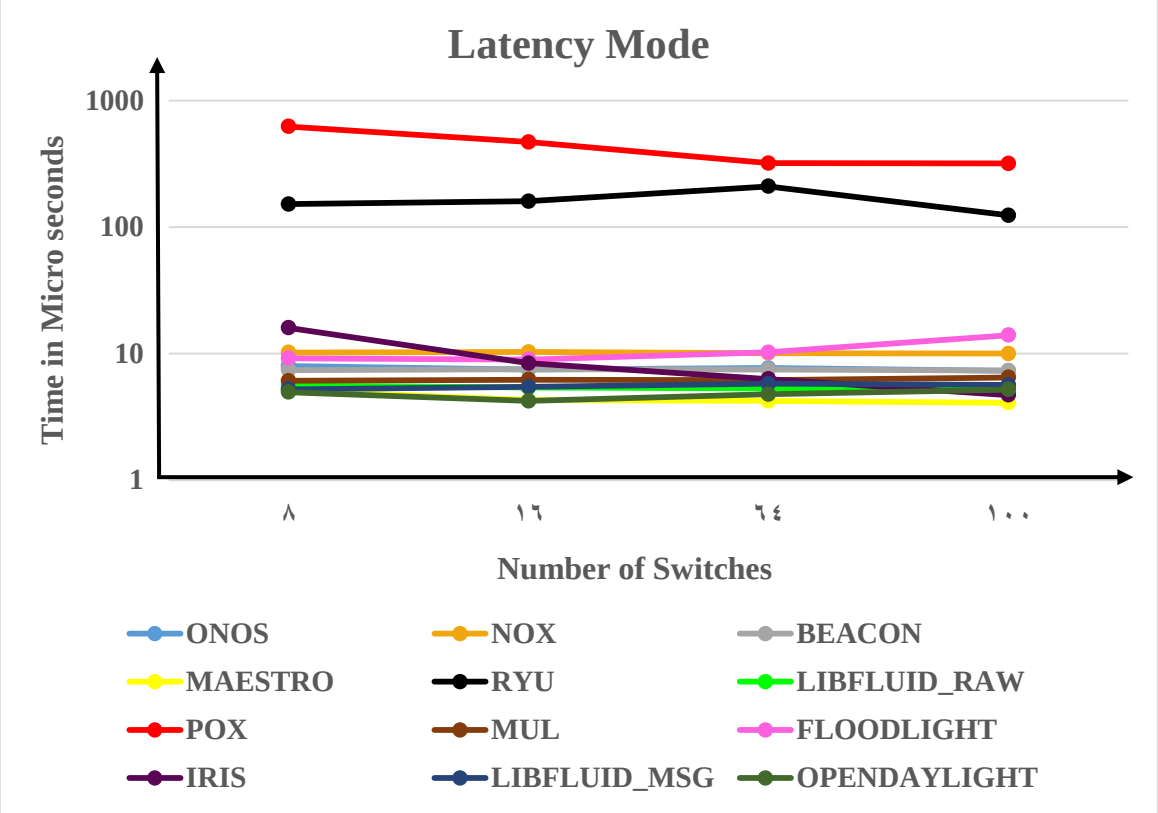


الشكل (1): زيادة عدد المبدلات في نمط الإنتاجية

تظهر النتائج في الشكل (1) أن المتحكم MUL يعطي أعلى إنتاجية من بين المتحكمات المدروسة، ثم يليه في المركز الثاني المتحكم Libfluid_RAW، ونلاحظ أن كلا هذين المتحكمين مكتوبات بلغة C، تعود هذه النتيجة إلى أن الحالة المدروسة هي حالة نيسب وحيد، حيث أن لغة C تعتبر الأفضل في المعالجة عند وجود نيسب وحيد. نجد في المراكز التالية المتحكمات Beacon و IRIS المكتوبان بلغة Java بالإضافة إلى المتحكم Libfluid_MSG المكتوب بلغة C++، بينما تعطي المتحكمات الباقية أداءاً متقارباً.

4-2- التجربة 2: زيادة عدد المبدلات في نمط زمن الانتظار:

في التجربة 2 قمنا بزيادة عدد المبدلات من 1 وحتى 100 مبدل وقمنا بدراسة النتائج التي أظهرتها الأداة Cbench في نمط زمن الانتظار كما هو موضح في الشكل (2).

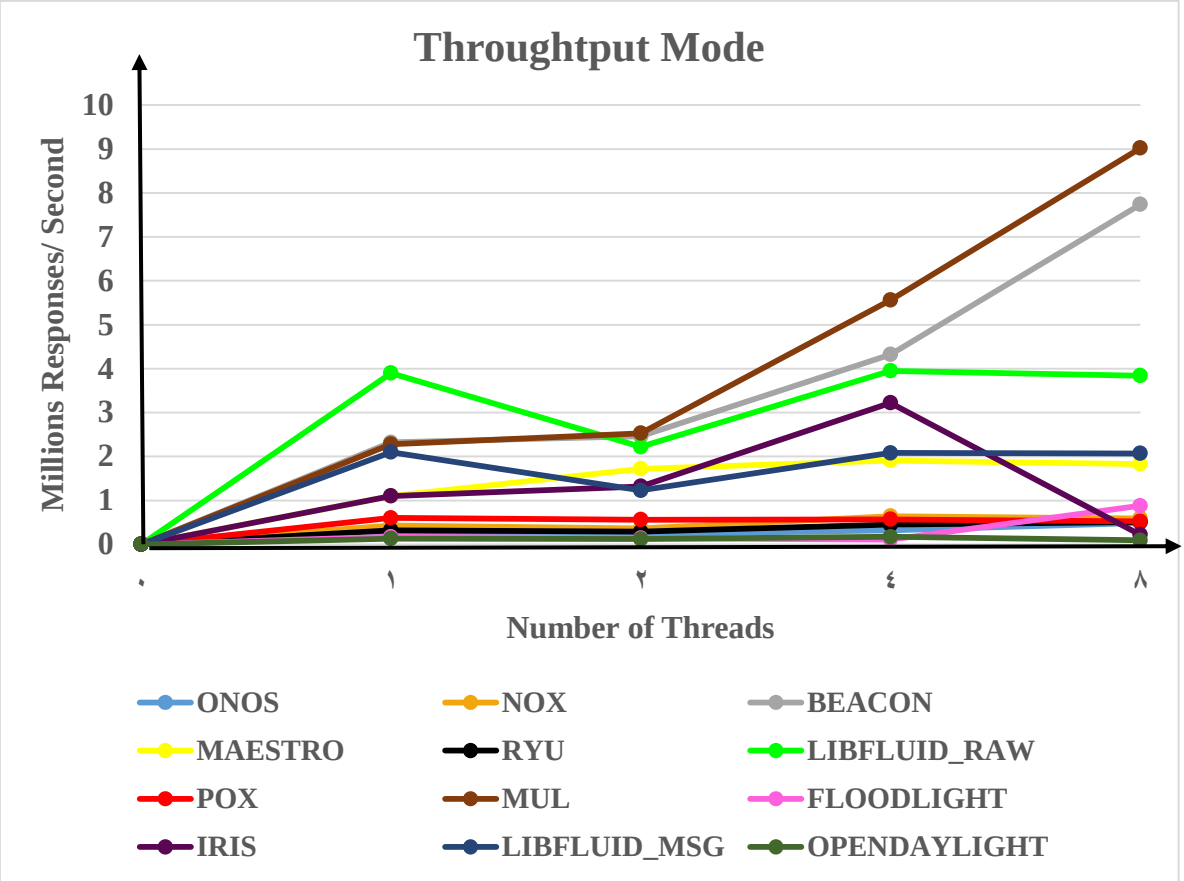


الشكل (2): زيادة عدد المبدلات في نمط زمن الانتظار

تظهر النتائج في الشكل (2) بأن المتحكم Maestro يعطي أفضل زمن انتظار بين المتحكمات الأخرى، يليه المتحكم OpenDayLight، ونلاحظ أن كلا هذين المتحكمين مكتوبات بلغة Java، حيث تدعم لغة Java تقنية تعدد المتحكمات أكثر من باقي اللغات. يأتي بعدهما متحكما Libfluid بنسختيه و متحكم Mul المكتوبان بلغة C، حيث تأتي لغة C في المركز الثاني من بين اللغات التي تدعم تقنيات تعدد المتحكمات. كما نلاحظ أن أسوأ نتيجة هي للمتحكمين POX و RYU المكتوبين بلغة Python، وذلك لأن هذه اللغة لا تدعم تقنيات تعدد المتحكمات بشكل جيد.

4-3- التجربة 3: زيادة عدد النياسب في كل متحكم في نمط الانتاجية:

في التجربة 3 قمنا بزيادة عدد النياسب في كل متحكم من 1 وحتى 8 نياسب وقمنا بدراسة النتائج التي أظهرتها الأداة Cbench في وضع الانتاجية كما هو موضح في الشكل (3).



الشكل (3): زيادة عدد النياسب في كل متحكم في نمط الانتاجية

في الشكل(3) نلاحظ أن المتحكم MUL يعطي أفضل نتيجة بين المتحكمات المدروسة يليه متحكم Beacon ، كما أن متحكمي Libfluid ومتحكم IRIS يعطيان نتائج مقبولة، بينما نلاحظ أن المتحكمات المكتوبة بلغة Python لا تعطي أي تحسن ملحوظ عند زيادة عدد النياسب وذلك لأن لغة Python بالأصل لا تدعم تعدد النياسب بشكل فعال.

يجب علينا توخي الحذر عند قراءة نتائج الاختبار هذه لأنه يوجد العديد من العوامل الأخرى المهمة التي يجب علينا أخذها بالحسبان، حيث أن المتحكمين ONOS و OpenDaylight يدعمان تعدد الأنماط وهو عامل مهم للغاية عند تطوير التطبيقات، كما يجب الأخذ بالحسبان نسخة بروتوكول OpenFlow التي نريد استخدامها، بالإضافة إلى أن بعض هذه المتحكمات لا تدعم بنية SDN الموزعة والتي نسعى لتطويرها وبالتالي لا يمكن استخدامها.

لذلك يجب أخذ الغرض والبيئة التي نريد استخدام المتحكم ضمنها بالحسبان وبعدها اختيار المتحكم الأنسب للوظيفة المطلوبة والبروتوكولات المستخدمة.

الاستنتاجات والتوصيات:

- في هذا البحث قمنا بدراسة عدد من المتحكمات وفقاً لعدد من العوامل باستخدام الأداة Cbench. لكن بسبب التنوع الكبير في تطبيقات شبكات SDN واستعمالات المتحكم فيها فإن عملية اختيار المتحكم الأنسب تعتبر معقدة نسبياً وتعتمد بشكل كبير على نوع التطبيق الذي نحتاج المتحكم فيه.

- ومن خلال التحليل والأبحاث وجدنا أن متحكم OpenDaylight هو المتحكم الأكمل نسبياً حيث أنه يقوم بدعم عدد كبير ومتنوع من التطبيقات والميزات الغير متوفرة بشكل كامل في باقي المتحكمات، كما أنه يعطي نتائج تعتبر جيدة في كافة مجالات التحليل مما يجعله ذو مستقبل واعد خصوصاً أنه يدعم مجالات الـ IoT المختلفة وذلك ما دفع البعض إلى تسميته متحكم الانترنت المستقبلي.

References:

[1] FEAMSTER, NICK, JENIFFER REXFORD, and ELLEN ZEGURA. *The road to SDN*. ACMQueue 11.12 (2013).

[2] DIEGO KREUTZ; FERNANDO M. V. RAMOS; PAULO VERISSIMO; SIAMAK AZODOLMOLKY and STEVE UHLIG, *Software-Defined Networking: A Comprehensive Survey* , Cs.ni, 8 oct 2014.

[3] CELLO TROIS; LUIS CARLOS ERPEN BONA; MARCOS DIDONET DEL FABRO, and MAGNOS MARTINELLO, *A Survey on SDN Programming Languages: Towards a Taxonomy*, IEEE Communications Surveys & Tutorials · April 2016.

[4] V R SUDARSANA RAJU, *SDN Controllers Comparison*, International Journal of Industrial Electronics and Electrical Engineering, Volume-6, Jul.-2018.

[5] ZOHAIB LATIF; KASHIF SHARIF; FAN LI; MD MONJURUL KARIM; SUJIT BISWAS, and YU WANG, *A Comprehensive Survey of Interface Protocols for Software Defined Networks*, J. Network Computer Applications (2020), Article 102563.

[6] POONAM VIJAY TIJARE, DEEPIKA VASUDEVAN,*THE NORTHBOUND APIs OF SOFTWARE DEFINED NETWORKS*, INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY, ISSN: 2277-9655, October2016.

[7]*Open Flow Tutorial*, 2020. [Online]. Available :https://github.com/mininet/openflow-tutorial

[8] MARK MITCHINER; REEMA PRASAD, *Software-Defined Networking and Network Programmability*: *Use Cases for Defense and Intelligence Communities*, 2019. [Online].Available: https://www.cisco.com/c/dam/en_us/solutions/industries/docs/gov/software_defined_networking

[9] ALEXANDER GELBERER; NIV YE MINI; RAN GILADI, *Performance Analysis of Software-Defined Networking (SDN)*, Conference: Proceedings of the 2013 IEEE 21st International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems.

[10] ZUHRAN KHAN KHATTAK; MUHAMMAD AWAIS;ADNAN IQBAL. *Performance Evaluation of OpenDaylight SDN Controller*, Proceeding of the 20th IEEE International Conference on Parallel and Distributed Systems (ICPADS), 16-19 Dec. 2014.

[11] *NOX detailed implementation*, 2019. [Online].Available: http://www.noxrepo.org

[12] *POX detailed implementation*, 2019. [Online].Available: http://www.noxrepo.org

[13] *NTT,NTT Laboratories OSRG Group*, 2019. [Online].Available: http://osrg.github.com/ryu/

[14] ERICKSON D. *The Beacon OpenFlow Controller*. HotSDN.ACM 2013.

[15] *Floodlight is a Java-based OpenFlow controller*, 2019. [Online].Available: http://floodlight.openflowhub.org

[16] *Open MUL SDN Platform*, 2019. [Online].Available: http://www.openmul.org/openmul-controller.html

[17] Z. CAI; A. COX; and E. T. S. NG, *Maestro: A System for ScalableOpenFlow Control*, Cs.Rice.Edu, p.10,2011. [Online]. Available: http://www.cs.rice.edu/eugeneng/papers/TR10-11.pdf
[18] U. KRISHNASWAMY; P. BERDE; J. HART; M. KOBAYASHI; P. RADOSLAVOV; T. LINDBERG; R. SVERDLOV; S. ZHANG; W. SNOW, and G. PARULKAR, *ONOS: An open source distributed SDN OS*, 2019. [Online]. Available: http://www.slideshare.net/umeshkrishnaswamy/open-network-operating-system
[19] BYUNGJOON LEE; SAE HYONG PARK; JISOO SHIN, and SUNHEE YANG. *IRIS: The OpenFlow-based Recursive SDN Controller*. MSIP (Ministry of Science, ICT & Future Planning), Korea in the ICT R&D Program 2013.
[20] *The libfluid OpenFlow documentation*, 2019. [Online].Available: https://opennetworkingfoundation.github.io/libfluid
[21] LUSANI MAMUSHIANE; ALBERT LYSKO, and SABELO DLAMINI, *A comparative evaluation of the performance of popular SDN controllers*, Wireless Days Conference 2018.
[22] LIEHUANG ZHU; MD MONJURUL KARIM; KASHIF SHARIF; FAN LI; XIAOJIANG, and MOHSEN GUIZANI, *SDN Controllers: Benchmarking & Performance Evaluation*, IEEE JSAC, 12 Feb 2019.