Bandwidth Control with QoS tools Through the Controller (Ryu) and (openflow) Protocol

Dr. Mothanna Alkubeily*
Boushra Hasan**

(Received 9 / 5 / 2021. Accepted 10 / 4 / 2022)

□ ABSTRACT □

The increased traffic on the network and the inability of the traditional networks with their current structure to deal with the fast traffic on the network, led to the emergence of a new dynamic flexible structure programmed according to the behavior of the data flow across the network, this structure is software defined networks SDN, which gives flexibility in controlling network resources by separating the control plane from the data plane. The problem of end to end QUALITY OF SERVICE in traditional networks is an ongoing problem. Many researchers have proposed a set of solutions, but most of them either failed or their solutions were not implemented, and software defined networks came as a solution to reduce the limitations in Current traditional network architectures

SDNs can help provide QoS in a variety of ways due to some of their features such as the concept of per flow control and multi-header fields based routing

In this research, we present two methods of controlling bandwidth using the open flow protocol and the RYU controller, namely the quality of service for each flow (per-flow Qos), which is able to precisely control the quality of service for each flow, but with the increase in the number of flows, this will lead to an increase in the rules or The flow entries in the flow table of the switch that are set to control the bandwidth and thus this method is not scalable and unable to handle with many flows, so the second method was used where the flows are divided into several categories depending on the value (DSCP), (DSCP) is 8 bits In the (ipv4) header within the (type of service TOS) field or within the (traffic class) field in the (ipv6) header, Then queues are allocated by service class that includes a group of flows that have the same required QoS level. And we explain how these methods can control the bandwidth According to the user's need and thus the control of network resources and the optimal use of them.

Keywords: software defined networks, Openflow protocol, Ryu controller, Quality of service QoS

*Associate Professor, Department of Communication and Electronics, Faculty of mechanical and Electrical Engineering, Tishreen University, Lattakia, Syria. Email: mothanna.alkubeily@gmail.com
**Postgraduate Student (PhD student), Department of Communication and Electronics, Faculty of Mechanical and Electrical Engineering, Tishreen University, Lattakia, Syria. bsa.fa@hotmail.com

التحكم بعرض الحزمة باستخدام أدوات جودة الخدمة من خلال المتحكم (Ryu) والبروتوكول (openflow)

د. مثنى القبيلي •

بشرى حسن••

(تاريخ الإيداع 9 / 5 / 2021. قُبِل للنشر في 10/ 4 / 2022)

□ ملخّص □

إن حركة المرور المتزايدة على الشبكة وعدم قدرة الشبكات التقليدية ببنيتها الحالية على التعامل مع حركة المرور السريعة على الشبكة، أدى إلى ظهور بنية جديدة ديناميكية مرنة ومبرمجة وفقاً لسلوك تدفق البيانات عبر الشبكة، تدعى هذه البنية بالشبكات المعرفة بالبرمجيات SDN والتي تعطي مرونة في ضبط موارد الشبكة عن طريق فصل مستوى التحكم عن مستوى البيانات. تعد مشكلة دعم جودة الخدمة الشاملة (end to end Quality of Service) في الشبكات التقليدية مشكلة مستمرة، فالعديد من الباحثين قد اقترحوا مجموعة من الحلول لكن أغلبها قد فشل أو لم يتم تنفيذها، فتم اقتراح الشبكات المعرفة بالبرمجيات للحد من القيود الموجودة في البنى المعمارية للشبكات التقليدية الحالية.
يمكن أن تساعد شبكات (SDN) في توفير جودة الخدمة بطرق متنوعة بسبب بعض ميزاتها مثل مفهوم التحكم بكل تدفق (per flow control) والتوجيه اعتماداً على عدة حقول في الترويسة الخاصة بالرزمة.
نقدم في هذا البحث طريقتين للتحكم بعرض الحزمة باستخدام بروتوكول المرور المفتوح (open flow) والمتحكم (RYU) وهما: جودة الخدمة لكل تدفق (per-flow Qos)، وهي قادرة على التحكم بدقة بجودة خدمة كل تدفق، لكن زيادة عدد التدفقات سيؤدي إلى زيادة قواعد التدفق (flow entries) في جدول التدفق للمبدل التي يتم تعيينها للتحكم بعرض النطاق الترددي، وبالتالي هذه الطريقة غير قابلة للتوسع والتعامل مع التدفقات الكثيرة. أما الطريقة الثانية فيتم فيها تقسيم التدفقات الى عدة فئات اعتماداً على قيمة (**DSCP differentiatted service codepoint**)، والتي هي عبارة عن 8 بتات موجودة في ترويسة (ipv4) ضمن حقل (type of service TOS) أو ضمن حقل (traffic class) في ترويسة (ipv6)، ومن ثم يتم تخصيص قوائم انتظار حسب صنف الخدمة الذي يشمل مجموعة من التدفقات التي تملك نفس سوية جودة الخدمة المطلوبة، ونوضح كيف يمكن لهذه الطرق من التحكم بعرض الحزمة وفقاً لحاجة المستخدم وبالتالي ضبط موارد الشبكة والاستخدام الامثل لها.

الكلمات المفتاحية: الشبكات المعرفة بالبرمجة، بروتوكول (open flow)، المتحكم (Ryu)، جودة الخدمة (Qos).

• **أستاذ مساعد، قسم هندسة الاتصالات والالكترونيات، كلية الهندسة الميكانيكية والكهربائية، جامعة تشرين، اللاذقية سورية. mothanna.alkubeily@gmai.com**

• • **طالبة دكتوراه، قسم هندسة الاتصالات والالكترونيات، كلية الهندسة الميكانيكية والكهربائية، جامعة تشرين، اللاذقية، سورية. bsa.fa@hotmail.com**

مقدمة:

لجودة الخدمة في الشبكات مجموعة من العوامل التي يمكن أن تحدد كيف يمكن لشبكة ما أن تتعامل مع الرزم داخل الشبكة [1]، بعضها عوامل فنيَة وبعضها عوامل غير فنيَة مثل (وثوقية الخدمة، التوافرية، التأخير، درجة الخدمة، التوسعية، الكفاءة). وقد كانت تلبية متطلبات جودة الخدمة لفترات طويلة أمراً صعب التحقيق من أجل بعض التطبيقات الحديثة وذلك بسبب تعقيد حلول تأمين متطلبات جودة الخدمة بسبب ضبط الإعدادات(manual configuration) بشكل يدوي لكل جهاز من قبل مدير الشبكة[2].

حالياً لا تدعم حلول جودة الخدمة التحكم الدقيق بمرور الشبكة، علاوة على ذلك تتطلَب حتى الإجراءات البسيطة إعدادات يدوية معقدة على أجهزة الشبكة [3]، وهنا يأتي دور الشبكات المعرفة بالبرمجيات (SDN)، فهذه الشبكات ليست مجبرة على التعامل مع إعدادات وتهيئة منخفضة المستوى لأجهزة الشبكة، بل تكون مزوَدة بنظرة مجردة للشبكة من قبل المتحكم الذي يقوم بالحصول على معلومات حالة الشبكة، ومن ثم يتم وضع سياسة التحكم بناءً على هذه المعلومات ويمكن أن يتم تعديلها بشكل ديناميكي دون الحاجة إلى إعادة تهيئة كل جهاز على حدى[4].

لا تكون سياسات التحكم وأصناف التدفق في (SDN) مقيدة، بل يمكن أن تكون قواعد التدفق محددة اعتماداً على كل تدفق على حدى، ومهمة المتحكم هي تثبيت هذه القواعد في المبدلات وكل هذه الأمور مهمَة جداً بالنسبة الى جودة الخدمة [5]، حيث أثبتت الدراسات أن الشبكات المعرفة بالبرمجيات هي من أهم المفاهيم من أجل تحقيق وتطوير جودة الخدمة في الشبكة [6].

أهمية البحث وأهدافه:

تأتي أهمية هذا البحث من خلال تناوله لموضوع جودة الخدمة باستخدام البروتوكول (openflow) في الشبكات المعرفة بالبرمجيات، حيث إن الحلول التقليدية لجودة الخدمة لم تكن عند المستوى المطلوب، ويهدف هذا البحث إلى تقديم مقارنة لطريقتين من أجل تنفيذ آليات جودة الخدمة باستخدام البروتوكول (openflow) والمتحكم (RYU)، ونوضح كيف يمكن لهذه الطرق التحكم بعرض الحزمة وفقاً لحاجة المستخدم وبالتالي ضبط موارد الشبكة والاستخدام الامثل لها.

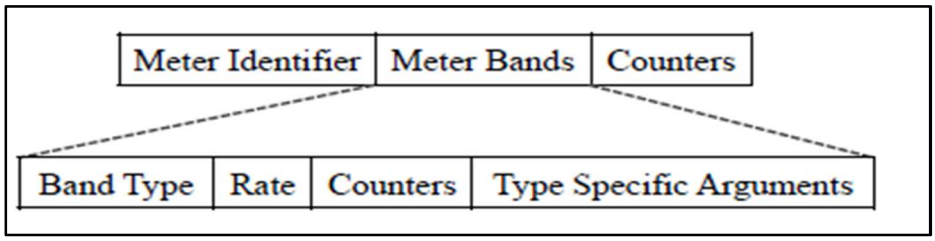
طرائق البحث ومواده:

تم في هذا البحث استخدام برنامج (mininet)، وهو محاكي قادر على محاكاة شبكة كبيرة من على جهاز حاسوب واحد، ومن أشهر المحاكيات المستخدمة في مجال الشبكات المعرفة بالبرمجيات، كما تم استخدام الأداة (miniedit) التي يوفرها برنامج (mininet) من أجل رسم طبولوجيا الشبكة، واستخدام الأداة (iperf) لتوليد مرور في الشبكة وقياس عرض الحزمة.

1. جودة الخدمة في البروتوكول (openflow):

يتمتَع كل إصدار من إصدارات البروتوكول (openflow) بميزات وخصائص وتغيرات تختلف عن الإصدار الذي سبقه، ومن أهم الميزات والتغيرات المتعلقة بجودة الخدمة التي طرأت على إصدارات البروتوكول (openflow) المختلفة [7,8] هي:

- **(OpenFlow 1.0)**: في الإصدار الأول يوجد إجراء اختياري يدعى (enqueue)، وهو إجراء يمكن من خلاله تخصيص قائمة انتظار لمنافذ المبدل، ويتعلق عدد قوائم الانتظار بعدد منافذ المبدل، ويستطيع المتحكم الاستعلام عن معلومات تخص قوائم الانتظار للمبدلات عن طريق البروتوكول (OF-CONFIG). لكن بشكل عام يكون ضبط إعدادات قوائم الانتظار خارج نطاق البروتوكول (open flow) ويتطلب نسخة من (1.2) وأعلى.
- **(OpenFlow 1.1):** يدعم هذا الإصدار (MPLS LABEL) وعلامات (VLAN) وتصنيف الرزم (traffic classes).
- **(OpenFlow 1.2):** في هذا الإصدار أصبح لدى المتحكم القدرة على الاستعلام عن قوائم الانتظار الموجودة على منافذ كل مبدل، واستخدام هذه القوائم من أجل تنظيم التدفقات وجدولتها وعزلها عن بعضها البعض حسب طبيعتها، كذلك يمكن أن يخصص لكل قائمة انتظار معدل نقل أعظمي خاص بها.
- **(OpenFlow 1.3):** أصبح هذا الإصدار يدعم جداول العدادات (meter tables)، والتي وظيفتها تحديد معدلات نقل البيانات من خلال إمكانية توجيه الرزم إلى (meter) له معرف خاص به ومعدل نقل خاص به، ويتألف كل جدول من مجموعة من المدخلات، ويتألف كل مدخل من معرف يمكن أن يكون له أكثر من نطاق، ويوضح الشكل(1) مكونات جدول العدادات في الإصدار الثالث من بروتوكول (openflow)



الشكل (1): مكونات جدول العدادات في الإصدار الثالث من البروتوكول (open flow)

- **(OpenFlow 1.4):** يسمح هذا الإصدار للمتحكم بمراقبة التغيرات التي يتم إجراءها من قبل أي متحكم آخر في الشبكة وماذا يمكن أن يطرأ على جداول التدفق من حيث (إضافة قاعدة تدفق جديد، حذف قاعدة تدفق، تعديل قاعدة تدفق)، حيث يكون هناك مجموعة من المتحكمات، وكل مجموعة فرعية من الجداول لهم (id) ونمط خاص بهم تقوم بإعلام المتحكم عن أي تغيير يمكن أن يطرأ على هذه الجداول.
- **(OpenFlow 1.5):** أصبح من الممكن في هذا الإصدار ربط عدة عدادات (meters) لمدخل تدفق معين.

2. جودة الخدمة في متحكمات (SDN):

لا يدعم البروتوكول (openflow) حالياً التحكم بسلوك قوائم الانتظار الموجودة على منافذ المبدلات وتهيئة إعداداتها، ويتم التعامل مع تهيئة قوائم الانتظار بواسطة بروتوكولين هما (OF-CONFIG Open Flow Configuration) و (OVSDB Open VSwitch Data Base management protocol) حيث يتم حالياً توحيد مواصفاته (standardization) البروتوكول الأول من قبل مؤسسة (ONF)،والبروتوكول الثاني تم مسبقاً توحيد مواصفاته من قبل (IETF) [9] .

وعلى الرغم من أن البروتوكول الثاني تم تنفيذه على مبدلات (OVS) فلا يوجد حتى الآن متحكمات تدعم إدارة قوائم انتظار بمعايير موحدة ، وفيما يلي بعض المتحكمات مع دعم جودة الخدمة الخاصة بكل متحكم.

2. 1. المتحكم (Open Daylight):

تم استخدام مكون إضافي (OVSDB southbound plugin) في إصدار المتحكم (odl lithium release) والذي يمكنه من إدارة قوائم الانتظار الموجودة على منافذ المبدلات وتهيئتها والتحكم بإعداداتها[10].

يوجد في متحكم (ODL) وحدة تسمى (reservation module) والتي تؤمن حجز موارد منخفضة المستوى بشكل ديناميكي بحيث يمكن لمستخدمي الشبكة من الحصول على الخدمات والاتصال أو مجموعة من الموارد (, ports bandwidth) لفترة زمنية محددة.

2. 2. المتحكم (ONOS open network operating system):

وهو منصة تحكم مفتوحة المصدر موزعة تهدف الى تزويد إمكانية التوسعية والأداء والتوافريَة لمزودي الخدمة[11]. يقدم هذا المتحكم حالياً دعم جودة خدمة محدود، حيث يدعم آليات (open flow metering mechanisms) لكن هذه الميزة مطبقة بشكل نادر في المبدلات الحالية.

2. 3. المتحكم (Ryu):

تم برمجته باستخدام لغة بايثون، وميزته الرئيسة هي أنه يدعم جميع الإصدارات[12,13]، وبسبب تصميمه القائم على الوحدات (modules) ودعمه لجميع الإصدارات فإنَه يُستخدم من أجل النماذج الأولية السريعة في (SDN)، ويوفر هذا المتحكم واجهات (API) من أجل إدارة وتهيئة قوائم الانتظار في مبدلات (open vSwitch) باستخدام البروتوكول (OVSDB) (وهو المتحكم المستخدم في هذا البحث).

الدراسات المرجعية:

تناولت عدة أبحاث موضوع أداء الشبكات المعرفة بالبرمجيات (SDN) وجودة الخدمة. في المرجع [1] قام الباحثون باستخدام مسارات توجيه بديلة للتدفقات التي تتطلب جودة خدمة من خلال تثبيت قواعد التدفق في المبدلات التي يمكن أن تحقق عرض نطاق ترددي أعلى، ويقدم المؤلفون في [5] صيغة تعتمد على البرمجة الخطية تهدف الى التخفيف من فقدان الرزم لتدفقات الفيديو مع السماح ببعض التأخير، الفكرة هي توجيه تدفقات الفيديو عبر المسارات الجديدة التي تم حسابها من قبل الصيغة المقترحة، بينما يبقى توجيه التدفقات الأخرى دون جودة خدمة (best effort) عبر المسارات الأقصر النموذجية كالعادة.

في [7] قدم الباحثون طريقة لجدولة الرزم اعتماداً على ترويسة IP وعلى نوع حركة المرور(streaming, burst, something else)، والذي أدى إلى تحكم أكبر بعرض النطاق الترددي، وضمان عدم تأثير نوع مرور معين على نوع آخر.

قدم الباحثون في [8] طريقة من أجل توجيه رزم الفيديو بجودة أعلى من خلال الاستفادة من مزايا المسارات السريعة ووضَحوا كيف يمكن لهذه الطريقة أن تقدم إدارة حركة مرور مرنة إضافة إلى تحسين في جودة التجربة (QoE: Quality of Experience).

في[14] قدم الباحثون مجموعة من الاختبارات لأداء وحدة التحكم (RYU sdn controller) اعتماداً على مجموعة من البارامترات مثل عرض الحزمة وزمن التأخير (Round Trip Time). كذلك قدم الباحثون في [15] تحليلاً لاختبار أداء وحدة التحكم (RYU Controller) وتقييم بارامترات الأداء من حيث عرض الحزمة والإنتاجية والرجرجة والضياع...الخ. قدمت الباحثة في المرجع [16] دراسة وتقييم لشبكة (SDN) مع وجود متحكم (OpenDaylight) واحد ومن ثم مقارنتها مع شبكة أخرى تحوي ثلاثة متحكمات ومدى تأثير زيادة عدد المتحكمات في الشبكة على بارامترات الأداء.

تم في المرجع [17] تقديم آلية للحد من استهلاك عرض الحزمة بحيث لا تتجاوز عرض الحزمة مقداراً محدداً وفق المنفذ المتصل به، وتعتمد هذه الآلية على التدفق وتقوم بحجز مسبق لعرض الحزمة على طول المسار وتتطلب عملية إعداد مسبقة، كما أن زيادة عدد التدفقات يؤدي إلى زيادة عدد قواعد التدفق المطلوب تثبيتها في المتحكم، لذا فإن هذه الطريقة غير فعالة من أجل الشبكات الكبيرة.

قمنا في هذا البحث بتجميع التدفقات وتصنيفها إلى عدة أصناف اعتماداً على قيم (DSCP) الموجودة في ترويسة الرزمة (IP) وتخصيص قائمة انتظار لكل صنف وبالتالي عدم التعامل مع كل تدفق على حدى وهذا يقدم مرونة وقابلية توسع أكبر للشبكة.

في المرجع [18] قدمت الباحثة دراسة للبروتوكول (openflow) والمتحكم (pox) وإمكانية برمجة المتحكم لينفذ أكثر من تطبيق وبالتالي تحقيق مفهوم الشبكات المعرفة بالبرمجيات.

في المرجع [19] قدم الباحث فكرة دمج الشبكات الضوئية الحالية(ASON/GMPLS) ضمن نموذج الشبكات المعرفة بالبرمجيات ريثما يتم التوصل إلى مستوى تحكم موحد ومستقر يعتمد على البروتوكول(openflow).

في المرجع [20] قدم الباحث دراسة للشبكات المعرفة بالبرمجيات وناقش مشكلة هجوم خداع البروتوكول(ARP) وتم التقييم من خلال البارامترات (الإنتاجية والتأخير وتوافرية الشبكة).

النتائج والمناقشة:

1. السيناريو الأول:

نقوم بإنشاء الشبكة الموضحة في الشكل (2) والتي تتألف من مبدل واحد(s1) وأربع أجهزة (h1,h2,h3,h4) ومتحكم (C0) إما باستخدام التعليمة:

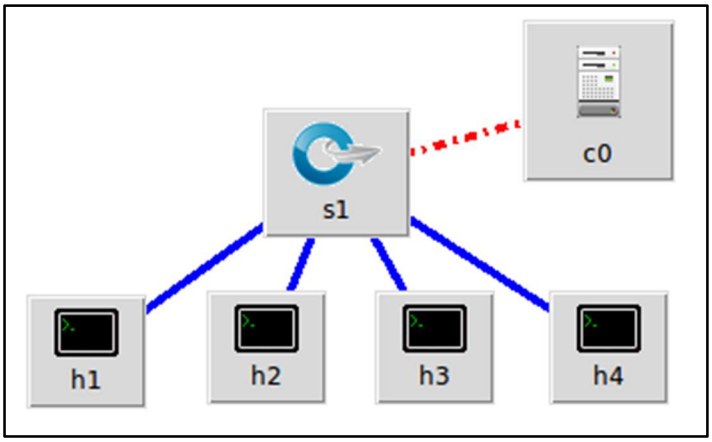
sudo mn –topo single,4 –switch ovsk –mac –controller remote

نقوم بضبط الإصدار (OpenFlow1.3) على المبدل من خلال التعليمة التالية:

switch: s1: ovs-vsctl set Bridge s1 protocols=OpenFlow13

يتم الآن تشغيل وظيفة (L2 switch) على المتحكم من خلال التعليمة الآتية:

controller: c0: cd ryu/; ./bin/ryu-manager --verbose ryu.app.simple_switch_13

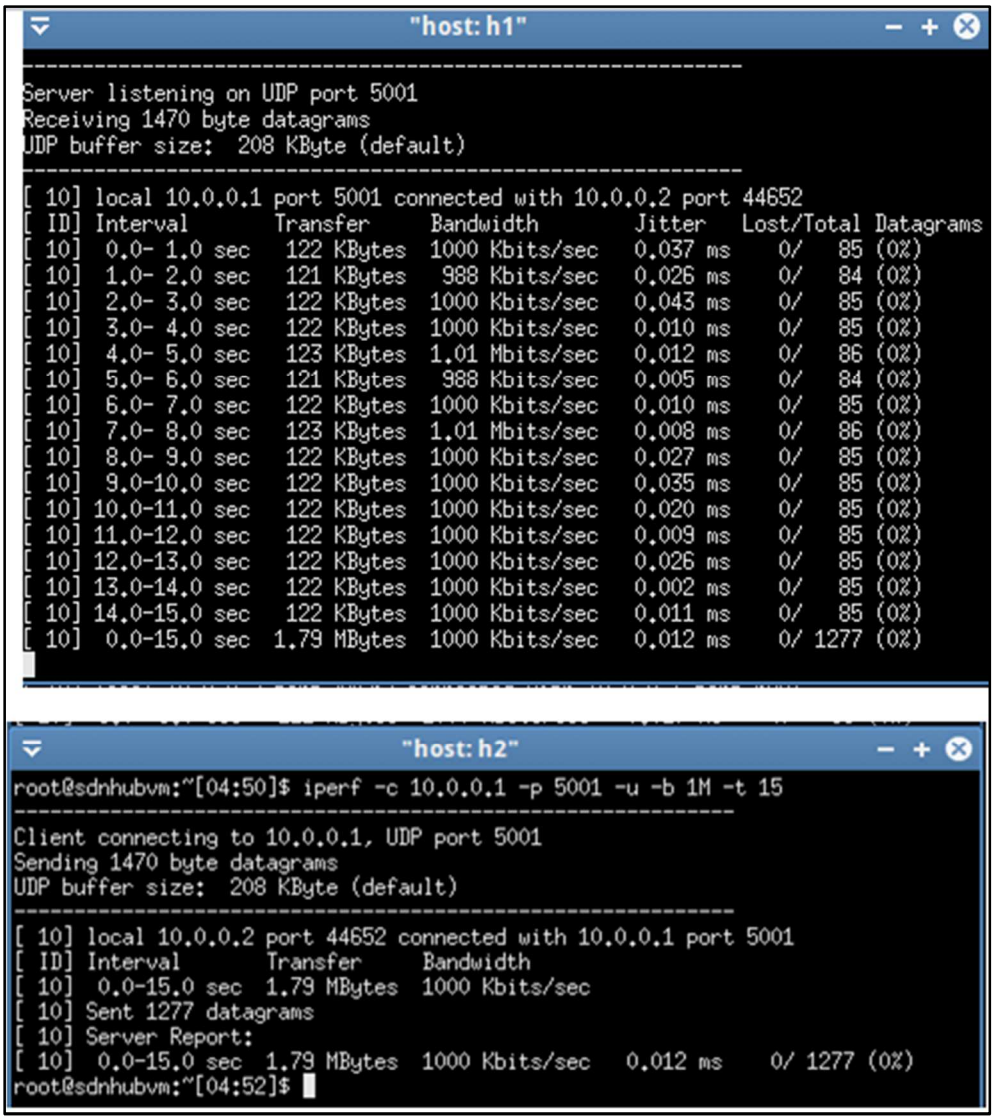


الشكل (2): طوبولوجيا الشبكة المستخدمة في السيناريو الأول

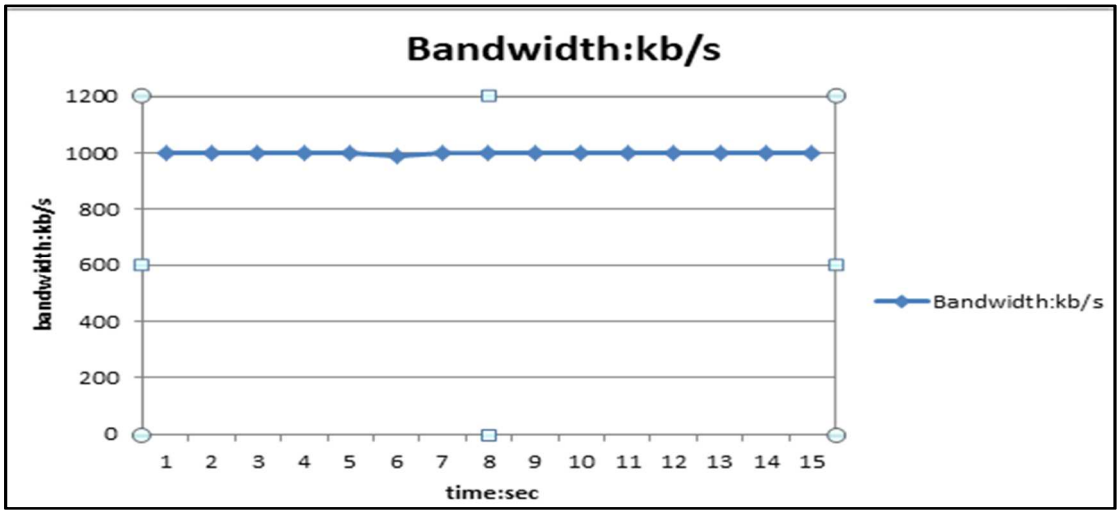
نقوم بتوليد مرور في الشبكة عن طريق الأداة (iperf) ، حيث يتم ضبط الجهاز (h1) ليعمل كمخدم(server) يستمع إلى المنفذ (5001)، والجهاز (h2) كزبون (client) يرسل (UDP traffic) بمعدل (1Mb/s) إلى الجهاز (h1:10.0.0.1) عبر المنفذ (5001) ويتم ذلك من خلال التعليمتين التاليتين على (console) الخاص بـ(h1,h2) :

h1: iperf –s –u –p 5001 –I 1
h2: iperf –c 10.0.0.1 –p 5001 –t 15 –u –b 1M

نلاحظ من نافذة (h1,h2) في الشكل (3) أنه تم استهلاك كامل عرض الحزمة المخصص وهو (1Mb/s)، حيث هنا لم يتم ضبط أي آلية للتحكم بعرض الحزمة، كما يوضح الشكل (4) الرسم البياني لعرض الحزمة المستهلك في السيناريو الأول.



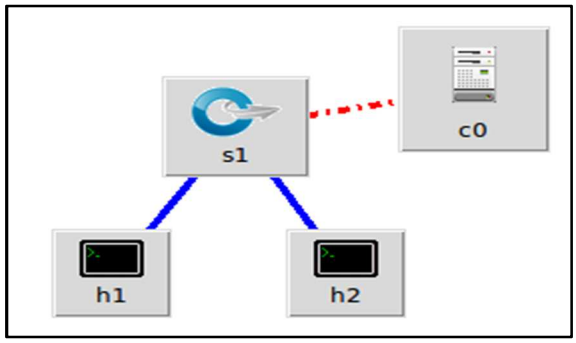
الشكل(3):عرض الحزمة المستهلك في السيناريو الأول



الشكل (4): الرسم البياني لعرض الحزمة المستهلك في السيناريو الأول

2. السيناريو الثاني: جودة الخدمة على مستوى كل تدفق (per-Flow Qos)

نقوم بإنشاء الشبكة الموضحة في الشكل(5) ثم نقوم في هذا السيناريو بتخصيص قوائم انتظار على منفذ المبدل (s1-eth1) من أجل التحكم بمعدل الإرسال وحجز نطاق ترددي معين في الشبكة، حيث نستخدم البروتوكول (OVSDB) من أجل إعداد قوائم الانتظار والتحكم بها.



الشكل (5): طبولوجيا الشبكة المستخدمة في السيناريو الثاني

نقوم في الخطوة التالية بضبط المبدل ليعمل على النسخة (openflow13) من خلال الاستعانة بالمكون (ovs-vsctl):

xtermS1: Ovs-vsctl set Bridge s1 protocols=OpenFlow13

ونضبط المبدل ليستمع الى المنفذ (6632) من أجل الوصول الى (OVSDB):

Xterms1: ovs-vsctl set-manager ptcp:6632

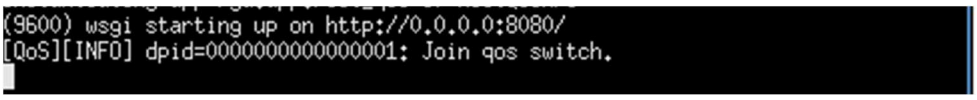
يتم استخدام الكود المصدري للتطبيق (simple_switch_13)، حيث يتم تعديله عن طريق الأمر (sed) ليسجل مدخل تدفق على جدول التدفق رقم 1 (table id:1) فنحصل على الملف (rest_qos.py) :

C0:sed'/OFPFlowMod(/,/)/s/)/,table_id=1)/ryu/ryu/app/simple_switch_13.py> ryu/ryu/app/ qos_simple_switch_13.py & cd ryu/; python ./setup.py install

أخيراً نقوم بتشغيل التطبيقات (rest_qos, qos_simple_switch_13 , rest_conf_switch) على المتحكم من خلال الأمر الآتي:

C0: cd ryu/; ./bin/ryu-manager ryu.app.rest_qos ryu.app.qos_simple_switch_13 ryu.app.rest_conf_switch

هنا تتم محاولة الاتصال بين المتحكم والمبدل، وعند نجاح عملية الاتصال تظهر الرسالة كما في الشكل(6) ليصبح المبدل يدعم جودة الخدمة.



الشكل(6): انضمام (s1) إلى الشبكة كمبدل يدعم جودة الخدمة

نقوم بضبط عنوان البروتوكول (ovsdb_addr) من أجل الوصول إلى البروتوكول (OVSDB):

Node: c0 (root): curl -X PUT -d '"tcp:127.0.0.1:6632"' http://localhost:8080/v1.0 /conf/switches /0000000000000001/ovsdb_addr

نقوم بتخصيص قائمتي انتظار إلى منفذ المبدل (s1- eth1) وتحديد الحد الأعظمي لحركة المرور المسموح نقلها من خلال كل قائمة، كما هو موضح في الجدول رقم (1) وذلك من خلال التعليمة الآتية والموضحة في الشكل (7):

C0: curl -X POST -d '{"port_name": "s1-eth1", "type": "linux-htb", "max_rate": "1000000", " queues": [{"max_rate": "400000"}, {"min_rate": "600000"}]}' http://localhost: 8080/qos/queue /0000000000000001

الجدول رقم (1): الحد الأدنى والأعظمي لقوائم الانتظار على منفذ المبدل (s1-eth0)

Queue id	Max rate(kb/s)	Min rate(kb/s)
0	400	-
1	1000000	600

الشكل (7): نتائج عملية تخصيص قوائم الانتظار في السيناريو الأول

يتم إضافة قاعدة تدفق إلى جدول تدفق المبدل (s1) كما هو موضح في الجدول رقم (2)، بحيث يتم وضع المرور الموجه إلى الجهاز الأول عبر عبر المنفذ (5002) ضمن قائمة الانتظار(1) وافتراضيا يتم تمرير المرور الموجه إلى الجهاز الأول عبر المنفذ (5001) إلى قائمة الانتظار (0):

Node: c0 (root): curl -X POST -d '{"match": {"nw_dst": "10.0.0.1", "nw_proto": "UDP", "tp_dst": "5002"}, " actions":{"queue": "1"}}' http://localhost:8080/qos/rules/ 0000000000000001

الجدول رقم (2): إضافة قاعدة تدفق إلى جدول تدفق المبدل (s1)

priority	Destination address	Destination port	Protocol	Queue id	Qos id
1	10.0.0.1	5002	Udp	1	1

للتحقق من أن عملية الضبط قد تمت يتم كتابة الأمر الموضح في الشكل (8):

Node: c0 (root): curl -X GET http://localhost:8080/qos/rules/0000000000000001

الشكل(8): عملية التحقق من إضافة قواعد التدفق لجدول تدفق المبدل(s1)

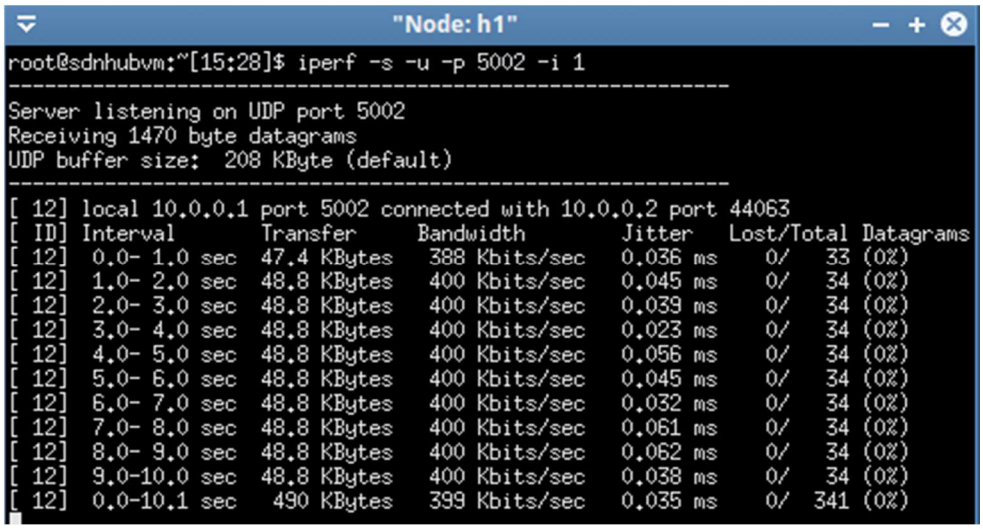
قياس عرض الحزمة باستخدام الأداة (iperf):

(h1): هو مخدم يستمع الى المنفذين (5001) (5002) مع البروتوكول (UDP).

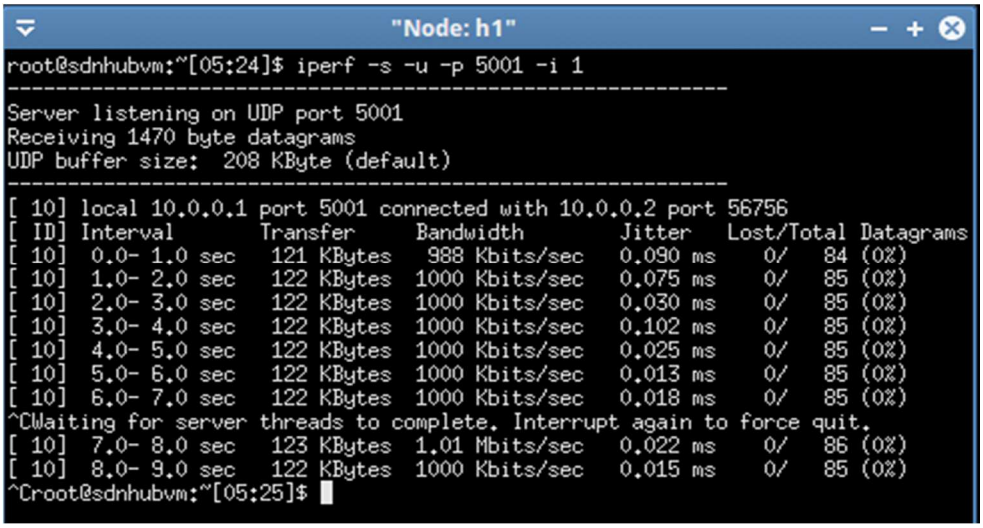
(h2): زبون يقوم بإرسال مرور (UDP traffic) الى السيرفر (h1) بمقدار (1Mb/s :5001) (1Mb/s:5002).

Node: h1(1) (root): iperf -s -u -i 1 -p 5001
Node: h1(1) (root): iperf -s -u -i 1 -p 5002
Node: h2(1) (root):iperf -c 10.0.0.1 -p 5001 -u -b 1M
Node: h2(2) (root): iperf -c 10.0.0.1 -p 5002 -u -b 1M

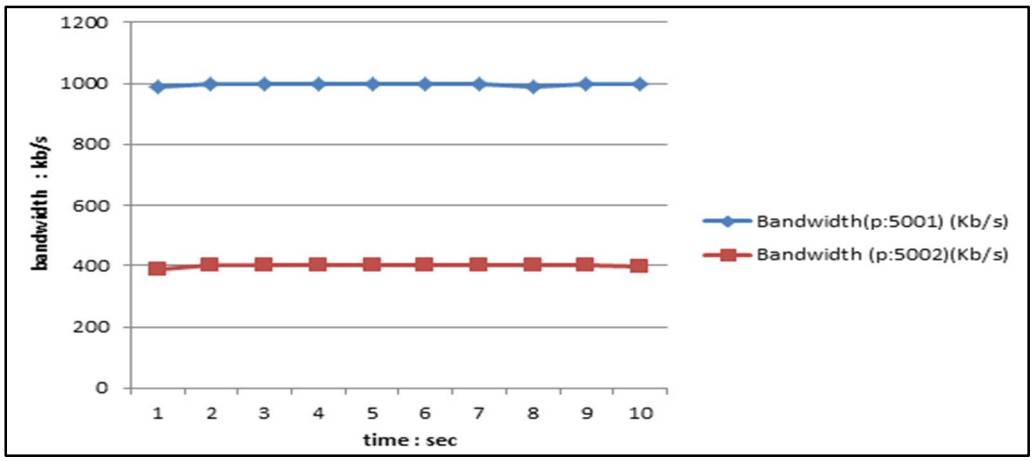
نلاحظ من الشكل (9) أن عرض الحزمة على المنفذ 5002 لم يتجاوز المعدل الأعظمي المحدد له وهو(400kb/s)، بينما عرض الحزمة على المنفذ 5001 الموضح في الشكل (10) هي بنفس معدل الإرسال (1Mb/s) ولم ينخفض تحت الحد الأدنى (600kb/s)، ويوضح الشكل (11) الرسم البياني لعرض الحزم المستهلك على المنفذين(5001،5002).



الشكل(9): عرض الحزمة المستهلك على المنفذ 5002



الشكل(10): عرض الحزمة المستهلك على المنفذ 5001



الشكل(11): الرسم البياني لعرض الحزمة المستهلك على المنفذين (5001,5002)

3. السيناريو الثالث: تنفيذ جودة الخدمة اعتماداً على تصنيف الخدمة باستخدام قيمة (DSCP differentiated service code) :

يتم في هذا السيناريو تقسيم التدفقات إلى عدة فئات اعتماداً على قيمة (**DSCP**) وهي عبارة عن 8 بتات موجودة في ترويسة (IPv4) ضمن حقل (type of service TOS) أو ضمن حقل (traffic class) في ترويسة (IPv6)، ومن ثم يتم تخصيص قوائم انتظار حسب صنف الخدمة الذي يشمل مجموعة من التدفقات والتي تملك نفس سوية جودة الخدمة المطلوبة.

نقوم في هذا السيناريو بتشغيل وظيفة الموجه(router) من خلال المتحكم (ryu) والبروتوكول (openflow)، حيث نقوم بإنشاء شبكة مؤلفة من مبدلين وكل مبدل يتصل به جهاز واحد كما في الشكل(12) حيث يتم إنشاء هذه الشبكة إما من خلال برنامج (miniedit) أو من خلال التعليمة الآتية :

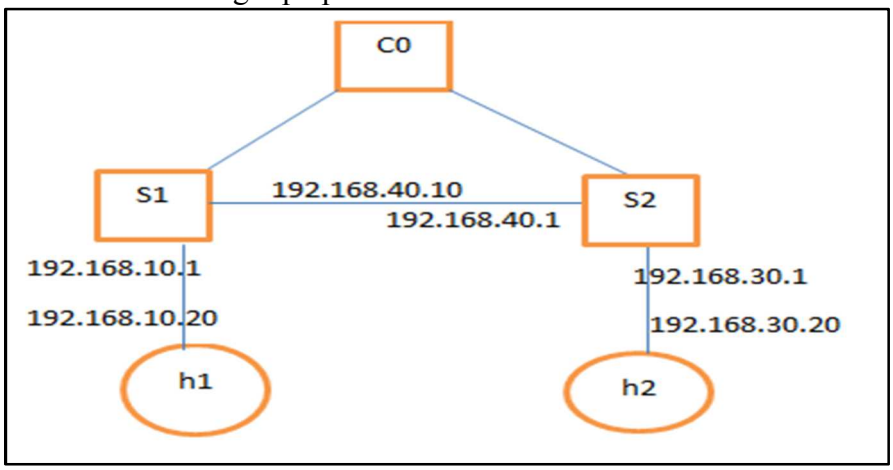
sudo mn --topo linear,2 --mac --switch ovsk --controller remote –x

ثم نقوم في الخطوة التالية بضبط المبدلات لتعمل على النسخ (openflow13):

xtermS1: Ovs-vsctl set Bridge s1 protocols=OpenFlow13
xtermS2: Ovs-vsctl set Bridge s2 protocols=OpenFlow13

ضبط المبدل(s1) ليستمع الى المنفذ (6632) من أجل الوصول الى (OVSDB):

Xterms1: ovs-vsctl set-manager ptcp:6632



الشكل(12): طوبولوجيا الشبكة المستخدمة في السيناريو الثالث

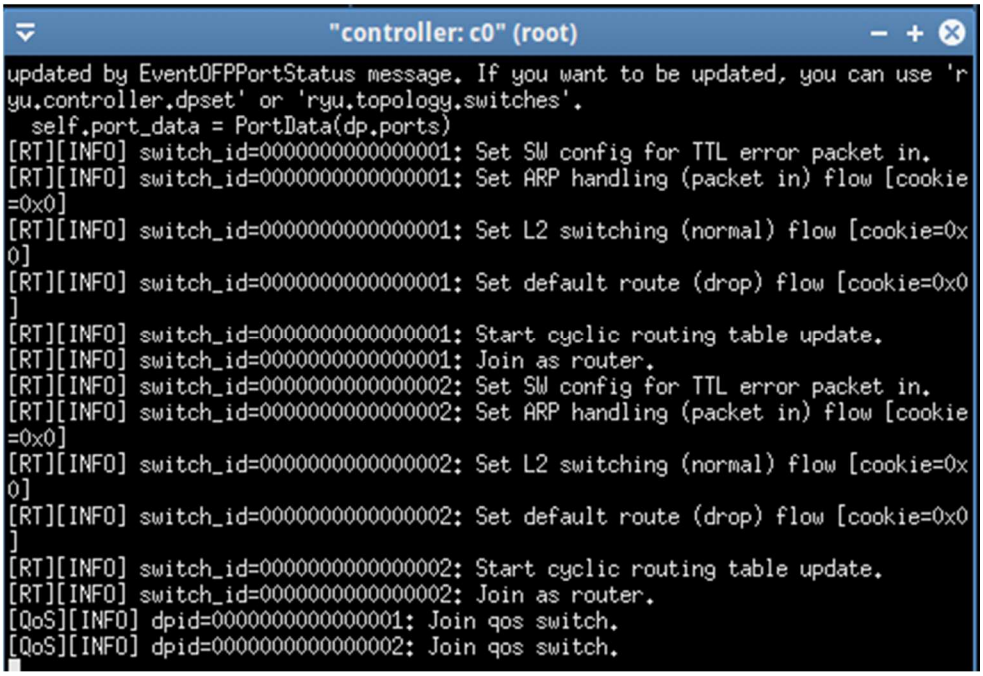
نقوم بتغيير عناوين (IP address) للمضيفين (h1,h2) كالتالي :

h1: ip addr del 10.0.0.1/8 dev h1-eth0
h1: ip addr add 192.168.10.20/24 dev h1-eth0
h2: ip addr del 10.0.0.2/8 dev h2-eth0
h2: ip addr add 192.168.30.20/24 dev h2-eth0

يتم تعديل الملف (rest_router.py) الموجود ضمن المسار (home/ubuntu/ryu/ryu/app) ليسجل مدخل تدفق أو قاعدة تدفق على جدول التدفق رقم 1 (table id:1) ونحصل على الملف (qos_rest_router.py) و أخيراً نقوم بتشغيل (rest_qos, qos_rest_router and rest_conf_switch) على المتحكم :

controller: c0 (root): ryu-manager ryu.app.rest_qos ryu.app.qos_rest_router ryu.app.rest_conf_switch

بعد نجاح الاتصال بين الموجه والمتحكم تظهر الرسالة الموضحة في الشكل(13) لتنضم المبدلات إلى الشبكة كموجهات تدعم جودة الخدمة:



الشكل(13): نتائج عملية الاتصال بين المتحكم والمبدلات

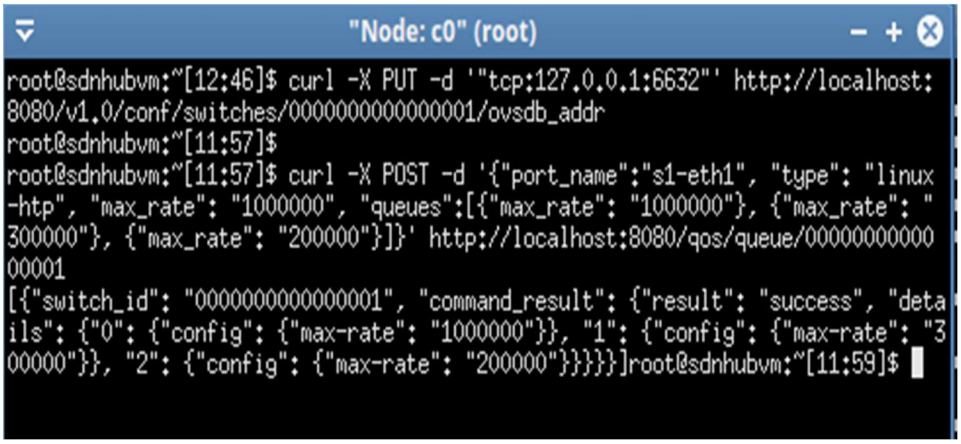
الجدول رقم (3): الحد الأدنى والأعظمي لقوائم الانتظار على منفذ المبدل (s1-eth0)

Queue id	Max rate(kb/s)	Min rate(kb/s)	class
0	1000	-	default
1	300	-	AF1
2	200	-	AF2

نقوم بضبط عنوان (ovsdb_addr) على المبدل (s1) من أجل الوصول الى بروتوكول (OVSDB):

Node:C0: curl -X PUT -d '"tcp:127.0.0.1:6632"' http://localhost:8080/v1.0/conf/switches /0000000000000001/ovsdb_addr

نقوم بتصنيف حركة المرور إلى ثلاث فئات وتخصيص قائمة انتظار لكل فئة ومن ثم تخصيص عرض حزمة محدد لكل فئة حسب احتياجاتها كما هو مبين في الجدول (3) وذلك باستخدام التعليمة التالية:



نقوم بضبط عناوين (ip addresses) لكل منفذ على المبدل (s1) وذلك من خلال الأوامر الآتية:

```
root@sdnhubvm:~[15:29]$ curl -X POST -d '{"address":"192.168.10.1/24"}' http://
localhost:8080/router/0000000000000001
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "det
ails": "Add address [address_id=1]"}]}]root@sdnhubvm:~[15:29]$ curl -X POST -d '
```

```
root@sdnhubvm:~[15:30]$ curl -X POST -d '{"address":"192.168.40.10/24"}' http:/
/localhost:8080/router/0000000000000001
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "det
ails": "Add address [address_id=2]"}]}]root@sdnhubvm:~[15:31]$
```

```
root@sdnhubvm:~[15:31]$ curl -X POST -d '{"gateway":"192.168.40.1"}' http://loc
alhost:8080/router/0000000000000001
[{"switch_id": "0000000000000001", "command_result": [{"result": "success", "det
ails": "Add route [route_id=1]"}]}]root@sdnhubvm:~[15:32]$
```

نقوم بضبط عناوين (ip addresses) لكل منفذ على المبدل (s2) وذلك من خلال الأوامر الآتية:

```
root@sdnhubvm:~[15:33]$ curl -X POST -d '{"address":"192.168.30.1/24"}' http://
localhost:8080/router/0000000000000002
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "det
ails": "Add address [address_id=1]"}]}]root@sdnhubvm:~[15:34]$
```

```
root@sdnhubvm:~[15:36]$ curl -X POST -d '{"gateway":"192.168.40.10"}' http://lo
calhost:8080/router/0000000000000002
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "det
ails": "Add route [route_id=1]"}]}]root@sdnhubvm:~[15:36]$
```

```
root@sdnhubvm:~[15:34]$ curl -X POST -d '{"address":"192.168.40.1/24"}' http://
localhost:8080/router/0000000000000002
[{"switch_id": "0000000000000002", "command_result": [{"result": "success", "det
ails": "Add address [address_id=2]"}]}]root@sdnhubvm:~[15:35]$
```

نقوم بتعيين مسار افتراضي لكل جهاز :

h1: ip route add default via 192.168.10.1
h2: ip route add default via 192.168.30.1

نقوم بثبيت قاعدتي تدفق في المبدل (s1) كما هو موضح في الجدول(4) :

قاعدة 1: الرزم المصنفة مع قيمة (DSCP=AF31 (26) توجه إلى قائمة الانتظار رقم (1).

قاعدة 2: الرزم المصنفة مع قيمة (DSCP=AF32 (28) توجه إلى قائمة الانتظار رقم (2).

```
Curl –X POST –d '{"match": {"ip_dscp":"26"},"actions": "queue":"1"}}'
http://localhost:8080/qos/rules/0000000000000001
Curl –X POST –d '{"match": {"ip_dscp":"28"},"actions": "queue":"2"}}'
http://localhost:8080/qos/rules/0000000000000001
```

الجدول رقم (4): إضافة قاعدتي تدفق إلى جدول تدفق المبدل (s1)

priority	DSCP	Queue id	Qos id
1	26(AF31)	1	1
1	28(AF32)	2	2

ثم نقوم بتثبيت القاعدتين على المبدل (s2) كما هو موضح في الجدول (5):

قاعدة 1: الرزم الموجهة إلى (h1) على المنفذ (5002) ضع عليها قيمة (DSCP=26).

قاعدة 2: الرزم الموجهة إلى (h1) على المنفذ (5003) ضع عليها قيمة (DSCP=28).

```
Curl -X POST –d '{"match": {"nw_dst": "192.168.10.20", "nw_proto": "UDP", "tp_dst":"5002"},"actions":{"mark":"26"}}'
http://localhost:8080/qos/rules/0000000000000002
Curl -X POST –d '{"match": {"nw_dst": "192.168.10.20", "nw_proto": "UDP", "tp_dst":"5003"},"actions":{"mark":"28"}}'
http://localhost:8080/qos/rules/0000000000000002
```

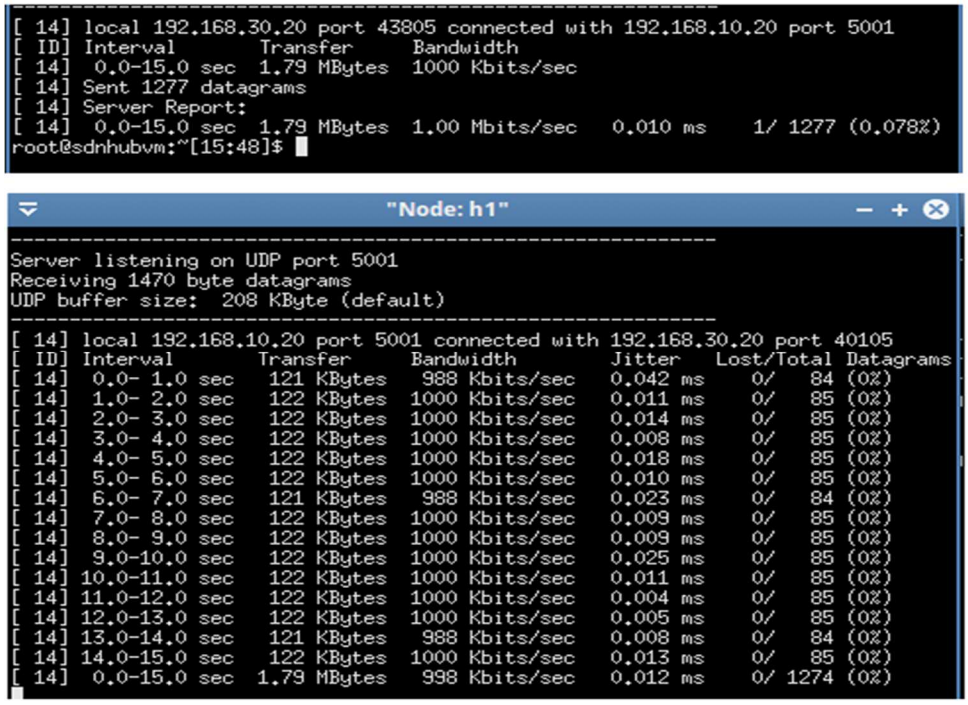
الجدول رقم (5): إضافة قاعدتي تدفق إلى جدول تدفق المبدل (s2)

priority	Destination address	Destination port	DSCP	protocol	Qos id
1	192.168.10.20	5002	26	UDP	1
1	192.168.10.20	5003	28	UDP	2

قياس عرض الحزمة:

تم استخدام الأداة (iperf) من أجل قياس عرض الحزمة: حيث (h1): هو خادم يستمع الى المنافذ (5001) (5002) (5003) مع بروتوكول (UDP) و(h2): زبون يقوم بإرسال مرور (UDP traffic) الى الخادم (h1) بمقدار (1Mb/s :5001) (1Mb/s :5002) (1Mb/s :5003).

يوضح الشكل (14) (15) (16) عرض الحزمة المستهلك على المنافذ (5001) (5003) (5002) على التوالي، كما يوضح الشكل (17) الرسم البياني لعرض الحزمة المستهلك على المنافذ الثلاثة(5001)(5002)(5003).



الشكل (14): عرض الحزمة المستهلك على المنفذ 5001

```
[ 14] local 192.168.30.20 port 52840 connected with 192.168.10.20 port 5003
[ ID] Interval       Transfer     Bandwidth
[ 14]  0.0-15.1 sec   369 KBytes   200 Kbits/sec
[ 14] Sent 257 datagrams
[ 14] Server Report:
[ 14]  0.0-15.1 sec   369 KBytes   200 Kbits/sec   0.045 ms    0/  257 (0%)
root@sdnhubvm:~[15:53]$
```

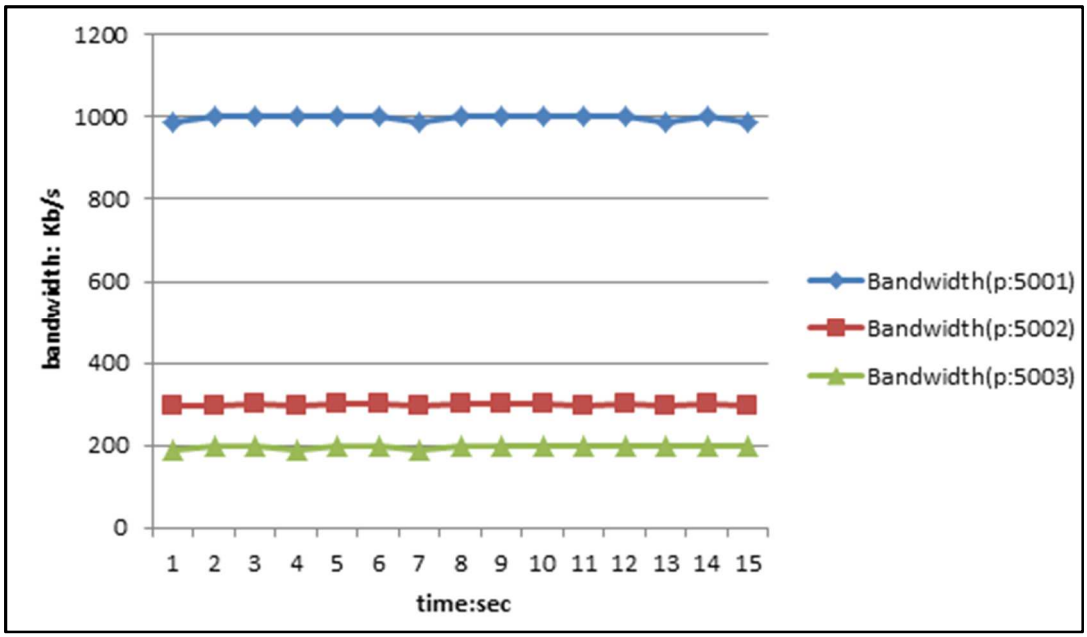
"Node: h1"

```
Server listening on UDP port 5003
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
[ 14] local 192.168.10.20 port 5003 connected with 192.168.30.20 port 39407
[ ID] Interval       Transfer     Bandwidth        Jitter   Lost/Total Datagrams
[ 14]  0.0- 1.0 sec  23.0 KBytes   188 Kbits/sec   0.007 ms    0/   16 (0%)
[ 14]  1.0- 2.0 sec  24.4 KBytes   200 Kbits/sec   0.032 ms    0/   17 (0%)
[ 14]  2.0- 3.0 sec  24.4 KBytes   200 Kbits/sec   0.030 ms    0/   17 (0%)
[ 14]  3.0- 4.0 sec  23.0 KBytes   188 Kbits/sec   0.021 ms    0/   16 (0%)
[ 14]  4.0- 5.0 sec  24.4 KBytes   200 Kbits/sec   0.051 ms    0/   17 (0%)
[ 14]  5.0- 6.0 sec  24.4 KBytes   200 Kbits/sec   0.041 ms    0/   17 (0%)
[ 14]  6.0- 7.0 sec  24.4 KBytes   200 Kbits/sec   0.038 ms    0/   17 (0%)
[ 14]  7.0- 8.0 sec  23.0 KBytes   188 Kbits/sec   0.032 ms    0/   16 (0%)
[ 14]  8.0- 9.0 sec  24.4 KBytes   200 Kbits/sec   0.037 ms    0/   17 (0%)
[ 14]  9.0-10.0 sec  24.4 KBytes   200 Kbits/sec   0.027 ms    0/   17 (0%)
[ 14] 10.0-11.0 sec  24.4 KBytes   200 Kbits/sec   0.030 ms    0/   17 (0%)
[ 14] 11.0-12.0 sec  23.0 KBytes   188 Kbits/sec   0.031 ms    0/   16 (0%)
[ 14] 12.0-13.0 sec  24.4 KBytes   200 Kbits/sec   0.039 ms    0/   17 (0%)
[ 14] 13.0-14.0 sec  24.4 KBytes   200 Kbits/sec   0.042 ms    0/   17 (0%)
[ 14] 14.0-15.0 sec  24.4 KBytes   200 Kbits/sec   0.041 ms    0/   17 (0%)
[ 14]  0.0-15.1 sec   363 KBytes   197 Kbits/sec   0.037 ms    0/  253 (0%)
```

الشكل (15): عرض الحزمة المستهلك على المنفذ 5003

```
[ 14] local 192.168.30.20 port 54003 connected with 192.168.10.20 port 5002
[ ID] Interval       Transfer     Bandwidth
[ 14]  0.0-15.1 sec   551 KBytes   300 Kbits/sec
[ 14] Sent 384 datagrams
[ 14] Server Report:
[ 14]  0.0-15.1 sec   551 KBytes   300 Kbits/sec   0.023 ms    0/  384 (0%)
root@sdnhubvm:~[15:52]$
```

الشكل (16): متوسط عرض الحزمة المستهلك على المنفذ 5002



الشكل (17): الرسم البياني لعرض الحزمة المستهلك على المنافذ (5001,5002,5003) في السيناريو الثالث

من النتائج السابقة نجد أن: حركة المرور (AF31) التي تم توجيهها إلى المنفذ (5002) لم تتجاوز عرض النطاق الترددي المخصص لها (300kb/s)، وكذلك حركة المرور (AF32) التي تم توجيهها الى المنفذ (5003) لم تتجاوز عرض النطاق الترددي المخصص لها (200kb/s).

الاستنتاجات والتوصيات:

قمنا في هذا البحث تنفيذ طريقتين لآليات جودة الخدمة من أجل التحكم بعرض الحزمة باستخدام البروتوكول (openflow) والمتحكم (RYU) ، وقمنا بالمقارنة بين الطريقتين، ففي الطريقة الأولى والتي استخدمتها الباحثة في المرجع رقم (17) تم تزويد جودة الخدمة لكل تدفق (QoS per-flow) بحيث لا تتجاوز عرض الحزمة مقداراً محدداً وفق المنفذ المتصل به، لكن هذه الآلية تعتمد على كل تدفق على حدى وتقوم بحجز مسبق لعرض الحزمة على طول المسار وتتطلب عملية إعداد مسبقة، كذلك إن زيادة عدد التدفقات يؤدي إلى زيادة عدد قواعد التدفق المطلوب إضافتها إلى جداول تدفق المبدل، وبالتالي هذه الطريقة غير مناسبة للتعامل مع التدفقات الكثيرة. بينما تم في الطريقة الثانية التي اقترحناها تجميع التدفقات وتصنيفها إلى عدة أصناف اعتماداً على قيمة (DSCP) للرزمة وقمنا بتوضيح كيف يمكن لهذه الطريقة التحكم بعرض الحزمة وفقاً لحاجة المستخدم وبالتالي ضبط موارد الشبكة والاستخدام الامثل لها.

يناسب هذا البحث التطبيقات التي تتطلب عرض حزمة كبير مثل تطبيقات الوسائط المتعددة، بحيث يتم تحديد حد أعظمي لمعدل التدفق لتدفقات معينة بحيث لا تؤثر على التدفقات الأخرى في الشبكة، وكعمل مستقبلي من الممكن استخدام جداول العدادات (meter tables) الذي يسمح بمراقبة معدل دخول التدفق ومن ثم القيام بإجراءات بناءً على هذا المعدل، على عكس الأرتال (queues) التي تعد خاصية لمنفذ مبدل ما (switch port)، فإن جدول العدادات خاص بمدخلات التدفق (flow entries).

References:

[1] ALHARBI, F, *SDN-Based Mechanisms For Provisioning Quality Of Service To Selected Network Flows*. Theses and dissertations –computer science. University of Kentucky,2018,6-10.

[2] KREUTZ, D; RAMOS, F; VERISSIMO, P;ROTHENBERG, E; AZODOLMOLKY, S; and UHLIG, S, *Software-defined networking: a comprehensive survey*, Proc. IEEE, Vol. 103, No.1, 2015, 14–76.

[3] KARAKUS, M and DURRESI, A, *Quality of Service (QoS) in Software Defined Networking (SDN): A survey*. Journal of Network and Computer Applications, Vol. 80, No.1, 2017, 200–218.

[4] Open Networking Foundation. *Software-defined networking (sdn) definition*, 03 2017. https://www.opennetworking. org/sdn-definition

[5] GIVANLAR, S; PARLAKISIK, M; TEKALB, A and GORKIMLI, B, *A qos-enabled openow environment for scalable video streaming*, GLOBECOMWorkshops (GC Wkshps),2010 IEEE.

[6] LOZANO, J.E; RIVERA, R; and NIETO, J, *Quality of Service in Software Defined Networks for Scientific Applications: Opportunities and Challenges*. Program Comput Soft **46,** 561–568 (2020).

[7] CHATO, O; and EMMANUEL, M, *An Exploration of Various Quality of Service Mechanisms in an OpenFlow and Software Defined Networking Environment in Terms of Latency Performance*. 2016 International Conference on Information Science and Security (ICISS), IEEE, Pattaya,2016

[8] BHAT, D; ANDERSON, J; RUTH, P; ZINK, M; and KEAHY, K, *Application-based QoE support with P4 and OpenFlow*. IEEE INFOCOM 2019 -IEEE Conference on Computer Communications Workshops(INFOCOM WKSHPS), IEEE,PARIS,2019

[9] OpenDaylight Project. < https://www.opendaylight.org>

[10] ONOS Project. <http://onosproject.org>

[11] Floodlight, 03 2017 < http://www.projectfloodlight.org/ floodlight>

[12] Ryu development team, ryu Documentation Release 4.34, Jan 20, 2021

[13] Ryu. Retrieved from <http://osrg.github.com/ryu >.

[14] HIMANSHI, B; and SHALLI, R, *Performance Evaluation of QoS metrics in Software Defined Networking using Ryu Controller*, Conference Paper *in* IOP Conference Series Materials Science and Engineering · January 2021.

[15] ISLAM,T; and ISLAM, N, *Node To Node Performance Evaluation Through Ryu Sdn Controller,* Springer science, 2020.

[16] NADH,R;AL-ATIKI,T and SENIO,R, *A Study and Performance Evaluation of OpenDaylight Controller in Software Defined Networks (SDN)*, Engineering Sciences Series, Tishreen University Journal**,** Vol. 42 ,No. 3 ,2020.

[17] AHMAD,S,A; and AFRAA,M, *Improving BandWidth Utilization in Software Defined Networks (SDN)*, Engineering Sciences Series, Tishreen University Journal**,** Vol.42 ,No. 5 ,2020.

[18] AHMAD,S,A; and AFRAA,M, *A Study of OpenFlow Protocol and POX Controller in Software Defined Networks(SDN) Using Mininet*, Engineering Sciences Series, Tishreen University Journal**,** Vol. 41 ,No. 1 ,2019.

[19] KHALIFA,J;JANOUD,E and HASAN,Y,*Control Plane Integration of Automatically Switched Optical Network(ASON)\Generalized Multi-Protocol Label Switching (GMPLS) Networks into Software Defined Networking(SDN)\OpenFlow Networks,* Engineering Sciences Series, Tishreen University Journal**,** Vol. 42 ,No. 5 ,2020.

[20] MALLA,B;and ABD AL-HAMID,M, *Studying the ARP Spoofing Attack Effect on SDN Networks*, Engineering Sciences Series, Tishreen University Journal**,**Vol. 42 ,No. 2 ,2020.