



جامعة تشرين

كلية الهندسة الميكانيكية والكهربائية

هندسة الاتصالات والإلكترونيات

السنة الخامسة

مشروع مادة برمجة الشبكات

OpenFlow Technology SDN

إعداد الطلاب

ربوع أحمد صبوح

الين عبد العزيز صالح

2493

2417

إشراف الدكتور

د. مهند عيسى

فهرس المحتويات

3 ملخص البحث
4 المقدمة
5 الفصل الأول : الشبكات المعرفة برمجيا SDN
9 الفصل الثاني : تطبيق عملي
16 المراجع

ملخص البحث

غالبًا ما يشار إلى الشبكات المعرفة برمجيا على أنها فكرة ثورية جديدة في شبكات الكمبيوتر ، تقوم بتبسيط التحكم في الشبكة وإدارتها وتمكين الابداع بشكل كبير من خلال قابلية برمجة الشبكة. عادةً ما يتم إنشاء شبكات الكمبيوتر من عدد كبير من أجهزة الشبكة مثل المبدلات Switches والموجهات Routers والجدران النارية Firewalls مع العديد من البروتوكولات المعقدة والتي يتم تنفيذها ومضمنة عليها. مهندسو الشبكات مسؤولون عن تكوين السياسات للاستجابة لمجموعة واسعة من أحداث الشبكة وسيناريوهات التطبيق. يقومون يدويًا بتحويل هذه السياسات عالية المستوى إلى أوامر تكوين منخفضة المستوى. غالبًا ما يتم إنجاز هذه المهام المعقدة للغاية من خلال الوصول إلى أدوات محدودة للغاية. في SDN يتم حل هذه المشكلات بسهولة مطلقة.

المقدمة

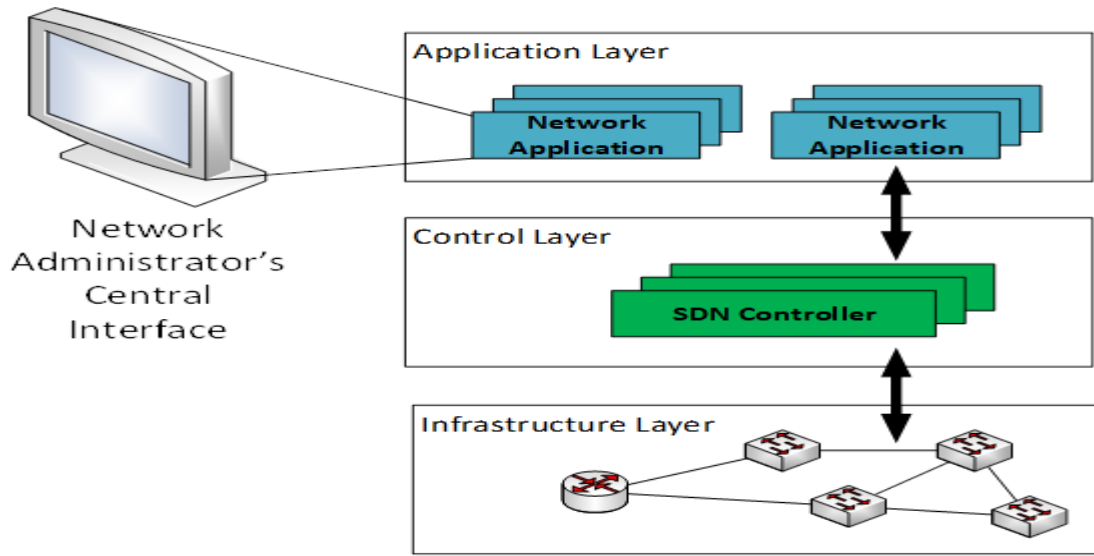
الحواسيب والهواتف المحمولة، وتطبيقات الإنترنت المتقدمة في عالم اليوم التي نستخدمها كلها تعتمد على الاتصالات الرقمية الكلاسيكية التي يكون شكل البيانات فيها سلاسل من الأصفار والواحدات. الإنترنت الكلاسيكي بشكله الحالي نشأ من تكامل الرياضيات ونظرية المعلومات لكلود شانون. لكن الإنترنت في هذه الأيام يشكل بيئة غنية للتنصت والاحتيال وسرقة المعلومات. وهذا يشكل أمراً خطيراً على خصوصية المستخدمين حيث بالكاد يمر أسبوع دون ورود تقارير عن بعض عمليات الاختراق الضخمة الجديدة التي كشفت عن كميات هائلة من المعلومات الحساسة ، من تفاصيل بطاقات الائتمان للأشخاص والسجلات الصحية إلى الملكية الفكرية للشركات. يجبر التهديد الذي تشكله الهجمات الإلكترونية الحكومات والجيش والشركات على استكشاف طرق أكثر أماناً لنقل المعلومات. دفع هذا الباحثين إلى التحول إلى تقنيات جديدة أكثر أماناً من وفعالية لإدارة الشبكات.

نقدم في حلقة البحث هذه لمحة عن الشبكات المعرفة برمجيا SDN وبروتوكولها الأساسي OpenFlow على فصلين. في الفصل الأول نتحدث عن بنية SDN ورسائل OpenFlow. أما في الفصل الثاني تطبق عملي على OpenFlow باستخدام لغة البرمجة بايثون وباستخدام المحاكى الشهير Mininet.

الفصل الأول

- الشبكات المعرفة برمجيا SDN Software Defined Network :

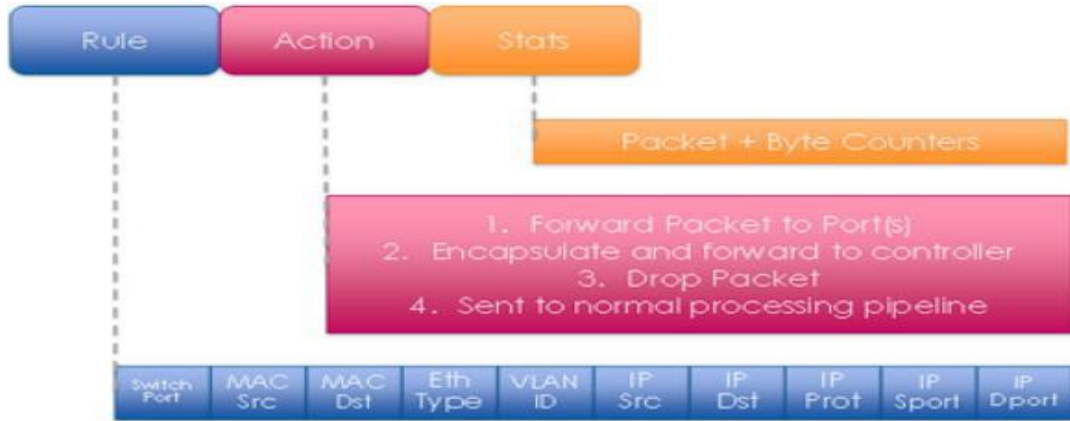
إن التحكم في إدارة الشبكة وضبط الأداء يمثلان مهامًا صعبة للغاية وعرضة للكثير من الأخطاء، يشكل استقرار الأداء في الشبكات تحديًا كبيرًا يتطلب الكثير من الجهد والمعرفة لمواجهة الكثير من المخاطر الأمنية التي تزداد يوما بعد يوم بسبب التضخم الآسي لتطبيقات الإنترنت وتأثيرها على جوانب مختلفة من حياتنا فأصبح تطوير الإنترنت من حيث البنية التحتية المادية إلى جانب بروتوكولاتها وأدائها. نظرًا لأن التطبيقات الجديدة أكثر تعقيدًا، يبدو أن الوضع الراهن للإنترنت غير قادر على التطور لمواجهة التحديات الناشئة. الشبكات المعرفة برمجيا SDN تسهل حل الكثير من هذه المشكلات حيث تجعل إدارة الشبكة أكثر ديناميكية من خلال قابلية برمجة كل أجهزة الشبكة لأداء مهام محددة بعناية وتفصيل.



الشكل 1-1 بنية SDN

تم اقتراح مفهوم الشبكات القابلة للبرمجة كطريقة لتسهيل تطور الشبكة. على وجه الخصوص ، يعد SDN نموذجًا جديدًا للشبكات ، يتم فيه فصل أجهزة إعادة التوجيه عن قرارات التحكم ما يجعل التعامل مع البنية التحتية للشبكات الأساسية من وجهة نظر التطبيق ليس من وجهة نظر الموجه. يوفر هذا الفصل بنية شبكة أكثر مرونة وقابلة للبرمجة وأكثر فعالية من حيث التكلفة. توفر بنية SDN مجموعة من واجهات برمجة التطبيقات (APIs) التي تبسط تنفيذ الخدمات الشبكية المشتركة (التوجيه والبنث المتعدد والأمان والتحكم في الوصول وإدارة النطاق الترددي والتحكم بالحركة وجودة الخدمة وكفاءة الطاقة ، وأشكال مختلفة من إدارة السياسات). في SDN ، يتمركز ذكاء الشبكة في وحدات التحكم القائمة على البرامج (في مستوى التحكم) ،

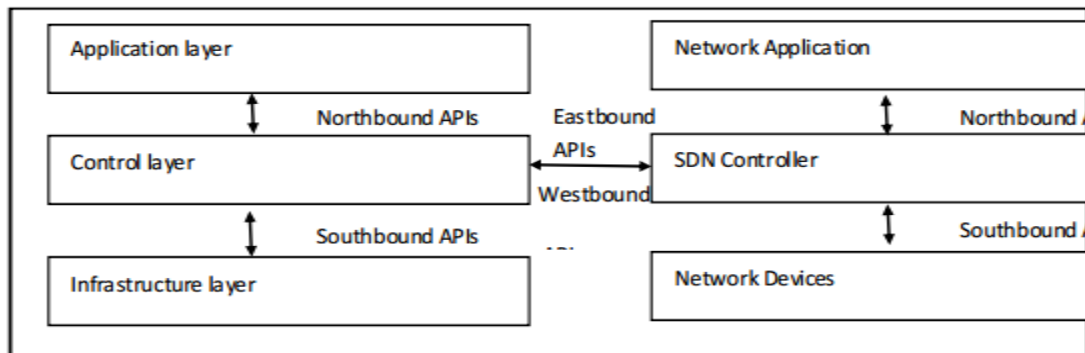
وتصبح أجهزة التوجيه تعامل مع حزم بسيطة (مستوى البيانات) التي يمكن برمجتها عبر واجهة GUI بشكل سهل. يُطلق على أحد التطبيقات الأساسية لهذه الواجهة اسم OpenFlow.



يُتيح فصل أجهزة إعادة التوجيه عن المتحكمات نشرًا أسهل للبروتوكولات والتطبيقات الجديدة ، وهيكلية الشبكة المباشرة وإدارتها. بدلاً من فرض السياسات وتشغيل البروتوكولات على كل أجهزة الشبكة المختلفة ، يتم اختصار الشبكة إلى أجهزة توجيه بسيطة ووحدة تحكم لاتخاذ القرار. تتكون أجهزة إعادة التوجيه مما يلي:

1. جدول توجيه يحتوي على مدخلات توجيه يتكون من قواعد وإجراءات مطابقة تتخذ على التدفقات النشطة.
 2. بروتوكول طبقة النقل الذي يتفاوض مع وحدة التحكم حول الإدخالات الجديدة غير الموجودة حاليًا في جدول التوجيه.
- مكونات النظام:

يتكون أي نظام SDN من مبدل SDN (SDN Switch) يعرف أيضًا ب OpenFlow Switch و SDN Controller بالإضافة إلى الواجهات الموجودة على وحدة التحكم للاتصال بأجهزة إعادة التوجيه ، وعادةً ما تكون الواجهة الجنوبية (OpenFlow) وواجهة تطبيقات الشبكة (الواجهة الشمالية) هي الوحدات الأساسية في SDN.



الشكل 1-2 مكونات نظام SDN

غالبًا ما يتم تمثيل المحولات في SDN كأجهزة إعادة توجيه أساسية يمكن الوصول إليها عبر واجهة مفتوحة ، حيث يتم إلغاء تحميل منطق التحكم والخوارزميات إلى وحدة تحكم.

لا تحتوي مبدلات Pure OpenFlow على ميزات قديمة أو تحكم داخلي ، وتعتمد تمامًا على المتحكم في قرارات إعادة التوجيه. تدعم مبدلات OpenFlow الهجينة بالإضافة إلى التشغيل والبروتوكولات التقليدية. معظم المبدلات التجارية المتاحة اليوم هجينة. يتكون مبدل OpenFlow من جدول تدفق يقوم بإجراء بحث عن الحزمة وإعادة توجيهها. يحتوي كل جدول تدفق في المحول على مجموعة من إدخلات التدفق تتكون من:

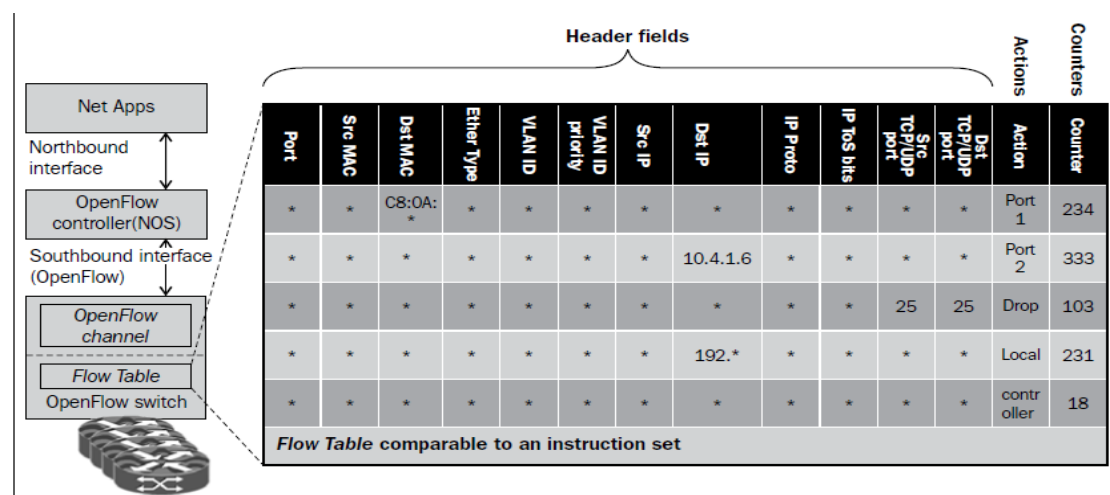
1. Headers تقوم بعملية مطابقة المعلومات الموجودة في رأس الحزمة ومنفذ الإدخال والبيانات المستخدمة لمطابقة الحزم الواردة.

2. عدادات ، تُستخدم لجمع الإحصائيات الخاصة بالتدفق المعين ، مثل عدد الحزم المتلقاة ، وعدد البايتات ، ومدة التدفق.

3. مجموعة من التعليمات أو الإجراءات التي سيتم تطبيقها بعد المطابقة التي تحدد كيفية التعامل مع الحزم المطابقة. على سبيل المثال ، قد يكون الإجراء هو إعادة توجيه حزمة إلى منفذ محدد.

يمكن مقارنة النظام المنفصل في SDN (OpenFlow) ببرنامج ونظام تشغيل في نظام حاسوبي عادي. في SDN ، توفر وحدة التحكم (أي نظام تشغيل الشبكة) واجهة برمجية للشبكة ، حيث يمكن كتابة التطبيقات لأداء مهام التحكم والإدارة وتقديم وظائف جديدة. تفترض طريقة العرض هذه أن عنصر التحكم مركزي وأن التطبيقات تتم كتابتها كما لو كانت الشبكة نظامًا واحدًا. في حين أن هذا يبسط تنفيذ السياسات ومهام الإدارة ، يجب الحفاظ على الارتباطات بين عنصر التحكم وعناصر إعادة توجيه الشبكة. كما هو موضح في الشكل التالي ، يجب أن تقوم وحدة التحكم بالعمل كنظام تشغيل شبكة بتنفيذ واجهتين على الأقل: واجهة متجهة جنوبياً (على سبيل المثال ، OpenFlow) تسمح للمبدلات بالاتصال بوحدة التحكم وواجهة متجهة شمالاً تقدم واجهة قابلة للبرمجة API للتحكم في الشبكة.

التطبيقات / الخدمات. حقول العنوان (حقول المطابقة) موضحة في الشكل التالي. يحتوي كل إدخال لجدول التدفق على قيمة محددة.



الشكل 3-1 حقول المطابقة

يجب أن تدعم محولات OpenFlow إعادة توجيه الحزمة إلى المنافذ الفعلية والمنافذ الافتراضية التالية:

- ❖ ALL: إرسال الحزمة إلى كافة الواجهات ، باستثناء المنفذ الوارد
- ❖ CONTROLLER: قم بتغليف الحزمة وإرسالها إلى وحدة التحكم
- ❖ LOCAL: أرسل الحزمة إلى مكس الشبكة المحلية للمحول
- ❖ TABLE (فقط لرسالة الحزمة): نفذ الإجراء في جدول التدفق
- ❖ IN_PORT: أرسل الحزمة إلى منفذ الإدخال
- ❖ Drop: يشير هذا إلى أنه يجب إسقاط جميع الحزم المتطابقة. يعتبر إدخال التدفق بدون إجراء محدد بمثابة إجراء إسقاط.

يحدث الاتصال بين وحدة التحكم والمحول باستخدام بروتوكول OpenFlow ، حيث يمكن تبادل مجموعة من الرسائل المحددة عبر قناة آمنة. القناة الآمنة هي الواجهة التي تربط كل مبدل OpenFlow بوحدة تحكم. يتم بدء اتصال بروتوكول أمان طبقة النقل (TLS) بوحدة التحكم المحددة بواسطة المستخدم بواسطة المبدل . يقوم المحول ووحدة التحكم بالمصادقة المتبادلة عن طريق تبادل الشهادات الموقعة بواسطة مفتاح خاص بالموقع.

لا يتم فحص حركة المرور من وإلى القناة الآمنة مقابل جدول التدفق ، وبالتالي يجب أن يحدد المحول حركة المرور الواردة على أنها محلية قبل التحقق منها مقابل جدول التدفق. في حالة فقد المحول الاتصال بوحدة التحكم ، نتيجة لانتهاء مهلة الطلب أو مهلة جلسة TLS أو أي انقطاع آخر ، يجب أن يحاول الاتصال بواحد أو أكثر من وحدات التحكم الاحتياطية. إذا فشل عدد من المحاولات للاتصال بوحدة تحكم ، يجب أن يدخل المحول في وضع الطوارئ ويعيد تعيين اتصال TCP الحالي على الفور. ثم يتم إملء عملية المطابقة من خلال إدخال جدول تدفق الطوارئ (المميزة بمجموعة بت الطوارئ). يجب ضبط قيمة المهلة في رسائل تدفق الطوارئ على صفر ويجب أن يرفض المبدل الإضافة وأن يستجيب برسالة خطأ. يتم حذف جميع الإدخالات العادية عند الدخول في وضع الطوارئ. عند الاتصال بوحدة تحكم مرة أخرى ، تظل الإدخالات تدفق الطوارئ. عندئذ يكون لدى وحدة التحكم خيار حذف جميع مداخل التدفق ، إذا رغبت في ذلك.

تقوم وحدة التحكم بتكوين المبدل وإدارته واستقبال الأحداث منه ، وإرسال الحزم إليه من خلال هذه الواجهة. باستخدام بروتوكول OpenFlow ، يمكن لوحدة التحكم إضافة أو تحديث أو حذف إدخال التدفق من جدول تدفق المبدل. يمكن أن يحدث ذلك بشكل تفاعلي (استجابة لوصول حزمة) أو بشكل استباقي. يمكن اعتبار بروتوكول OpenFlow أحد التطبيقات الممكنة لتفاعلات تبديل وحدة التحكم (الواجهة الجنوبية) ، حيث إنه يحدد الاتصال بين أجهزة التبديل ووحدة التحكم في الشبكة. للأمان ، يوفر الإصدار OpenFlow 1.3.x دعمًا اختياريًا لاتصال TLS المشفر وتبادل الشهادات بين المبدلات / المتحكمات.

يحدد بروتوكول OpenFlow ثلاثة أنواع من الرسائل ، ولكل منها أنواع فرعية متعددة:

- ❖ Controller-to-switch
- ❖ Symmetric

• Controller-to-switch:

تبدأ رسائل وحدة التحكم إلى المبدل بواسطة وحدة التحكم وتستخدم مباشرة لإدارة حالة المبدل أو فحصها. قد يتطلب أو لا يتطلب هذا النوع من الرسائل استجابة من المبدل ويتم تصنيفها في الأنواع الفرعية التالية:

- Features: عند إنشاء جلسة TLS ، ترسل وحدة التحكم رسالة طلب إلى المحول. يجب أن يرد المحول برسالة تحدد الميزات والإمكانيات التي يدعمها المحول.
- Configuration: وحدة التحكم قادرة على ضبط معلومات التكوين والاستعلام عنها في المحول. يستجيب رمز المبدل فقط لاستعلام من وحدة التحكم.
- Modify-State: يتم إرسال هذه الرسائل بواسطة وحدة التحكم لإدارة حالة المبدلات. يتم استخدامها لإضافة أو حذف أو تعديل إدخالات جدول التدفق أو لتعيين أولويات منفذ التبديل.
- Read-State: تجمع هذه الرسائل الإحصائيات من جداول تدفق المبدل والمنافذ وإدخالات التدفق الفردية.
- Send-Packet: يتم استخدام هذه بواسطة وحدة التحكم لإرسال الحزم من المنفذ المحدد على المحول.
- Barrier: يتم استخدام رسائل طلب بواسطة وحدة التحكم لضمان تلبية تبعيات الرسالة أو لتلقي إشعارات للعمليات المكتملة.

• Asynchronous messages:

يبدأ المبدل بإرسال الرسائل غير المتزامنة وتستخدم لإعلام وحدة التحكم بأحداث الشبكة والتغييرات. ترسل المبدلات رسائل غير متزامنة إلى وحدة التحكم للإشارة إلى وصول الحزمة أو تغيير حالة التبديل أو حدوث خطأ. مثال عنها: Packet-in, Flow-Removal, Port-Status, Error.

• Symmetric messages:

يتم بدء الرسائل المتماثلة إما عن طريق المفتاح أو وحدة التحكم ويتم إرسالها دون التماس. مثال عنها: Hello, Echo, Vendor.

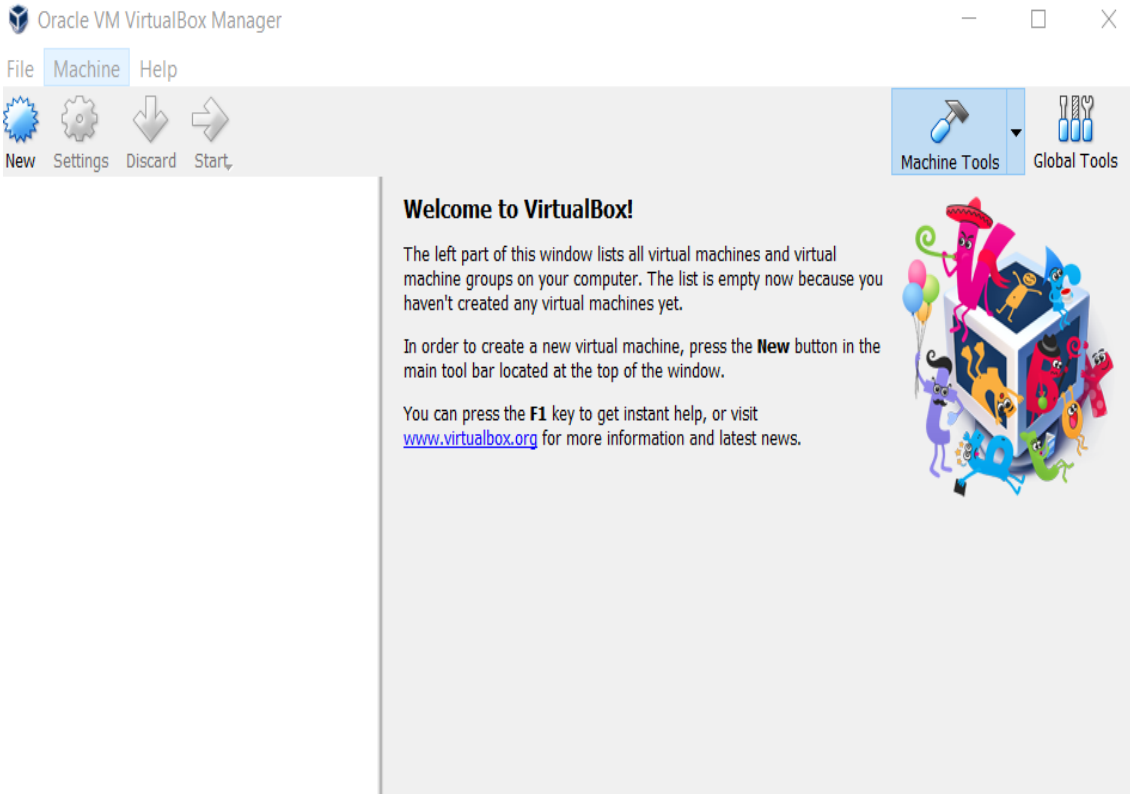
.....

الفصل الثاني

• Open Flow laboratory with Mininet

Mininet هي أداة برمجية تسمح بمحاكاة شبكة OpenFlow بالكامل على جهاز كمبيوتر واحد. لتشغيل العديد من المضيفين والمبدلات (على سبيل المثال 4096) على نواة نظام تشغيل واحدة. يمكنه إنشاء نواة أو مبدلات OpenFlow ، ووحدات تحكم للتحكم في المبدلات ، ومضيفين للتواصل عبر الشبكة التي تم إنشاؤها. Mininet يربط المبدلات والمضيفين باستخدام أزواج إيثرنت الافتراضية (veth). إنه يبسط عملية التطوير الأولية وتصحيح الأخطاء والاختبار والنشر. يمكن تطوير تطبيقات الشبكة الجديدة أولاً واختبارها على محاكاة شبكة النشر المتوقعة ثم نقله إلى البنية التحتية التشغيلية الفعلية.

نقوم في البداية بتثبيت Oracle VM Virtual Box وبعد تثبيته نقوم بتشغيله فنحصل على الواجهة:



الشكل 2-1 تثبيت Oracle VM Virtual Box


لإضافة بيئة افتراضية جديدة نختار new فنحصل على الواجهة التالية:
حيث نقوم بضبط الاسم ب mininet

← Create Virtual Machine

Name and operating system

Please choose a descriptive name for the new virtual machine and select the type of operating system you intend to install on it. The name you choose will be used throughout VirtualBox to identify this machine.

Name:

Type: 

Version:

الشكل 2-2 إضافة البيئة الجديدة

يتم بعدها اختيار next فنحصل على الواجهة التي من خلالها نقوم باختيار حجم الذاكر التي سيتم تخصيصها للبيئة الافتراضية

← Create Virtual Machine

Memory size

Select the amount of memory (RAM) in megabytes to be allocated to the virtual machine.

The recommended memory size is **1024 MB**.

MB

4 MB 4096 MB

الشكل 3 – 2 تحديد الذاكر المخصصة للنظام الجديد

بعد ذلك يجب تحديد البيئة الافتراضية أو النظام الوهمي المراد بناءه

? X

← Create Virtual Machine

Hard disk

If you wish you can add a virtual hard disk to the new machine. You can either create a new hard disk file or select one from the list or from another location using the folder icon.

If you need a more complex storage set-up you can skip this step and make the changes to the machine settings once the machine is created.

The recommended size of the hard disk is **10.00 GB**.


- ☐ Do not add a virtual hard disk
- ☐ Create a virtual hard disk now
- ☒ Use an existing virtual hard disk file

mininet-vm-i386.vmdk (Normal, 8.00 GB)



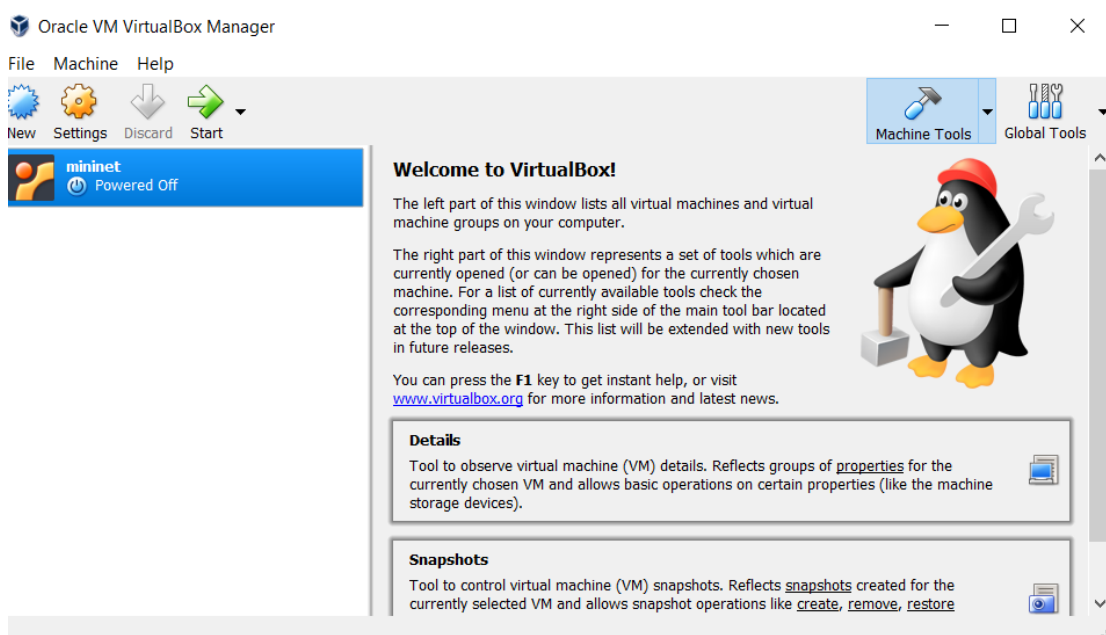
Create

Cancel

Name	Date modified	Type	Size
 mininet-vm-i386	3/21/2017 11:25 PM	Virtual Machine Di...	1,794

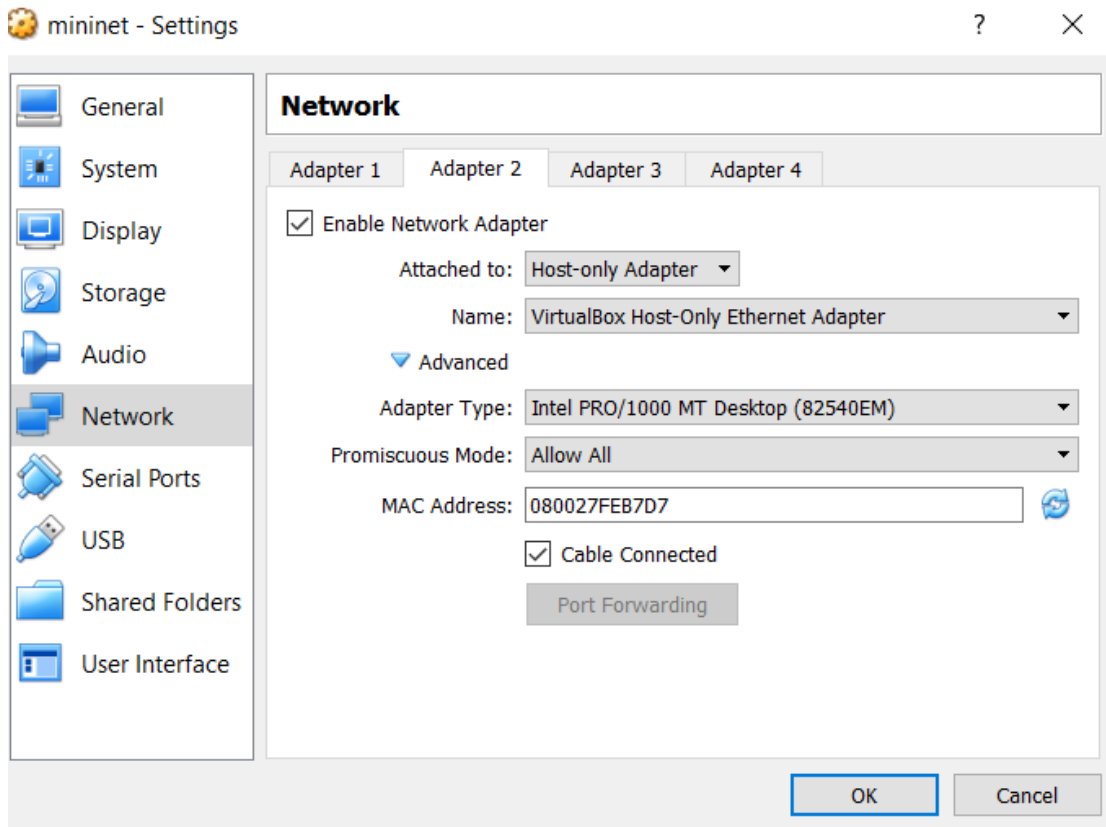
الشكل 2-3 تحديد النظام المراد تشغيله

نختار بعد ذلك create فنحصل على الواجهة:



الشكل 4 - 2 الواجهة النهائية

ثم نقوم بضبط بعض البارامترات ونختار Ok ليقفل النظام:



الشكل 5 - 2 ضبط البارامترات

فتظهر شاشة سوداء تدل على اقلاع النظام ثم ندخل اسم المستخدم وكلمة المرور: mininet
باستخدام 'Python API Mininet' ، من الممكن تحديد طبولوجيا مخصصة للتجارب.
يربط هذا المثال محولين مباشرة ، مع مضيف واحد متصل بكل محول:

```
from mininet.topo import Topo
class MyTopo( Topo ):
    def __init__( self ):
        "Create custom topo."
        # Initialize topology
        Topo.__init__( self )
        # Add hosts and switches
        leftHost = self.addHost( 'h1' )
        rightHost = self.addHost( 'h2' )
        leftSwitch = self.addSwitch( 's3' )
        rightSwitch = self.addSwitch( 's4' )
        # Add links
        self.addLink( leftHost, leftSwitch )
        self.addLink( leftSwitch, rightSwitch )
        self.addLink( rightSwitch, rightHost )
topos = { 'mytopo': ( lambda: MyTopo() ) }
```

يمكن تمرير نص Python هذا كمعامل سطر أوامر إلى Mininet. عندما يتم توفير ملف Mininet مخصص ، يمكنه إضافة طبولوجيا وأنواع تبديل واختبارات جديدة إلى سطر الأوامر. على سبيل المثال ، يمكن تنفيذ اختبار pingall باستخدام الهيكل المذكور مع استدعاء Mininet التالي:

```
$ sudo mn --custom ~/mininet/custom/1st code.py --topo mytopo
--test pingall
```

يتم ارسال رسالة OpenFlow كما هو موضح بالكود التالي:

```
from OpenFlow import ofp_packet_out
match = of.ofp_match()
match.in_port = 3
ofp_packet_out OpenFlow message
def send_packet (self, buffer_id, raw_data, out_port, in_port):
    """
    Sends a packet out of the specified switch port.
    If buffer_id is a valid buffer on the switch, use that.
    Otherwise, send the raw data in raw_data.
    The "in_port" is the port number that packet arrived on. Use
    OFPP_NONE if you're generating this packet.
    """
```

```

msg = of.ofp_packet_out()
msg.in_port = in_port
if buffer_id != -1 and buffer_id is not None:
    msg.buffer_id = buffer_id
else:
    if raw_data is None:
        return
    msg.data = raw_data
action = of.ofp_action_output(port = out_port)
msg.actions.append(action)
self.connection.send(msg)
ofp_flow_mod OpenFlow message

```

ثم ننفذ الأمر التالي:

```
$ sudo mn --custom ~/mininet/custom/2d code.py --topo mytopo
```

```
--test pingall
```

تم اقتراح مفهوم الشبكات القابلة للبرمجة كطريقة لتسهيل تطور الشبكة. على وجه الخصوص ، يعد SDN نموذجًا جديدًا للشبكات ، يتم فيه فصل أجهزة إعادة التوجيه (على سبيل المثال ، محركات إعادة توجيه الحزمة المتخصصة) عن قرارات التحكم (على سبيل المثال ، البروتوكولات وبرامج التحكم). يتيح ترحيل منطق التحكم ، الذي كان يتم دمجها بإحكام في أجهزة الشبكات (على سبيل المثال ، محولات Ethernet) إلى وحدات تحكم مركزية منطقية ويمكن الوصول إليها ، استخلاص البنية التحتية للشبكات الأساسية من وجهة نظر التطبيق.

واجهة miniedit :

تمكن هذه الواجهة من تسهيل طبولوجيا خاصة بالمستخدم ولكنها لا تولد جميع الخصائص التي يمكن الاستفادة منها ضمن mininet.

هذه الواجهة مكتوبة باستخدام لغة Python.

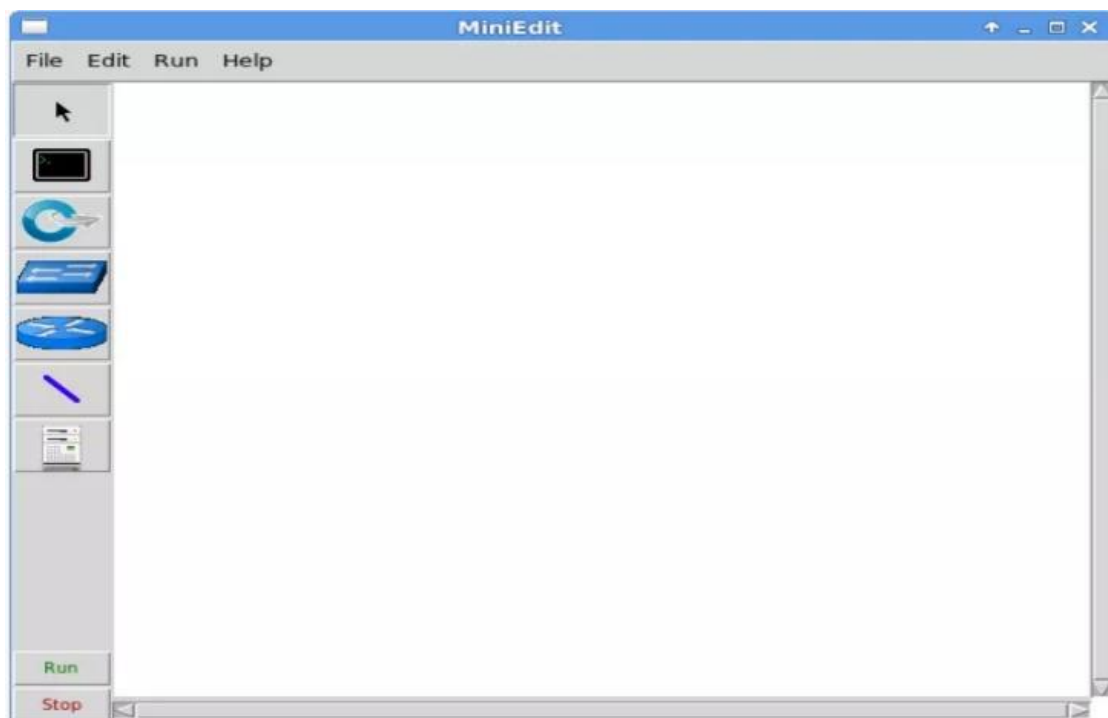
نقوم أولاً بإنشاء الاتصال وتشغيل mininet كما هو مبين في الصورة التالية

```
mininet@mininet-vm:~$ ls
install-mininet-vm.sh  loxigen  mininet  oflops  oftest  openflow  pox
mininet@mininet-vm:~$ cd mininet
mininet@mininet-vm:~/mininet$ ls
bin          custom  doc      LICENSE  mininet.egg-info  mnexec.1  setup.py
build        debian  examples Makefile  mn.1         mnexec.c  util
CONTRIBUTORS  dist    INSTALL  mininet  mnexec       README.md
mininet@mininet-vm:~/mininet$ cd examples
mininet@mininet-vm:~/mininet/examples$ ls
baresshd.py      cpu.py          multilink.py    README.md
bind.py           emptynet.py    multiping.py     scratchnet.py
clustercli.py     hwintf.py       multipoll.py     scratchnetuser.py
clusterdemo.py   __init__.py     multitest.py     simpleperf.py
cluster.py        intfoptions.py  mytest.py        sshd.py
clusterSanity.py  limit.py        natnet.py        test
consoles.py       linearbandwidth.py  nat.py          tree1024.py
controllers2.py   linuxrouter.py  numberedports.py  treeping64.py
controllers.py    miniedit.py     popenpoll.py     vlanhost.py
controlnet.py     mobility.py     popen.py
mininet@mininet-vm:~/mininet/examples$
```

نقوم بتشغيل واجهة miniedit باستخدام terminal وبما أنها مكتوبة باستخدام Python يعد أمر الفتح أمر خاص بهذه اللغة، يتم ذلك كما يلي:

```
sudo python miniedit.py
```

بعد تنفيذ هذا الأمر تظهر لنا الواجهة كما هو مبين في الصورة التالية:



سنقوم الان بمحاكاة شبكة شجرية باستخدام Openflow وواجهة miniedit وفق السيناريو التالي:

شبكة شجرية بعمق يساوي 2 وعدد ابناء أعظمي 2.

المتحكم لدينا هو الافتراض

عرض حزمة الوصلات:

Mbit 10

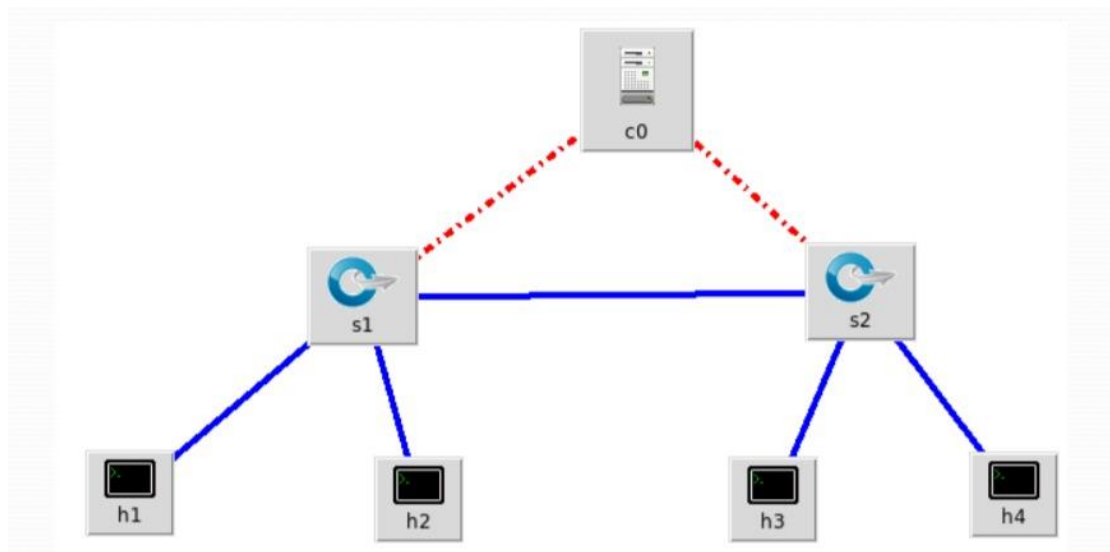
التأخير:

ms14

الخسارة:

% 2

نقوم بادراج الشبكة كما هو مبين في الشكل التالي:



هذه الشبكة مكونه من controller واحد و 2 switch وكل switch يتصل معه جهازين فقط وهو عدد الابناء .

C0 هو المتحكم يتم الوصل بينه وبين ال s0 باستخدام ليف ضوئي (fiber optic)

ونصل بين ال switch وال host المتصل معه باستخدام وصلات RG45

الآن ننفذ الأمر nano nname.py

nname.py هو اسم الملف الذي قمت بحفظ المشروع ضمنه

فيظهر الكود الخاص بالشبكة كما يلي:

```

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSKernelSwitch
from mininet.node import CPULimitedHost, Host, Node
from mininet.node import OVSKernelSwitch, UserSwitch
from mininet.node import IVSSwitch
from mininet.cli import CLI
from mininet.log import setLogLevel, info
from mininet.link import TCLink, Intf
from subprocess import call

```

```

def myNetwork():

    net = Mininet( topo=None,
                  build=False,
                  ipBase='10.0.0.0/8')

    info( '*** Adding controller\n' )
    c0=net.addController(name='c0',
                        controller=Controller,
                        protocol='tcp',
                        port=6633)

    info( '*** Add switches\n')
    s1 = net.addSwitch('s1', cls=OVSKernelSwitch)
    s2 = net.addSwitch('s2', cls=OVSKernelSwitch)

    info( '*** Add hosts\n')
    h1 = net.addHost('h1', cls=Host, ip='10.0.0.1', defaultRoute=None)
    h2 = net.addHost('h2', cls=Host, ip='10.0.0.2', defaultRoute=None)
    h3 = net.addHost('h3', cls=Host, ip='10.0.0.3', defaultRoute=None)
    h4 = net.addHost('h4', cls=Host, ip='10.0.0.4', defaultRoute=None)

```

```

info( '*** Add links\n')
h1s1 = {'bw':10,'delay':'14','loss':2}
net.addLink(h1, s1, cls=TCLink , **h1s1)
s1h2 = {'bw':10,'delay':'14','loss':2}
net.addLink(s1, h2, cls=TCLink , **s1h2)
h3s2 = {'bw':10,'delay':'14','loss':2}
net.addLink(h3, s2, cls=TCLink , **h3s2)
s2h4 = {'bw':10,'delay':'14','loss':2}
net.addLink(s2, h4, cls=TCLink , **s2h4)
s2s1 = {'bw':10,'delay':'14','loss':2}
net.addLink(s2, s1, cls=TCLink , **s2s1)

info( '*** Starting network\n')
net.build()
info( '*** Starting controllers\n')
for controller in net.controllers:
    controller.start()

info( '*** Starting switches\n')
net.get('s1').start([c0])
net.get('s2').start([c0])

info( '*** Post configure switches and hosts\n')

CLI(net)

```

```
if __name__ == '__main__':  
    setLogLevel('info')  
    myNetwork()
```

نلاحظ أن واجهة mimiedit تقوم بتسهيل العمل باستخدام شبكات SDN وتقلل الحاجة لاستخدام موجه الأوامر أو كتابة التعليمات البرمجية مما يثبت سهولة العمل باستخدام الشبكات المرفقة برمجيا حتى لمن ليس لديه خلفية برمجية.

نقوم الآن بعمل اختبارات عمل الشبكة:

نقوم بتشغيل الشبكة باستخدام الأمر:

sudo python nname.py

الأمر pingall يقوم باختبار الاتصال بين المضيفين

```
mininet> pingall  
*** Ping: testing ping reachability  
h1 -> h2 h3 h4  
h2 -> h1 h3 h4  
h3 -> h1 h2 h4  
h4 -> h1 h2 h3
```

الأمر iperf لتفحص عرض الحزمة بين أي مضيفين في الشبكة:

```
mininet> iperf h1 h4  
*** Iperf: testing TCP bandwidth between h1 and h4  
*** Results: ['5.66 Mbits/sec', '5.76 Mbits/sec']
```

مما سبق قمنا بتشكيل شبكة SDN باستخدام بروتوكول Openflow علما أنه تم ضبط العناوين على الشبكة باستخدام الشبكة:

172.172.1.0/16

وتم ضبط الحركة للعمل على البورت 3001

الشبكة السابقة مناسبة لتنفيذ والتطبيق في شركة صغيرة تضم بالحد الأقصى 10 موظفين حيث يمكن تطوير العمل المستقبلي وجعل كل مبدل يقوم بتخديم خمس مضيفين بدلا من اثنين.

المراجع:

1. Lara, A.; Kolasani, A.; Ramamurthy, B. Network Innovation Using OpenFlow: A Survey. IEEE Commun. Surv. Tutor. 2013, 16, 1–20.
2. Astuto, B.N.; Mendonça, M.; Nguyen, X.N.; Obraczka, K.; Turetti, T. A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks. IEEE Commun. Surv. Tutor. 2014, doi:10.1109/SURV.2014.012214.00180.
3. Jain, R.; Paul, S. Network Virtualization and Software Defined Networking for Cloud Computing: A Survey. IEEE Commun. Mag. 2013, 51, 24–31.
4. Open Networking Foundation. Available online: <https://www.opennetworking.org/> (accessed on 22 July 2013).
5. Doria, A.; Salim, J.H.; Haas, R.; Khosravi, H.; Wang, W.; Dong, L.; Gopal, R.; Halpern, J. Forwarding and Control Element Separation (ForCES) Protocol Specification. RFC 5810 (Proposed Standard), 2010. Available online: <https://datatracker.ietf.org/doc/rfc5810/> (accessed on 22 July 2013).
6. Yang, L.; Dantu, R.; Anderson, T.; Gopal, R. Forwarding and Control Element Separation (ForCES) Framework. RFC 3746 (Informational), 2004. Available online: <https://datatracker.ietf.org/doc/rfc3746/> (accessed on 22 July 2013).