تحليل أداء وحدات تحكم الشبكات المعرفة برمجياً: POX وOpendaylight

الباحثة: م. ميساء الضاهر*

الملخص

لقد تناولنا في هذا البحث نوع جديد وحديث من الشبكات وهي الشبكات المعرَفة برمجياً والتي ابتُكرت لتجاوز السلبيات والتحديات التي تعاني منها الشبكات التقليدية الموجودة حالياً من خلال فصل ذكاء الشبكة عن أجهزة التبديل ووضعه على وحدات تحكم خارجية وهذا ما يوفر بيئة أبسط للبرمجة ولإدارة الشبكة وحرية أكبر للبرمجيات لتحديد سلوك الشبكة. يوجد حالياً العديد من تطبيقات وحدات التحكم المتوفرة، في هذا البحث قمنا بمقارنة أداء اثنين من وحدات التحكم مفتوحة المصدر وهي pox و opendaylight باستخدام محاكي mininet والأداة D-ITG من حيث متوسط تأخير التدفق والإنتاجية وأي من وحدتي التحكم كانت أكثر ثباتاً مع زيادة الحمل من خلال قياس معامل jitter في عدد من بُنى الشبكات الأساسية لتحديد وحدة التحكم ذات السلوك الأفضل.

الكلمات المفتاحية: الشبكات المعرَفة برمجياً، وحدة التحكم POX، وحدة التحكم Opendaylight، محاكي mininet ، الأداة D-ITG .

*م. ميساء الضاهر: قائم بالأعمال عضو هيئة فنية في كلية الهندسة المعلوماتية – جامعة البعث

Performance Analysis of Software Defined Networks Controllers: POX and Opendaylight

Abstract

In this research we have dealt with a new and modern type of networks, namely, Software Defined Networks, which were created to overcome the negatives and challenges faced by the existing traditional networks by separating network intelligence from switching devices and placing it on external controllers, and this provides a simpler environment for programming, network management and greater software freedom To determine network behavior. There are currently many controller applications available, in this paper we compared the performance of two open source controllers which are POX and Opendaylight using mininet emulator and D-ITG tool in terms of average flow delay and throughput and which of the two controllers was more stable with increased load through Measurement of the jitter parameter in a number of basic network architectures to determine which controller has the best behavior.

KEYWORDS: SDN, POX, Opendaylight, mininet, D-ITG.

1. المقدمة:

اقتُرِحَت فكرة الشبكات المعرَفة برمجياً لتجاوز كل السلبيات التي تعاني منها الشبكات التقليدية الحالية والتي تتمثل بالنقاط التالية:

أولاً: عدم وجود نقطة مركزية للإعدادات: حيث يتم إعداد كل جهاز على حدى، حتى لوكان الإعداد موحداً، وهذا الإعداد اليدوي ممل وعرضة للخطأ، وفي الوقت نفسه يلزم بذل جهد كبير لتحري الخلل وإصلاحه.

ثانياً: تعدد مستويات المهام على الجهاز الشبكي: ففي كل جهاز شبكي مستويين هما مستوى التحكم وهو من يقوم باتخاذ القرارات ومستوى البيانات أو التوجيه وهو الذي يقوم بتنفيذ ما قرره مستوى التحكم، وكما نعلم كلما زادت مستويات المهام على الجهاز كلما أدى ذلك إلى التقليل من أداء الجهاز الشبكي.

ثالثاً: تعددية الشركات المصنعة للتجهيزات الشبكية: حيث في الشركات الكبيرة ومراكز البيانات لا يتم الاعتماد على شركة مصنعة واحدة، والمشكلة أنّ كل شركة تملك نظام تشغيل خاص بها يختلف عن الشركات الأخرى أي ليس لدينا لغة موحدة تتمثل بنظام تشغيل موحد فيما بين الأجهزة الشبكية المختلفة التصنيع.

رابعاً: تظهر بعد فترة من الزمن من شراء الأجهزة الشبكية بروتوكولات وتقنيات جديدة قد نحتاج إليها، لكن هذه التجهيزات الموجودة لدينا في الشبكة لا تدعمها، كون هذه البروتوكولات والتقنيات ظهرت بعد تصنيعها وتحتاج إلى مواصفات أعلى. هنا سنضطر للتخلي عن هذه التقنيات والبروتوكولات، أما إذا كنا في حاجة ماسة إليها فسنضطر إلى تغيير كامل الأجهزة، ولكن عالم الشبكات سريع التقدم فمهما غيرنا الأجهزة وأحضرنا أجهزة جديدة فهي ستصبح قديمة بعد مرور الوقت.

خامساً: عند استخدام middleboxes مثل جدران الحماية وأنظمة كشف التسلل فإنها تراكب فوق البنية التحتية للشبكة الأساسية [1].

لذلك تم اقتراح فكرة الشبكات المعرَفة برمجياً حيث عن طريق بنية SDN تصبح الشبكة عبارة عن عناصر توجيه للرزم "بسيطة"، من ناحية أخرى فإن قرارات التوجيه عالية المستوى ومعلومات الحالة تكون مركزية في وحدة تحكم مركزية خارجية ومنفصلة، بدلاً من فرض السياسات وتشغيل البروتوكولات على الأجهزة المتفرقة [2]. وبما أن وحدة التحكم هي من تمثل ذكاء الشبكة وهي من تقوم بصنع القرارات ولسلوكها الأثر الأكبر على أداء الشبكة لذلك قمنا بتحليل أداء اثنين من وحدات التحكم المتوفرة والمفتوحة المصدر وهي Opendaylight وPOX.

2. وحدة التحكم Opendaylight:

هي منصة مفتوحة المصدر، تم تطويرها من قبل مؤسسة linux بلغة ODL.java لاتدعم openflow فحسب بل تدعم أيضاً southbound APIs أخرى مثل BGP–LS وLisp، قام بعض البائعين مثل Big switch وCisco بترويج ODL [3].

3. وحدة التحكم POX:

تم تطوير منصة العمل هذه بالكامل باستخدام لغة Python [4]. تدعم بروتوكول openflow فقط وتتميز بأنه يمكن بوقت قليل فهم مكوناتها ومعرفة كيفية تشغيل هذه المكونات ويمكن تطوير تطبيقات خاصة بنا لتأدية وظائف معينة في الشبكة وإضافتها وتشغيلها بسهولة على وحدة التحكم POX فهي موجهة نحو البحث والتعليم.

4. مواد وطرق البحث:

سنستخدم في الاختبارات برنامج المحاكاة Mininet، حيث يستند هذا المحاكي إلى Linux لنمذجة الشبكات المعرَفة برمجياً فهو يوفر طريقة بسيطة لاختبار الشبكات

لتطوير تطبيقات Openflow حيث يسمح باختبار طوبولوجيا كبيرة ومعقدة دون الحاجة إلى شبكة مادية [12].

سنستخدم أيضاً الأداة D-ITG والتي تستخدم لتوليد أنواع مختلفة من المرور وقياس معاملات الأداء في الشبكة مثل متوسط تأخير التدفق والإنتاجية وjitter.

5. الدراسات السابقة:

في الورقة البحثية[5] تم اختبار العديد من وحدات التحكم باستخدام أداة تسمى cbench وتم قياس الإنتاجية عن طريق إرسال أكبر عدد من رزم packet-in لحساب العدد الأقصى لعدد الرزم التي تتعامل معها وحدة التحكم وتم قياس وقت الاستجابة حيث ترسل cbench رزمة packet-in وينتظر الرد لحساب الوقت المستغرق لمعالجة رزمة واحدة بواسطة وحدة التحكم، وتم تشغيل cbench ووحدة التحكم على نفس الجهاز. وفي الدراسة[6] تم تقديم تحليل لوحدات التحكم الشائعة مفتوحة المصدر(nox,pox,beacon,floodlight,mul,ryu,maestro) حيث تم تحليل الأداء وقابلية التوسع والوثوقية والأمان ولأغراض الاختبار تم استخدام أداة تسمى hcprobe وأداة cbench .وفي الورقة [7] قام الباحثون بدراسة وتقييم أداء عدد من وحدات التحكم وهي(onons,ruy,floodlight,opendaylight) من حيث زمن الاستجابة والإنتاجية باستخدام الأداة cbench .وفي البحث [8] تم إجراء تقييم لأداء خمس وحدات تحكم(onos,libfluid,ruy,pox,opendaylight) من حيث الإنتاجية ومتوسطRTT (Round Trip Time) باستخدام أوامر ipref و ping ،تم إجراء الاختبار باستخدام الهيكل الخطي فقط في محاكي mininet .أما في [9] تم إجراء مقارنة نظرية بين عدد من وحدات التحكم .وفي [3] تم مقارنة pox و floodlight وopendaylight في شبكة محاكاة تم إنشاؤها بواسطة محاكي mininet من حيث إنتاجيةtcp وudp باستخدام أمر ipref ،ومتوسطRTT للرزمة الأولى فقط من التدفق عبر أمر ping ،تم إجراء الاختبارات باستخدام الهيكل

الشجري وهيكل محدد من قبل المستخدم. وركزت الورقة البحثية [10] على اثنين من وحدات التحكم وهما pox و floodlight وقاموا بمقارنة الأداء عن طريق تحليل الإنتاجية والتي تم حسابها من خلال النظر في استخدام عرض النطاق الترددي وRTT، كما تم تحليل RTT باستخدام الأمر ping، وذلك في عدد من بنى الشبكات الأساسية المضمنة في mininet وهي single و linear و tree بالإضافة إلى طوبولوجيا محددة من قبل المستخدم.

6. تحليل الأداء:

ما يميز بحثنا عن الأبحاث السابقة هو قيامنا بتحليل أداء وحدتي التحكم pox و opendaylight من حيث العوامل التالية المؤثرة على أداء الشبكة وهي متوسط تأخير التدفق ومتوسط معدل البتات ومتوسط معدل الرزم و jitter باستخدام الأداة D-ITG لتحديد وحدة التحكم ذات الأداء الأفضل في عدد من بُنى الشبكات الأساسية المضمنة في mininet وهي single وlinear و tree.

6-1 توليد المرور(Traffic):

في كل الاختبارات قمنا باستخدام الأداة D-ITG لتوليد تدفقين UDP مرسلة من المضيف الأول إلى المضيف التاسع بحيث يكون عدد الرزم المولدة من التدفق الأول كل ثانية 1000 رزمة وعدد الرزم المولدة من التدفق الثاني كل ثانية 2000 رزمة، كل رزمة في أي تدفق مؤلفة من 512 بايت (القيمة الافتراضية) علماً أن مدة توليد أي تدفق 10 ثوانٍ (القيمة الافتراضية). ومن أجل كل تدفق قمنا بإعادة تنفيذ كل اختبار خمس مرات ثم أخذنا المتوسط الحسابي للنتائج. من المقاييس التي يتم الحصول عليها بتحليل ملفات السجل التي قمنا بتوليدها على طرف المستقبل باستخدام أداة D-ITG :

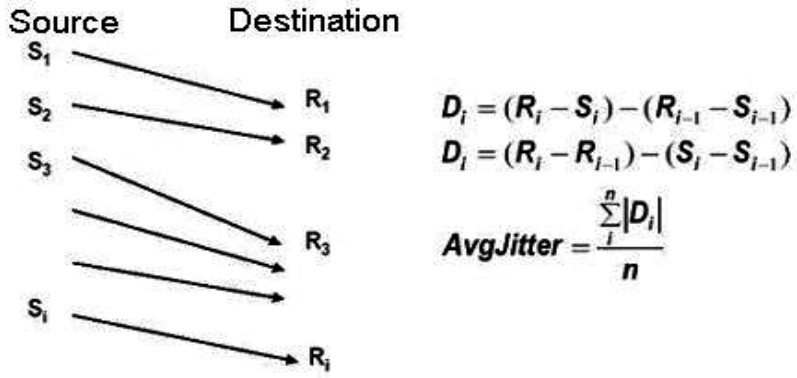
1-متوسط تأخير التدفق: وهو عبارة عن حاصل قسمة التأخير لجميع رزم التدفق المستلمة على عددها، والتأخير لكل رزمة هو الفرق بين وقت الاستقبال ووقت الإرسال لهذه الرزمة.

$$\text{Average Delay}=\sum (\text{rxTime} - \text{txTime}) / \text{Number of received packets} \quad (1)$$

وكما نعلم أن هذا المعامل يعتبر من المعاملات الأكثر أهمية عند دراسة سلوك أي شبكة، فكلما قل التأخير كلما زادت فعالية الشبكة، أما عند تجاوزه لحدود التسامح فإن ذلك سيجعلها غير فعالة.

2-متوسط معدل البتات المستلمة كل ثانية ومتوسط معدل الرزم المستلمة كل ثانية: كما نعلم أنه كلما كانت كمية البيانات المستلمة كل ثانية أكبر كلما كان ذلك يعبر عن سلوك أفضل للشبكة.

3- jitter: من العوامل الرئيسية المؤثرة على أداء الشبكة هو القدرة على التعامل مع الحمل بطريقة فعالة، لذلك قمنا باختيار هذا المعامل لنرى أياً من وحدتي التحكم هي أكثر ثباتاً مع زيادة الحمل. وتقوم أداة D-ITG بحساب هذا المعامل بالصيغة التالية:



الشكل (1) صيغة حساب jitter [11].

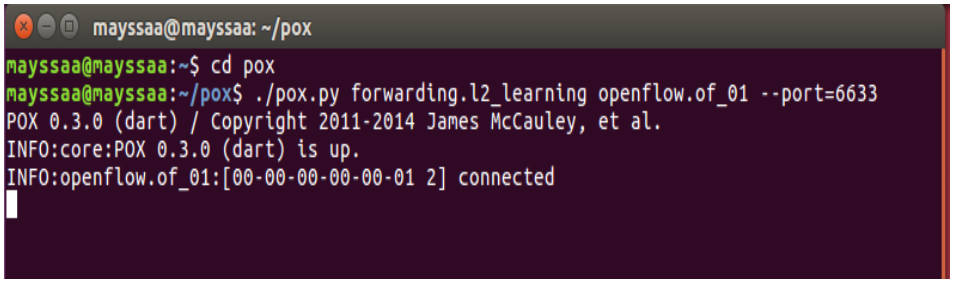
حيث يتوافق Si و Ri على التوالي مع txTime (وقت الإرسال للرزمة) و rxTime (وقت الاستقبال للرزمة).

6-2 الاختبار الأول:

استخدمنا في هذا الاختبار طوبولوجيا single والتي تتألف من مبدل واحد متصل بمضيفات متعددة ويتصل المبدل بدوره بوحدة التحكم. تم تصميم هذه الطوبولوجيا مع 9 مضيفات باستخدام محاكيmininet من خلال الأمر التالي:

```
Sudo mn --topo single,9 --
controller=remote,ip=controller_ip,port=controller_port
```

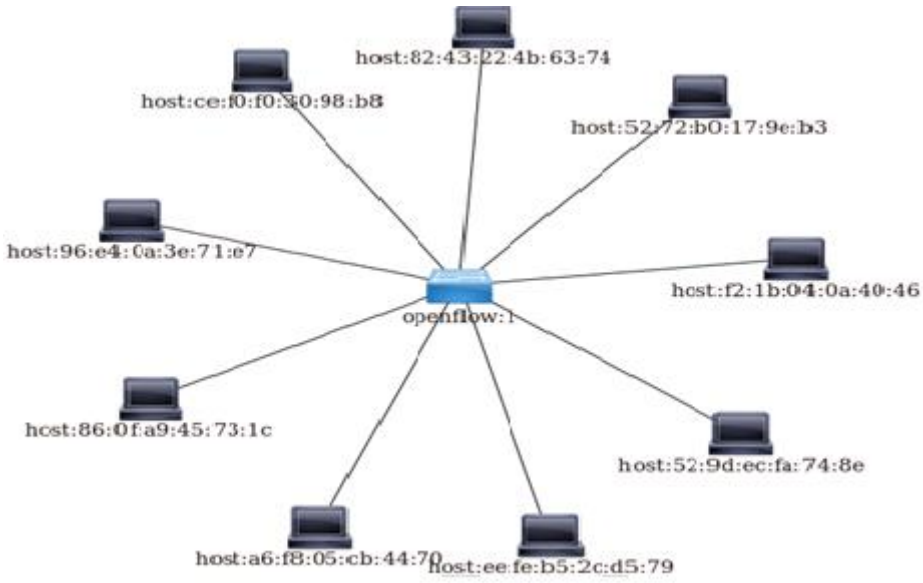
مع استبدال controller_ip بعنوان وحدة التحكم pox أو opendaylight، و controller_port برقم المنفذ الذي يتم التنصت عليه، علماً أن وحدات التحكم في جميع الاختبارات توجد على vm مختلفة عن vm التي تم تنصيب محاكي mininet عليها. وعند تنفيذ الاختبار مع وحدة التحكم pox يتم استدعاؤها على النحو الموجود في الشكل (2) حيث قمنا باستخدام الوحدة forwarding.l2_learning والتي تجعل مبدلات openflow تتصرف ك l2 learning switch ، ومن الشكل(2) نلاحظ اتصال وحدة التحكم بمبدل واحد.



الشكل (2) استدعاء وحدة التحكم POX.

وعند تنفيذ الاختبار مع وحدة التحكم opendaylight والتي تدعم واجهة مستخدم رسومية مستندة إلى الويب يمكن رؤية طوبولوجيا single عن طريق إدخال عنوان

URL التالي http://${ODL_IP}:8181/index.html على متصفح الويب وهذا ما يوضحه الشكل (3) .



الشكل(3) الطوبولوجيا الفردية في واجهة opendaylight الرسومية.

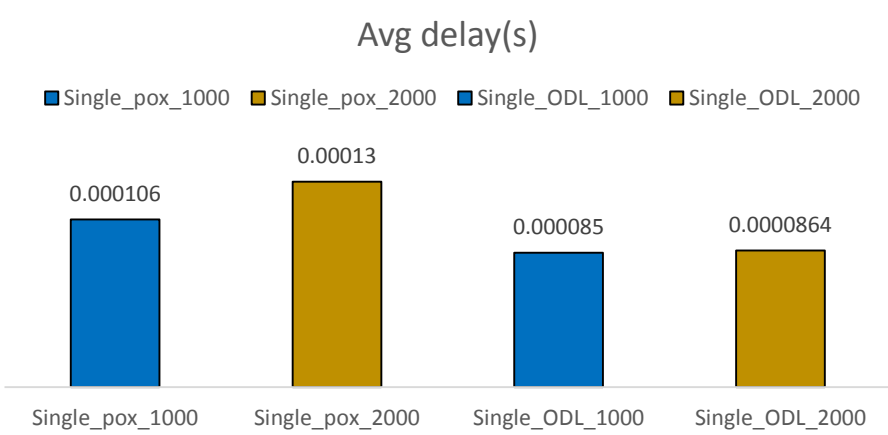
وبتوليد الtraffic وتحليل ملفات السجل بشكل مكافئ للفقرة **6-1** نحصل على النتائج التالية:

Average delay

Tests	Single_pox _1000	Single_pox _2000	Single_ODL _1000	Single_ODL _2000
Avg delay(s)	0.000106	0.00013	0.000085	0.0000864

الجدول (1) قيم متوسط التأخير لطوبولوجيا Single.

لفهم الفرق بشكل أوضح بين قيم متوسط التأخير رسمنا المخطط البياني التالي:



الشكل (4) متوسط التأخير لطوبولوجيا Single.

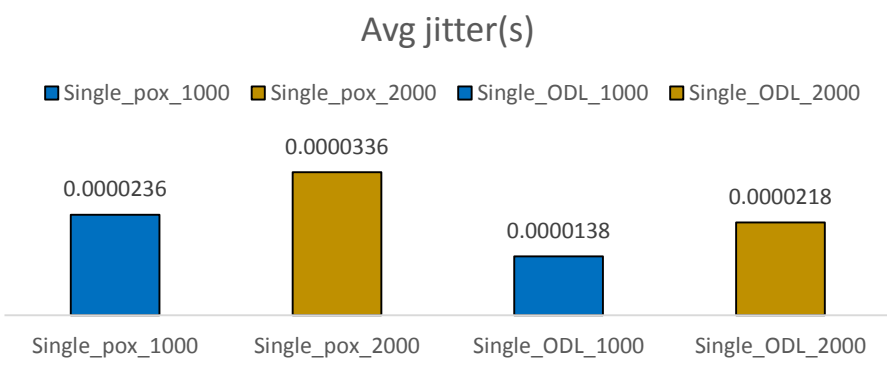
نلاحظ أن قيم متوسط التأخير في طوبولوجيا single هي أعلى مع وحدة التحكم pox وبالتالي يكون أداءها أفضل مع وحدة التحكم opendaylight.

Average jitter

Tests	Single_pox _1000	Single_pox _2000	Single_ODL _1000	Single_ODL _2000
Avg jitter(s)	0.0000236	0.0000336	0.0000138	0.0000218

الجدول (2) قيم متوسط jitter لطوبولوجيا single.

لفهم الفرق بشكل أوضح بين قيم متوسط jitter رسمنا المخطط البياني التالي:



الشكل (5) متوسط jitter لطوبولوجيا Single.

نلاحظ من أجل أي تدفق كان الانحراف بين قيم تأخير الرزم أقل مع وحدة التحكم opendaylight، وإذا ما قارنا القيم من ناحية زيادة الحمل نلاحظ أن opendaylight أكثر ثباتاً مع زيادة الحمل حيث كان الفرق في قيم jitter بين التدفقين مع opendaylight هو 0.000008 أقل من الفرق في قيم jitter بين التدفقين مع pox وهو 0.00001.

- Average bitrate و Average packet rate

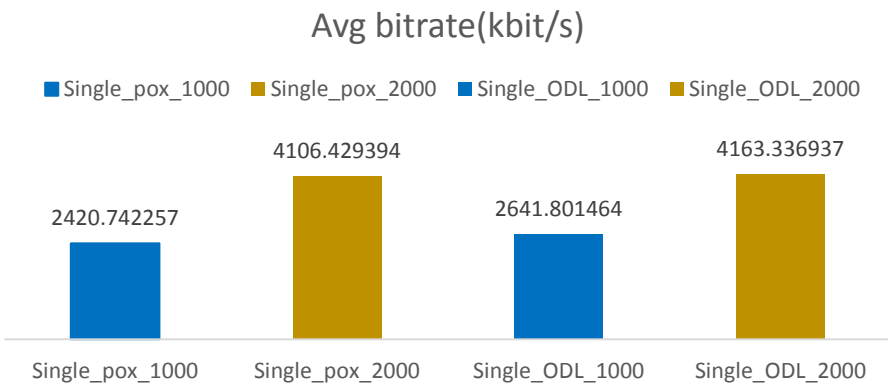
Tests	Single_pox_1000	Single_pox_2000	Single_ODL_1000	Single_ODL_2000
Avg bitrate(kb/s)	2420.742257	4106.429394	2641.801464	4163.336937

الجدول (3) قيم متوسط معدل البتات لطوبولوجيا single.

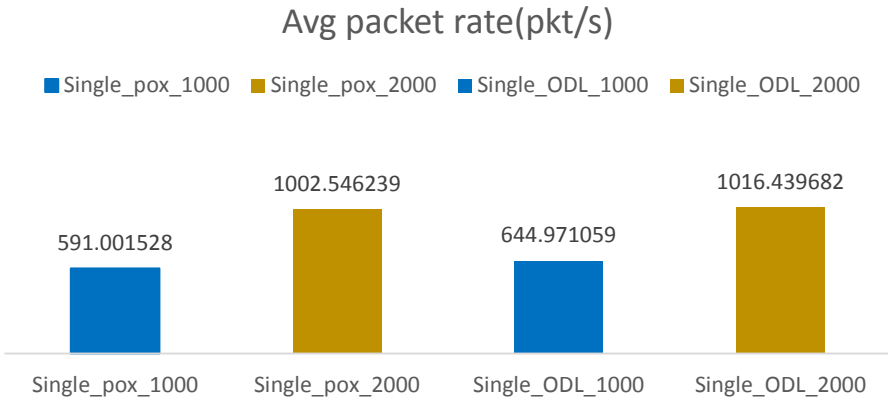
Tests	Single_pox_1000	Single_pox_2000	Single_ODL_1000	Single_ODL_2000
Avg pkt rate(pkt/s)	591.001528	1002.546239	644.971059	1016.439682

الجدول (4) قيم متوسط معدل الرزم لطوبولوجيا single.

لفهم الفرق بشكل أوضح بين القيم رسمنا المخططات البيانية التالية:



الشكل (6) متوسط معدل البتات لطوبولوجيا Single.



الشكل (7) متوسط معدل الرزم لطوبولوجيا Single.

نلاحظ أنَ كمية البيانات المستلمة كل ثانية مع وحدة التحكم opendaylight أكبر منها مع pox.

6-3 الاختبار الثاني:

استخدمنا في هذا الاختبار الطوبولوجيا الخطية Linear، مع هذه الطوبولوجيا يمكن تحديد عدد المبدلات في الشبكة، يتصل كل مبدل إلى مضيف، وجميع المبدلات متصلة ببعضها البعض والتي بدورها تتصل إلى وحدة التحكم. تم تصميم هذه

الطوبولوجيا مع 9 مبدلات وهذا أدى إلى وجود 9 مضيفات باستخدام محاكيmininet من خلال الأمر التالي:

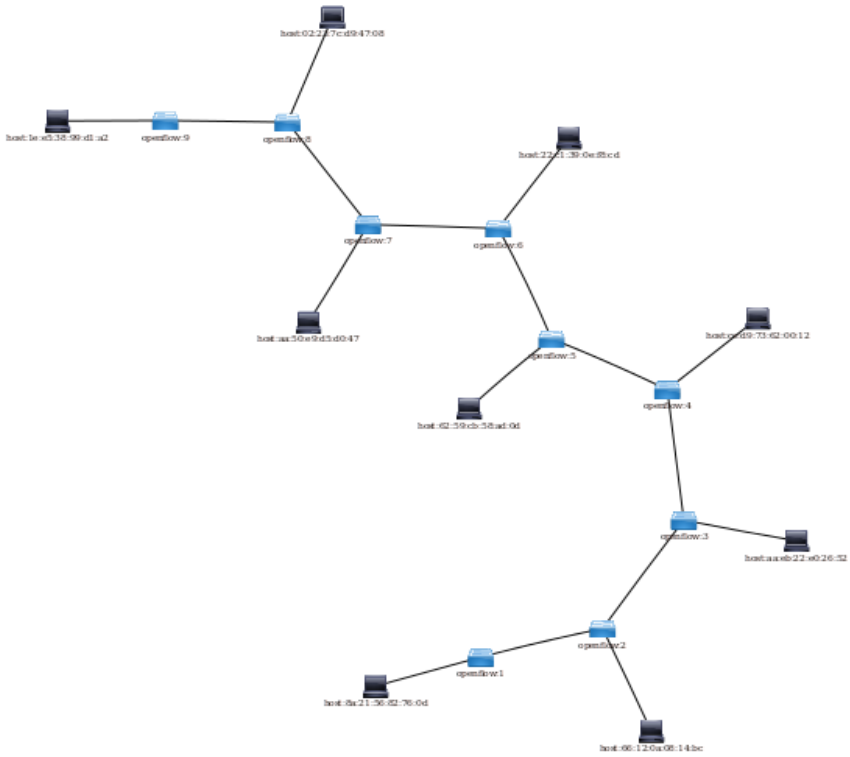
```
Sudo mn --topo linear,9 --
controller=remote,ip=controller_ip,port=controller_port
```

مع استبدال controller_ip بعنوان وحدة التحكم pox أو opendaylight، و controller_port برقم المنفذ الذي يتم التنصت عليه. وعند تنفيذ الاختبار مع وحدة التحكم pox يتم استدعاؤها على النحو الموجود في الشكل (8) ونلاحظ اتصال وحدة التحكم ب 9 مبدلات.

```
mayssaa@mayssaa:~/pox$ ./pox.py forwarding.l2_learning openflow.of_01 --port=6633
POX 0.3.0 (dart) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.3.0 (dart) is up.
INFO:openflow.of_01:[00-00-00-00-00-02 5] connected
INFO:openflow.of_01:[00-00-00-00-00-09 7] connected
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
INFO:openflow.of_01:[00-00-00-00-00-06 3] connected
INFO:openflow.of_01:[00-00-00-00-00-03 4] connected
INFO:openflow.of_01:[00-00-00-00-00-08 6] connected
INFO:openflow.of_01:[00-00-00-00-00-07 8] connected
INFO:openflow.of_01:[00-00-00-00-00-04 9] connected
INFO:openflow.of_01:[00-00-00-00-00-05 10] connected
```

الشكل (8) استدعاء وحدة التحكم POX.

وعند تنفيذ الاختبار مع وحدة التحكم opendaylight يمكن رؤية طوبولوجيا Linear على متصفح الويب كما في الشكل (9).



الشكل (9) الطوبولوجيا الخطية في واجهة opendaylight الرسومية.

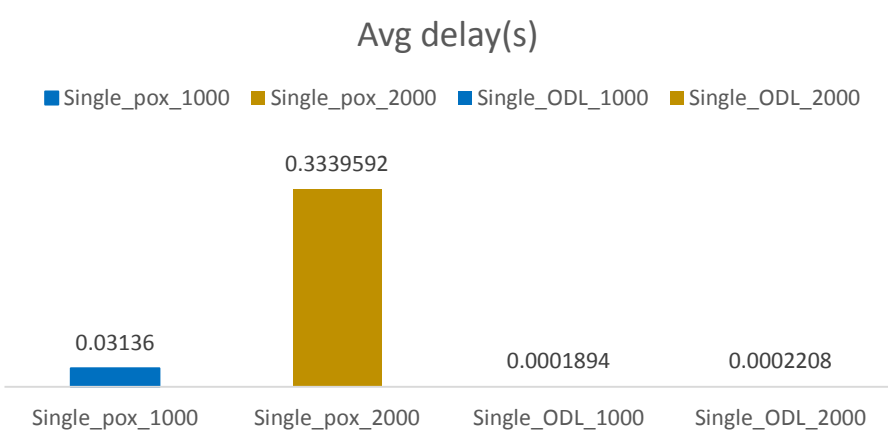
وبتوليد الtraffic وتحليل ملفات السجل بشكل مكافئ للفقرة **6-1** نحصل على النتائج التالية:

Average delay

Tests	Single_pox _1000	Single_pox _2000	Single_ODL _1000	Single_ODL _2000
Avg delay(s)	0.03136	0.3339592	0.0001894	0.0002208

الجدول (5) قيم متوسط التأخير لطوبولوجيا Linear.

لفهم الفرق بشكل أوضح بين قيم متوسط التأخير رسمنا المخطط البياني التالي:



الشكل (10) متوسط التأخير لطوبولوجيا Linear.

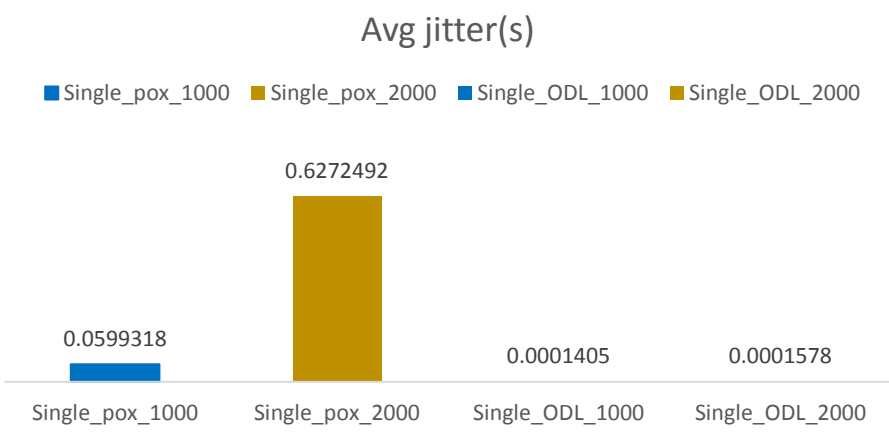
نلاحظ أن قيم متوسط التأخير في طوبولوجيا linear هي أعلى بكثير مع وحدة التحكم pox وبالتالي يكون سلوكها أفضل بكثير مع وحدة التحكم opendaylight. وإذا ما قارنا قيم متوسط التأخير في هذه الطوبولوجيا مع قيم التأخير في طوبولوجيا single نجد أن القيم في الطوبولوجيا الخطية أكبر منها في الفردية بسبب أن عدد المبدلات في الطوبولوجيا الخطية أكبر وبالتالي تحتاج الرزمة وقت أكبر حتى تصل إلى الهدف بسبب مرورها على أكثر من مبدل بينما في الطوبولوجيا الفردية لن تحتاج الرزمة سوى المرور عبر مبدل واحد فقط حتى تصل إلى الهدف.

Average jitter

Tests	Single_pox _1000	Single_pox _2000	Single_ODL _1000	Single_ODL _2000
Avg jitter(s)	0.0599318	0.6272492	0.0001405	0.0001578

الجدول (6) قيم متوسط jitter لطوبولوجيا Linear.

لفهم الفرق بشكل أوضح بين قيم متوسط jitter رسمنا المخطط البياني التالي:



الشكل (11) متوسط jitter لطوبولوجيا Linear.

نلاحظ من أجل أي تدفق كان الانحراف بين قيم تأخير الرزم أقل بكثير مع وحدة التحكم opendaylight، وإذا ما قارنا القيم من ناحية زيادة الحمل نلاحظ أن opendaylight أكثر ثباتاً مع زيادة الحمل حيث كان الفرق في قيم jitter بين التدفقين مع opendaylight هو 0.0000173 أقل بكثير من الفرق في قيم jitter بين التدفقين مع pox وهو 0.5673174 .

Average bitrate و Average packet rate

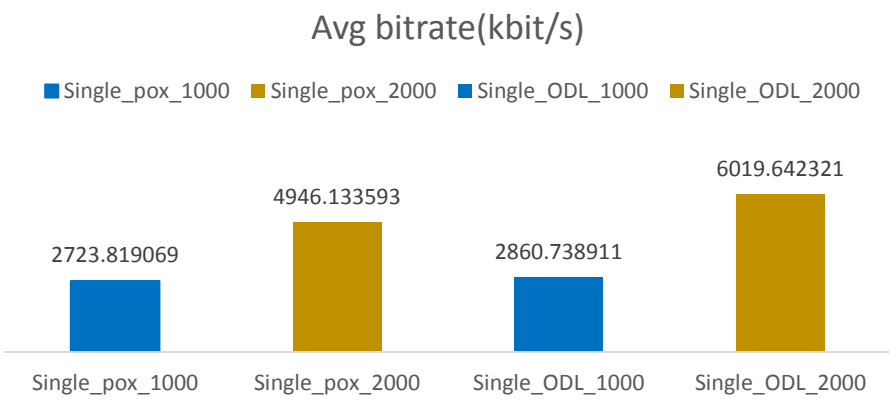
Tests	Single_pox_1000	Single_pox_2000	Single_ODL_1000	Single_ODL_2000
Avg bitrate(kb/s)	2723.819069	4946.133593	2860.738911	6019.642321

الجدول (7) قيم متوسط معدل البتات لطوبولوجيا Linear.

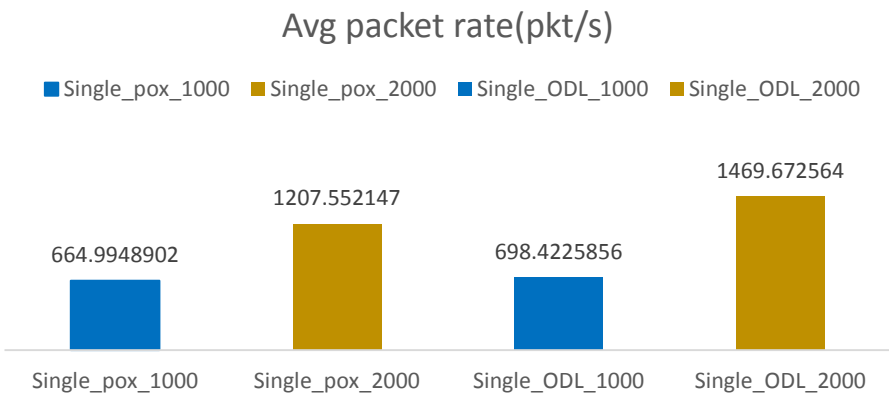
Tests	Single_pox _1000	Single_pox _2000	Single_ODL _1000	Single_ODL _2000
Avg pkt rate(pkt/s)	664.9948902	1207.552147	698.4225856	1469.672564

الجدول (8) قيم متوسط معدل الرزم لطوبولوجيا Linear.

لفهم الفرق بشكل أوضح بين القيم رسمنا المخططات البيانية التالية:



الشكل (12) متوسط معدل البتات لطوبولوجيا Linear.



الشكل (13) متوسط معدل الرزم لطوبولوجيا Linear.

نلاحظ أنَ كمية البيانات المستلمة كل ثانية مع وحدة التحكم opendaylight أكبر منها مع pox. وإذا ما قارنا كمية البيانات المستلمة كل ثانية في هذه الطوبولوجيا مع كمية البيانات المستلمة كل ثانية في طوبولوجيا single نجد أن القيم في الطوبولوجيا الخطية أكبر منها في الفردية بسبب توزيع الحمل على المبدلات في الطوبولوجيا الخطية.

4-6 الاختبار الثالث:

استخدمنا في هذا الاختبار الطوبولوجيا الشجرية tree، مع هذه الطوبولوجيا يمكن توصيل المبدلات والمضيفات في هيكل شجري. تم تصميم هذه الطوبولوجيا باستخدام محاكيmininet من خلال الأمر التالي:

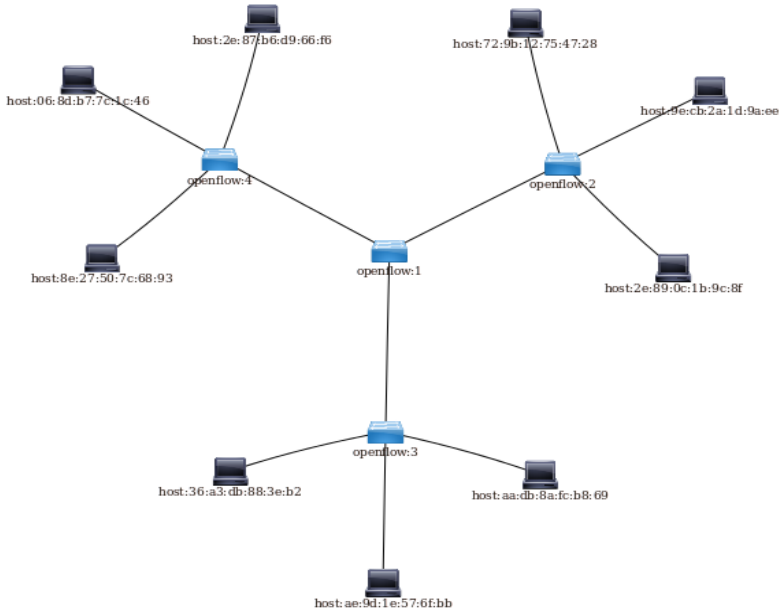
```
Sudo mn --topo tree,depth=2,fanout=3 --
controller=remote,ip=controller_ip,port=controller_port
```

هذا الأمر سيؤدي إلى إنشاء طوبولوجيا بعمق مستويين من المبدلات مع ثلاث أولاد لكل عقدة، أي سيمتلك الهيكل الشجري 4 مبدلات و 9 مضيفات مع استبدال controller_ip بعنوان وحدة التحكم pox أو opendaylight، و controller_port برقم المنفذ الذي يتم التنصت عليه. وعند تنفيذ الاختبار مع وحدة التحكم pox يتم استدعاؤها على النحو الموجود في الشكل (14) و نلاحظ اتصال وحدة التحكم ب 4 مبدلات.

```
mayssaa@mayssaa:~/pox$ ./pox.py forwarding.l2_learning openflow.of_01 --port=6633
POX 0.3.0 (dart) / Copyright 2011-2014 James McCauley, et al.
INFO:core:POX 0.3.0 (dart) is up.
INFO:openflow.of_01:[00-00-00-00-00-01 2] connected
INFO:openflow.of_01:[00-00-00-00-00-02 4] connected
INFO:openflow.of_01:[00-00-00-00-00-03 3] connected
INFO:openflow.of_01:[00-00-00-00-00-04 5] connected
```

الشكل(14) استدعاء وحدة التحكم POX.

وعند تنفيذ الاختبار مع وحدة التحكم opendaylight يمكن رؤية طوبولوجيا tree على متصفح الويب كما في الشكل (15).



الشكل (15) الطوبولوجيا الشجرية في واجهة opendaylight الرسومية.

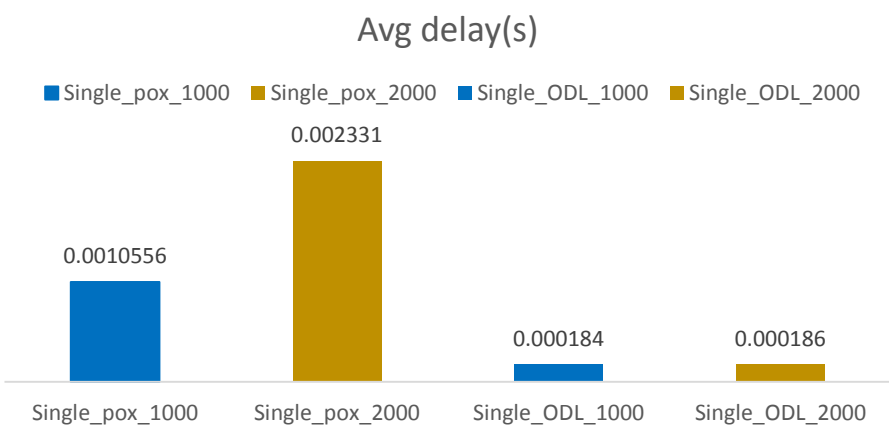
وبتوليد الtraffic وتحليل ملفات السجل بشكل مكافئ للفقرة **6-1** نحصل على النتائج التالية:

Average delay

Tests	Single_pox _1000	Single_pox _2000	Single_ODL _1000	Single_ODL _2000
Avg delay(s)	0.0010556	0.002331	0.000184	0.000186

الجدول (9) قيم متوسط التأخير لطوبولوجيا Tree.

لفهم الفرق بشكل أوضح بين قيم متوسط التأخير رسمنا المخطط البياني التالي:



الشكل (16) متوسط التأخير لطوبولوجيا Tree.

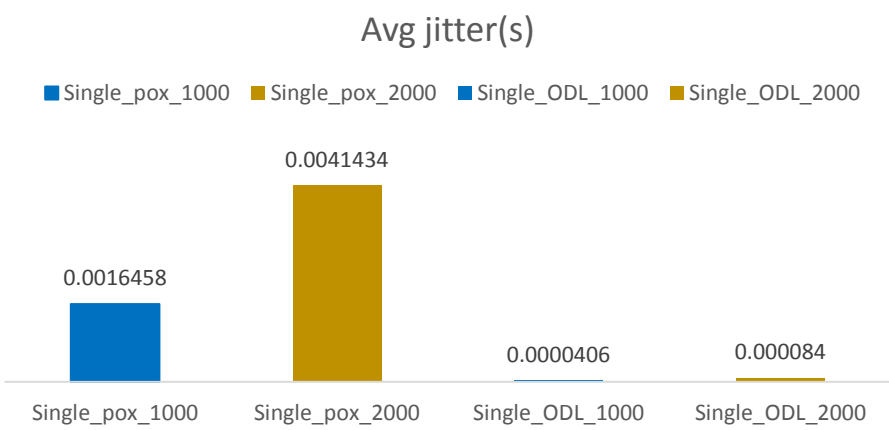
نلاحظ أن قيم متوسط التأخير في طوبولوجيا Tree هي أعلى بكثير مع وحدة التحكم pox وبالتالي يكون أداء الشبكة أفضل مع وحدة التحكم opendaylight. وإذا ما قارنا قيم متوسط التأخير في هذه الطوبولوجيا مع قيم التأخير في طوبولوجيا linear نجد أن القيم في الطوبولوجيا الشجرية أقل منها في الخطية بسبب أن عدد المبدلات في الطوبولوجيا الشجرية أقل وبالتالي ستحتاج الرزمة وقت أقل لتصل إلى الهدف.

Average jitter

Tests	Single_pox _1000	Single_pox _2000	Single_ODL _1000	Single_ODL _2000
Avg jitter(s)	0.0016458	0.0041434	0.0000406	0.000084

الجدول (10) قيم متوسط jitter لطوبولوجيا Tree.

لفهم الفرق بشكل أوضح بين قيم متوسط jitter رسمنا المخطط البياني التالي:



الشكل (17) متوسط jitter لطوبولوجيا Tree.

نلاحظ من أجل أي تدفق كان الانحراف بين قيم تأخير الرزم أقل بكثير مع وحدة التحكم opendaylight، وإذا ما قارنا القيم من ناحية زيادة الحمل نلاحظ أن opendaylight أكثر ثباتاً مع زيادة الحمل حيث كان الفرق في قيم jitter بين التدفقين مع opendaylight هو 0.0000434 أقل بكثير من الفرق في قيم jitter بين التدفقين مع pox وهو 0.0024976 .

Average bitrate و Average packet rate

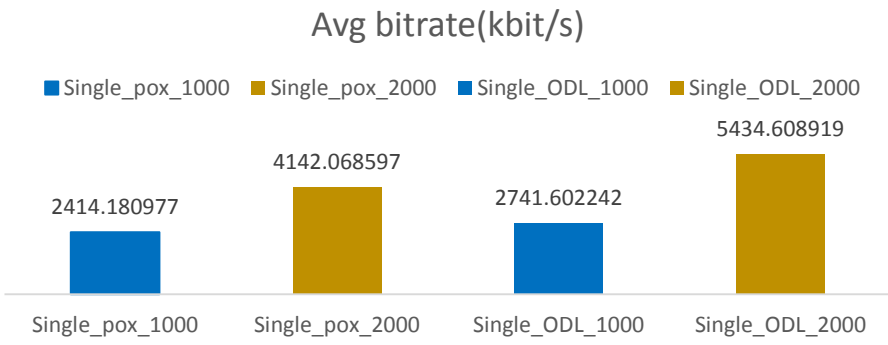
Tests	Single_pox _1000	Single_pox _2000	Single_ODL _1000	Single_ODL _2000
Avg bitrate(kb/s)	2414.180977	4142.068597	2741.602242	5434.608919

الجدول (11) قيم متوسط معدل البتات لطوبولوجيا Tree.

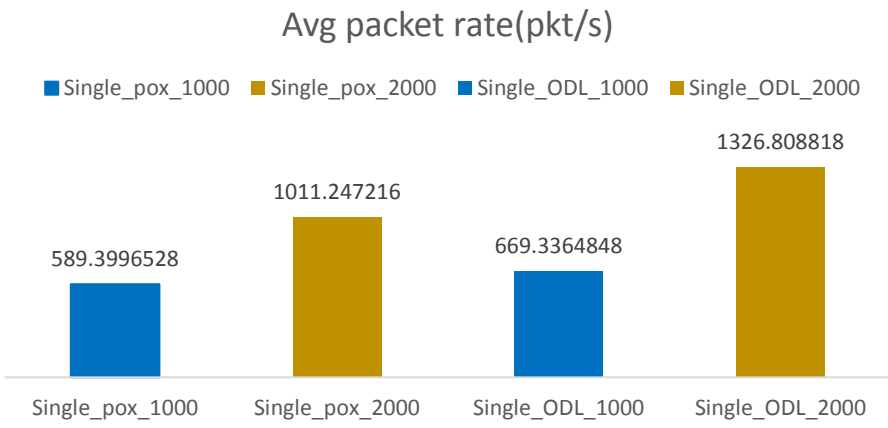
Tests	Single_pox _1000	Single_pox _2000	Single_ODL _1000	Single_ODL _2000
Avg pkt rate(pkt/s)	589.3996528	1011.247216	669.3364848	1326.808818

الجدول (12) قيم متوسط معدل الرزم لطوبولوجيا Tree.

لفهم الفرق بشكل أوضح بين القيم رسمنا المخططات البيانية التالية:



الشكل (18) متوسط معدل البتات لطوبولوجيا Tree.



الشكل (19) متوسط معدل الرزم لطوبولوجيا Tree.

نلاحظ أنَ كمية البيانات المستلمة كل ثانية مع وحدة التحكم opendaylight أكبر منها مع pox. وإذا ما قارنا كمية البيانات المستلمة كل ثانية في هذه الطوبولوجيا مع

كمية البيانات المستلمة كل ثانية في طوبولوجيا Linear نجد أن القيم في الطوبولوجيا الخطية أكبر منها في الشجرية بسبب توزيع الحمل على مبدلات أكثر في الطوبولوجيا الخطية.

7. ملخص النتائج:

وفقاً للنتائج التي تم الحصول عليها نستنتج أن وحدة التحكم opendaylight توفر أداءً أفضل للشبكة مقارنة بوحدة التحكم pox، فعند استخدام وحدة التحكم opendaylight حصلنا على تأخير أقل وإنتاجية أعلى وتعاملت مع زيادة الحمل بطريقة أفضل من وحدة التحكم pox، وكما رأينا أن وحدة التحكم opendaylight توفر واجهة مستخدم رسومية مستندة إلى الويب من خلالها يمكن رؤية طوبولوجيا الشبكة، ولكن يجب الإشارة إلى أن وحدة التحكم pox توفر سهولة أكبر من ناحية تشغيلها وفهم مكوناتها وتطويرها.

8. المراجع :

[1] A Survey of Software-Defined Networking: Past,Present, and Future of Programmable Networks Bruno Nunes Astuto, Marc Mendonça, Xuan Nam Nguyen, Katia Obraczka,Thierry Turletti, HAL Id: hal-00825087 https://hal.inria.fr/hal-00825087v5 Submitted on 19 Jan 2014.

[2] Enhancing quality of service in software-defined networks, professor antonio corradi, academic year 2013-2014.

[3] Performance analysis of SDN controllers: POX, Floodlight and Opendaylight. GERELTSETSEG Altangerel, TUGSJARGAL Chuluuntsetseg, DASHDORJ Yamkhin/Ph.D/. Department of Information network and Security ,The school of Information and Communication Technology, Mongolian University of Science and Technology ,Ulaanbaatar, Mongolia, Conference Paper April 2019

[4] POX Wiki, Added by Ali Al-Shabibi, last edited by Murphy McCauley On Mar 05, 2015

https://openflow.stanford.edu/display/ONL/POX+Wiki.

[5] SDN Controllers: A Comparative Study,Ola Salman Imad Elhajj Ayman Kayssi Ali Chehab,Electrical and Computer Engineering ,Department American University of Beirut,Beirut 11 07 2020, Lebanon.

[6] Advanced Study of SDN/OpenFlow controllers, Alexander Shalimov, Dmitry Zuikov, Daria Zimarina, Vasily Pashkov, Ruslan Smeliansky, Conference Paper October 2013 .

[7] A Comparative Evaluation of the Performance of Popular SDN Controllers, Lusani Mamushiane, Albert Lysko, Sabelo Dlamini ,CSIR Pretoria, South Africa, 2018 .

[8] Performance Evaluation and Comparison of Software

Defined Networks Controllers, Mahmood Z. Abdullah, Nasir A. Al-awad, Fatima W. Hussein,Computer Engineering Department, Al-Mustansiriyah University, Baghdad, Iraq, 2018.

[9] sdn controllers comparison, v r sudarsana raju ,Srit, bangalore, 7, Jul.-2018.

[10] Performance Analysis of SDN/OpenFlow Controllers: POX Versus Floodlight ,Idris Z. Bholebawa Upena D. Dalal , Department of Electronics and Communication Engineering, S. V. National Institute of Technology, Surat, Gujarat 395007, India, published online :30 August 2017 .

[11] D-ITG 2.8.1 Manual, Alessio Botta, Walter de Donato, Alberto Dainotti, Stefano Avallone, and Antonio Pescap´e,COMICS (COMputer for Interaction and CommunicationS) Group,Department of Electrical Engineering and Information Technologies ,University of Napoli Federico, October 28, 2013

[12] Development of a performance measurement tool for SDN, Marion MAUGENDRE, Date :07/10/2015.