ورقة بحثية مجلة مفتوحة الوصول

استخدام المحاكي Mininet في الشبكات المعرفة بالبرمجيات SDN

د. بشرى معلا*، أ.د. مثنى القبيلي**، م. محمد عبد الحميد***

*(كلية الهندسة ،جامعة المنارة ، البريد الإلكتروني: boushra.maala@manara.edu.sy)

**(كلية الهندسة ،جامعة المنارة ، البريد الإلكتروني: mothanna.alkubeily@manara.edu.sy)

***(كلية الهندسة الميكانيكية والكهربائية ،جامعة تشرين ، البريد الإلكتروني: mohammedabdlhamed589@gmail.com)

الملخص

تمثل الشبكات المعرفة بالبرمجيات العصر الحالي من الشبكات وذلك بسبب ميزتيها في فصل مستوى التحكم عن مستوى البيانات وإضافة قابلية البرمجة للشبكة من خلال البرمجيات. يستخدم المحاكي mininet في شبكات SDN لمحاكاة سلوك الشبكة والمتحكمات فيها ويمكن استخدامه لدراسة أداء الشبكة من ناحية التأخير والإنتاجية. سنقدم في هذه الورقة دراسة عملية تفصيلية لمحاكي mininet وكيفية تطبيق شبكات SDN باستخدام هذا المحاكي.

كلمات مفتاحية: محاكي mininet ، طوبولوجيا، متحكم، شبكات SDN.

I. مقدمة

يعد mininet برمجية مفتوحة المصدر [1] وتستخدم من أجل محاكاة الشبكات المعرفة بالبرمجيات Software Defined Networks (SDNs) والأجهزة التي تتواجد ضمنها مثل المتحكمات المعقدة والمبدلات وأجهزة المضيفين. يستخدم برنامج mininet من قبل محترفي الشبكات والمبتدئين من أجل الحصول على الخبرة للتعامل مع تكنولوجيا SDN المتطورة بشكل متسارع. تقوم شبكات SDN بفصل مستوى التحكم عن مستوى البيانات وتجعل أجهزة الشبكة مثل المبدلات والموجهات قابلة للبرمجة بشكل كامل ومن ثم فإنه يمكن التحكم بسلوك الشبكة بحسب متطلبات المستخدمين. يدعم محاكي mininet أنواع متحكمات ومبدلات افتراضية ويمكن أيضاً تثبيت متحكمات أو مبدلات محددة بدلاً من الافتراضية.

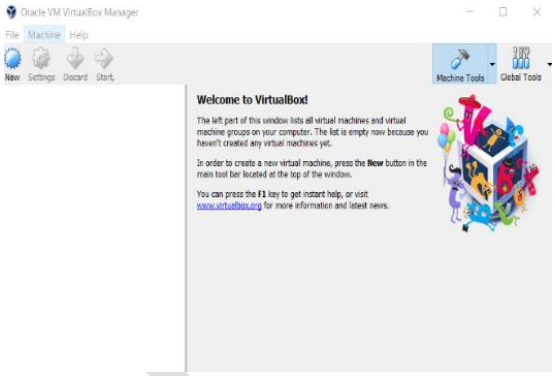
يدعم mininet البروتوكول OpenFlow والذي يؤمن واجهة ما بين طبقة التحكم وطبقة تمرير البيانات. يستخدم البروتوكول OpenFlow من أجل التحكم بتدفق الرزم من خلال الأوامر التي تكتب في المتحكم. يدعم أيضاً برنامج mininet مجموعة من الطوبولوجيات ويدعم أيضاً إمكانية إضافة طوبولوجيات خاصة بالمستخدم. تعد واجهة سطر الأوامر Command Line Interface (CLI) المزودة من قبل mininet مريحة جداً للتعامل معها ولكن تحتاج بعض التدريب.

قبل البدء بالتطبيق على البرنامج (لمستخدمي windows) يجب تثبيت الأداة putty.exe والأداة Xming.

II. آلية تثبيت البرنامج على نظام WINDOWS

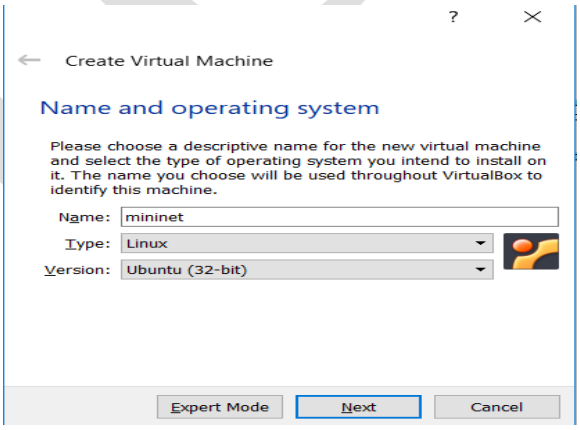
سنتبع الخطوات التالية لتنصيب mininet على نظام تشغيل windows:

1. من الموقع الرسمي [1] لـ mininet نقوم بتحميل نسخة mininet وهو مبني على نظام Ubuntu.
2. نقوم بتحميل oracle vm virtualbox من الموقع الرسمي [2] ومن ثم نقوم ببدء التنصيب.
3. بعد فتح virtualbox نحصل على الواجهة.



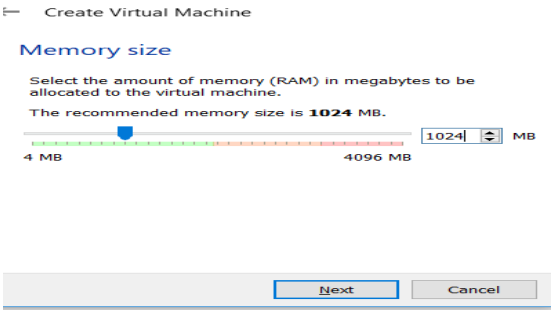
الشكل 1. واجهة برنامج VBox

نقوم باختيار new فنحصل على واجهة تضبط فيها الإعدادات كما يلي بحيث يحدد اسم النظام mininet ونوعه Linux والإصدار هو نسخة Ubuntu (32-bit).



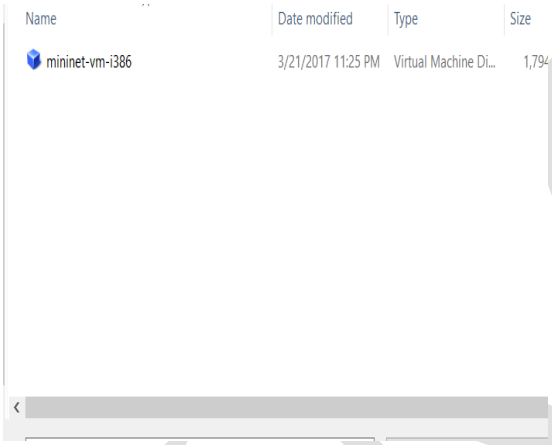
الشكل 2. تحديد الاسم ونوع نظام التشغيل

بعد الضغط على زر next نقوم باختيار حجم ذواكر RAM المطلوب للنظام الوهمي، وكلما قمنا بتخصيص حجم ذواكر أكبر يعمل النظام الوهمي بكفاءة أعلى ولكن يجب التخصيص بما يتوافق مع حجم ذواكر RAM للجهاز.



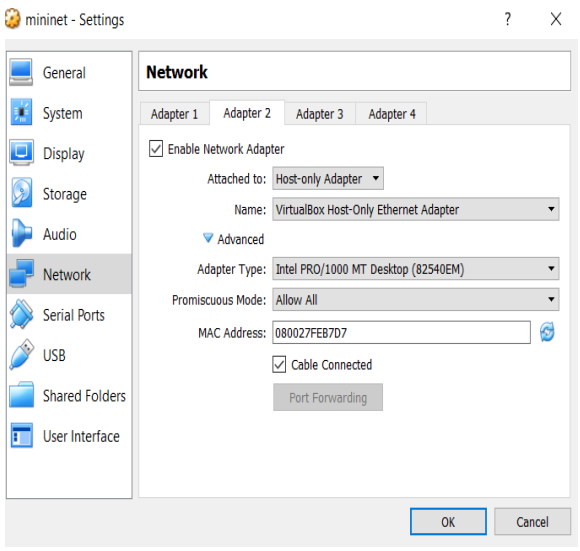
الشكل 3. تخصيص حجم ذاكرة للنظام الافتراضي

ثم بعد الضغط على next يجب أن نختار النظام الوهمي من مسار المجلدات الذي تم حفظه فيه.

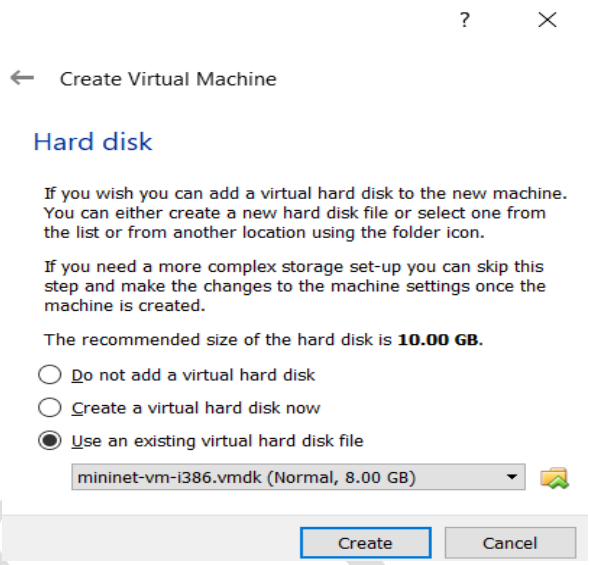


الشكل 4. اختيار نظام mininet للتثبيت

نضغط على mininet ونختار open فنحصل على الواجهة التالية بحيث نقوم باختيار الخيار الثالث الذي يقوم باستخدام القرص الصلب الحالي للتثبيت.



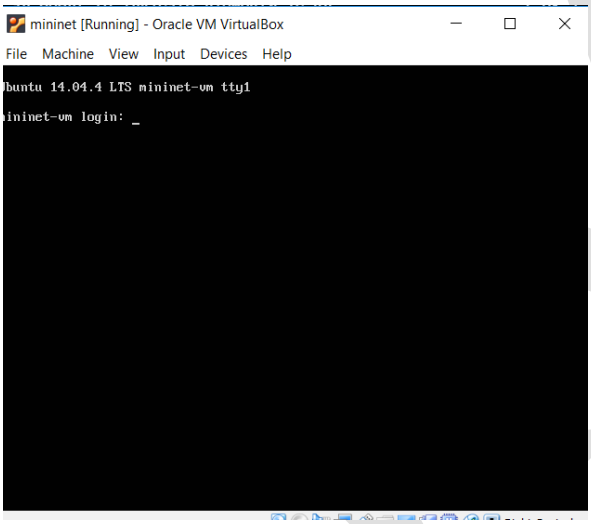
الشكل 7. ضبط إعدادات الشبكة للنظام الوهمي



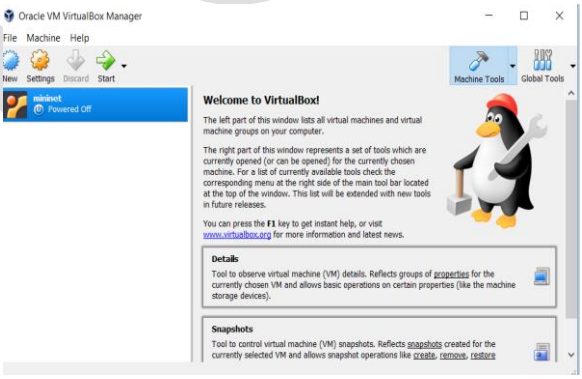
الشكل 5. إدراج النظام ضمن برنامج VBox

بعدها نضغط على start وننتظر بدء التشغيل بعد ذلك نحصل على الواجهة التالية:

نختار بعد ذلك create فنحصل على الوجهة التالية ويصبح لدينا محاكي mininet مثبت ضمن VBox.



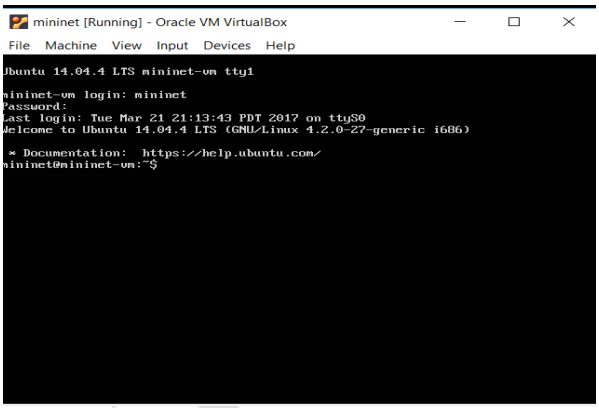
الشكل 8. واجهة تسجيل الدخول للنظام



الشكل 6. ارتباط النظام مع برنامج VBox

اسم المستخدم هو mininet وكلمة المرور هي mininet ومن ثم نضغط على زر Enter.

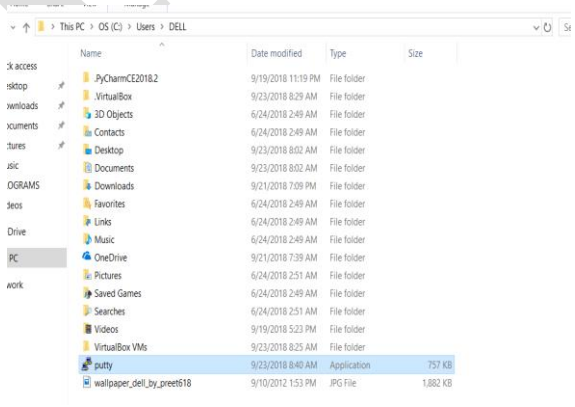
يجب ضبط إعدادات الشبكة للنظام الوهمي لكي يعمل بشكل سليم. بعدها نقوم بالضغط على setting وضمنها نختار network ومن ثم adapter2 ومن ثم enable network adapter ومن ثم host only adapter ومن خلال advanced من خلال السماحيات نختار allow all ثم ok والغرض من هذه الخطوة هو إتاحة الاتصال عن بعد مع نظام mininet والتحكم به



الشكل 9. واجهة النظام بعد نجاح الدخول

إن هذه الواجهة ليست برنامج mininet وإنما هي واجهة Ubuntu ويجب الدخول منها إلى mininet. تعد هذه الواجهة معقدة في التعامل لذلك يتم الاتصال عادة مع نظام mininet خارجياً والتحكم به من خلال النظام الأساسي windows وللقيام بذلك نقوم أيضاً بتحميل putty.exe من موقعه [3] إضافة لـ xming أيضاً [4].

بعد تحميل putty.exe نفتح واجهة cmd ونقوم بنسخ putty.exe إلى المكان الذي يشير له المسار.

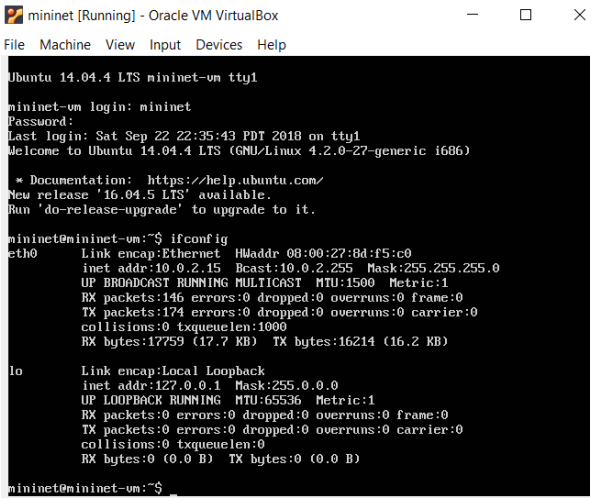


الشكل 10. نقل الاداة putty إلى مسار موجه الأوامر

بعد ذلك نقوم بتشغيل xming والذي سيعمل في خلفية الحاسب وبالعودة إلى mininet نكتب التعليمة

Ifconfig

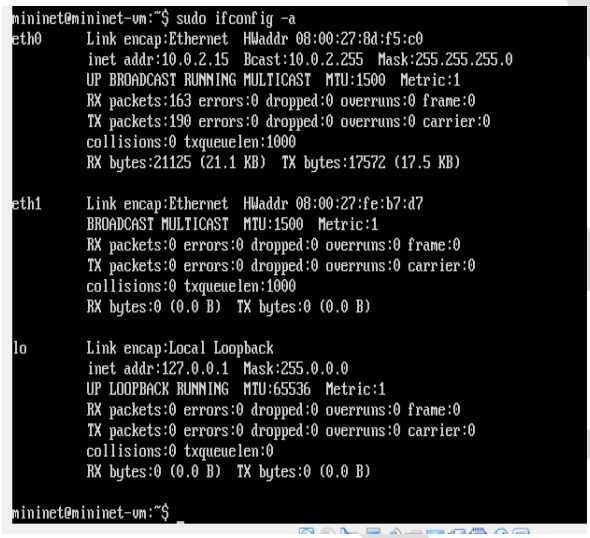
لنحصل على الواجهات الموجودة



الشكل 11. استعراض الواجهات ضمن النظام

ويوجد واجهتين الواجهة المحلية lo وواجهة eth0 والتي تمثل adapter 1 للنظام الوهمي. نستخدم التعليمة التالية والتي تظهر جميع الواجهات

sudo ifconfig –a



الشكل 12. عرض جميع الواجهات المتاحة

فتظهر لدينا الواجهة eth1 وهي الواجهة التي قمنا بتفعيلها كـ host only apater.

وسوف نستخدم هذه الواجهة eth1 لذلك يجب تفعيلها من خلال الأمر

sudo dhclient eth1

ويمكن استخدام clear لحذف محتوى الواجهة.

بالعودة إلى شاشة cmd لنظام windows نكتب

Putty.exe –X mininet@(eth1–ip)

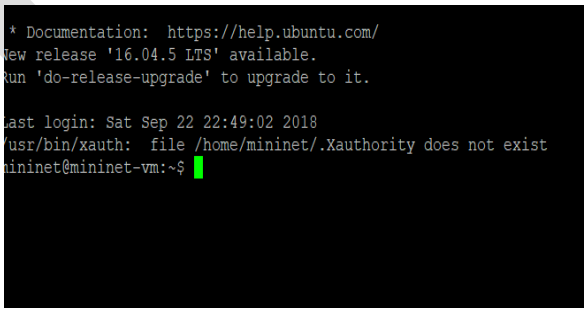
مع مراعاة أن X حرف كبير وتمثل Xming بينما العنوان –eth1 ip نحصل عليه من واجهة mininet بعد كتابة التعليمة

Ifconfig –a

ونأخذه من eth1 وتصبح التعليمة ضمن cmd وفق الشكل

الشكل 13. الاتصال من النظام مع النظام الوهمي

وبعد الضغط على enter تظهر واجهة وكلمة المرور هي mininet



الشكل 14. واجهة النظام من الاداة putty.

بعد تشغيل البرنامج نستخدم الأمر

sudo mn

الذي يقوم بإنشاء شبكة SDN بمتحكم وحيد مع مبدل ومضيفين ويقوم بإنشاء وصلات ما بين المبدلات والمضيفين ضمن الشبكة ويفتح واجهة CLI من أجل بدء تنفيذ الأوامر

```
mininet@mininet-vm:~$ sudo mn --test
Usage: mn [options]
(type mn -h for details)

mn: error: --test option requires an argument
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

الشكل 15. تشغيل شبكة بسيطة.

في حال الحصول على الشكل السابق يكون تم تثبيت المحاكي بشكل صحيح.

III. العمل على برنامج MININET

A. الأوامر المستخدمة في محاكي mininet

يوجد مجموعة من الأوامر المستخدمة في واجهة CLI مثلا:

الأمر **help** يستعرض جميع التعليمات التي من الممكن استخدامها ضمن هذه الواجهة

```
mininet> help

Documented commands (type help <topic>):
========================================
EOF    gterm  iperfudp  nodes       pingpair      py      switch
dpctl  help   link      noecho      pingpairfull  quit    time
dump   intfs  links     pingall     ports         sh      x
exit   iperf  net       pingallfull px            source  xterm

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2
```

الشكل 16. نتيجة تنفيذ الأمر help.

الأمر **links** يظهر جميع الوصلات ضمن الشبكة

```
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
```

الشكل 17. نتيجة تنفيذ الأمر links.

الأمر **nodes** يظهر جميع العقد ضمن الشبكة

```
mininet> nodes
available nodes are:
c0 h1 h2 s1
```

الشكل 18. نتيجة تنفيذ الأمر nodes.

الأمر **pingall** من أجل إجراء عملية ping بين جميع المستخدمين ضمن الشبكة

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

الشكل 19 . نتيجة تنفيذ الأمر pingall.

الأمر **net** من أجل إظهار جميع خصائص الشبكة

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
```

الشكل 20 . نتيجة تنفيذ الأمر net.

الأمر **iperf** من أجل إظهار عرض الحزمة بين المضيفين في الشبكة حيث يقوم باختبار عرض الحزمة وفق بروتوكول طبقة النقل TCP.

```
mininet> iperf
*** Iperf: testing TCP bandwidth between h1 and h2
.*** Results: ['11.9 Gbits/sec', '11.9 Gbits/sec']
```

الشكل 21 . نتيجة تنفيذ الأمر iperf.

الأمر **py h.IP()** يستخدم من أجل معرفة عنوان IP الخاص بعقدة معينة.

الأمر **py h.MAC()** يستخدم من أجل معرفة عنوان MAC الخاص بعقدة معينة.

حيث أن IP() و MAC() هما طريقتين مكتوبتين بلغة Python ضمن المحاكي لذلك تستدعى من خلال الأمر py.

الأمر **n ifconfig** يظهر تفاصيل العقدة n (حيث n قد تكون مضيف او مبدل).

```
mininet> py h1.IP()
10.0.0.1
mininet> py h1.MAC()
56:bc:2a:6d:1b:b6
mininet> h1 ifconfig
h1-eth0   Link encap:Ethernet  HWaddr 56:bc:2a:6d:1b:b6
          inet addr:10.0.0.1  Bcast:10.255.255.255  Mask:255.0.0.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:100110 errors:0 dropped:0 overruns:0 frame:0
          TX packets:155490 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:6607280 (6.6 MB)  TX bytes:7466686340 (7.4 GB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

الشكل 22. نتيجة تنفيذ الأوامر الخاصة بالعناوين.

الأمر link يستخدم لتغيير حالة وصلة ما (إما up أو down) مثلا الأمر link s1 h1 down يوقف الوصلة ما بين المبدل والمضيف h1 بينماالأمر link s1 h1 up يفعل الوصلة ما بين المبدل والمضيف h1.

B. الطوبولوجيات المستخدمة في محاكي mininet

تستخدم الخاصية **topo**-- أثناء إنشاء الشبكة من أجل تحديد طبولوجيا الشبكة ويوجد عدة خيارات:

1) **minimal** وهي الحالة الافتراضية حيث يتم إنشاء مبدل ومضيفين متصلين معه.

```
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

الشكل 23 . طوبولوجيا نوع minimal.

2) **single,x** طوبولوجيا تحوي على مبدل وحيد و x مضيف متصل معه.

```
mininet@mininet-vm:~$ sudo mn --topo=single,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1) (h4, s1)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

الشكل 24. طوبولوجيا نوع star.

3) **linear,x** ينشأ x مبدل متصلة مع بعضها خطياً وكل مبدل يحوي مضيف واحد متصل معه.

```
mininet@mininet-vm:~$ sudo mn --topo=linear,4
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3 s4
*** Adding links:
(h1, s1) (h2, s2) (h3, s3) (h4, s4) (s2, s1) (s3, s2) (s4, s3)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 4 switches
s1 s2 s3 s4 ...
*** Starting CLI:
```

الشكل 25. طوبولوجيا نوع linear.

4) **tree,x,y** ينشأ شبكة شجرية بعمق x وعدد أبناء y في كل مستوى.

```
mininet@mininet-vm:~$ sudo mn --topo=tree,2,2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4
*** Adding switches:
s1 s2 s3
*** Adding links:
(s1, s2) (s1, s3) (s2, h1) (s2, h2) (s3, h3) (s3, h4)
*** Configuring hosts
h1 h2 h3 h4
*** Starting controller
c0
*** Starting 3 switches
s1 s2 s3 ...
*** Starting CLI:
```

الشكل 26. طوبولوجيا نوع tree.

C. الخصائص المستخدمة في محاكي mininet

عند استخدام الأمر sudo mn لإنشاء شبكة SDN فإنه إضافة لتحديد الطوبولوجيات فإنه يمكن تحديد مجموعة من الخصائص الإضافية. تستخدم **الخاصية mac**-- من أجل إنشاء تخطيط بين العناوين الفيزيائية MAC والعناوين المنطقية IP وفق رقم المضيف أي مثلا المضيف h1 يملك عنوان IP هو 10.0.0.1 ويكون عنوانه الفيزيائي 00:00:00:00:00:01. يبين الشكل كيف تكون العناوين دون استخدام الخاصية.

```
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> py h1.MAC()
96:d1:25:c1:0d:f8
mininet> py h1.IP()
10.0.0.1
```

الشكل 27. عناوين الشبكة دون الخاصية mac.

بينما يبين الشكل كيف تصبح العناوين مع الخاصية

```
mininet@mininet-vm:~$ sudo mn --mac
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> py h1.MAC()
00:00:00:00:00:01
mininet> py h1.IP()
10.0.0.1
```

الشكل 28. عناوين الشبكة مع الخاصية mac.

تستخدم **الخاصية link**-- لتحديد نوع الوصلات المستخدمة وخصائص هذه الوصلات ضمن الشبكة ويبين الشكل 26 مثال على إنشاء شبكة مع استخدام الخاصية.

حيث ضمن الخاصية يحدد خصائص الوصلات حيث tc تعني أن الوصلات متناظرة بينما bw تحدد عرض الحزمة للوصلات ويكون الرقم المعطى بواحدة Mbit وتحدد delay التأخير على هذه الوصلات.

```
mininet@mininet-vm:~$ sudo mn --mac --link=tc,bw=10,delay=10ms
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(10.00Mbit 10ms delay) (10.00Mbit 10ms delay) (h1, s1) (10.00Mbit 10ms delay)
(10.00Mbit 10ms delay) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...(10.00Mbit 10ms delay) (10.00Mbit 10ms delay)
*** Starting CLI:
mininet>
```

الشكل 29. إنشاء شبكة SDN مع مواصفات للوصلات.

تستخدم **الخاصية test**-- من أجل إجراء اختبار على الشبكة عند تشغيلها مباشرة أي تشغل الشبكة بالخصائص المحددة ومن ثم تختبر وفق الحالة المحددة بالخاصية ويبين الشكل مثالاً على استخدامها.

```
mininet@mininet-vm:~$ sudo mn --mac --topo=single,3 --test=pingall
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Waiting for switches to connect
s1
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
*** Stopping 1 controllers
c0
*** Stopping 3 links
...
*** Stopping 1 switches
s1
*** Stopping 3 hosts
h1 h2 h3
*** Done
completed in 1.267 seconds
```

الشكل 30. تشغيل شبكة وإجراء اختبار باستخدام الخاصية test.

حيث نلاحظ كيف تم تشغيل الشبكة بطوبولوجيا ذات مبدل وحيد وعدد مضيفين 3 ومن ثم بسبب الخاصية test تم تنفيذ الأمر pingall أي اختبار الاتصال بين جميع أجهزة الشبكة وبعدها تم إيقاف الشبكة.

تستخدم الخاصية **--controller** من أجل تشغيل الشبكة وربطها مع متحكم خارجي وذلك يتطلب وجود متحكم في حالة عمل ويملك منفذ اتصال كي تتمكن الشبكة من الاتصال معه وتصبح مدارة من قبل المتحكم ويبين الشكل مثالاً على الخاصية.

```
mininet@mininet-vm:~$ sudo mn --controller=remote
*** Creating network
*** Adding controller
Unable to contact the remote controller at 127.0.0.1:6653
Unable to contact the remote controller at 127.0.0.1:6633
Setting remote controller to 127.0.0.1:6653
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

الشكل 31. ربط الشبكة مع متحكم خارجي

حيث نلاحظ أنه عند تحديد قيمة الخاصية بـ remote فإن الشبكة تحاول الاتصال مع متحكم خارجي وتظهر الرسالة أثناء التشغيل أن الشبكة غير قادرة على الاتصال لعدم وجود متحكم متاح.

يوجد مجموعة من الخصائص الإضافية والتي من الممكن استخدامها أثناء تشغيل الشبكة وهي مبينة بالجدول:

الخاصية	الوصف
--arp	من أجل إعداد جداول ARP ضمن الشبكة
--switch	لتحديد نوع المبدلات المستخدمة ضمن الشبكة
--host	لتحديد نوع أجهزة المضيفين ضمن الشبكة
--custom	لاستدعاء كود معرف مسبقاً لطوبولوجيا مبينة من قبل المستخدم

الجدول 1: الخصائص المستخدمة في إنشاء شبكة

IV. العمل على برنامج MINIEDIT:

تمكن هذه الواجهة من تشكيل طوبولوجيات خاصة بالمستخدم ولكنها لا تولد جميع الخصائص التي يمكن الاستفادة منها ضمن mininet. بعد تشغيل البرنامج يمكن الوصول إلى الأداة minedit وفق التسلسل المبين في الشكل:

حيث يستخدم الأمر ls: لاستعراض الملفات بينما يستخدم الأمر cd للدخول إلى ملف محدد.

```
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$
```

الشكل 32. استعراض الملفات ضمن النظام

حيث تتواجد الأداة ضمن المسار mininet/examples. حيث نلاحظ ضمن المسار السابق يوجد العديد من الملفات بامتداد py وهي ملفات مكتوبة بلغة Python لأمثلة عن شبكات وتحوي مجموعة من التوابع التي تساعد في فهم آلية برمجة الشبكة.

لتشغيل أي من هذه الملفات يستخدم الأمر:

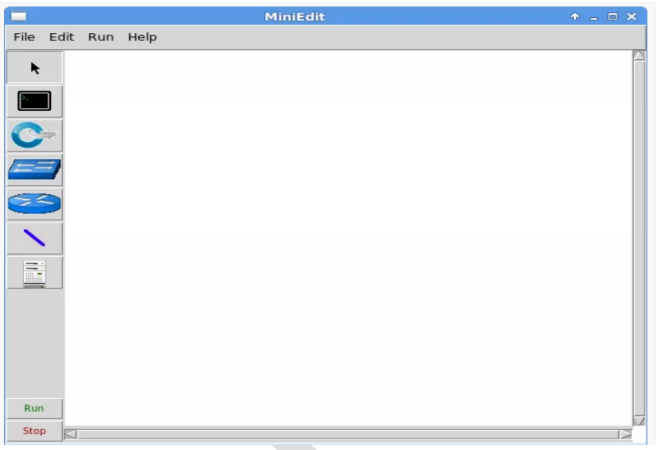
sudo python [file-name.py]

حيث تمثل file-name اسم الملف المراد تشغيله.

لتشغيل الأداة miniedit نستخدم الأمر:

sudo python miniedit.py

ولكن ذلك يفترض وجود برنامج Xming يعمل فنحصل على:



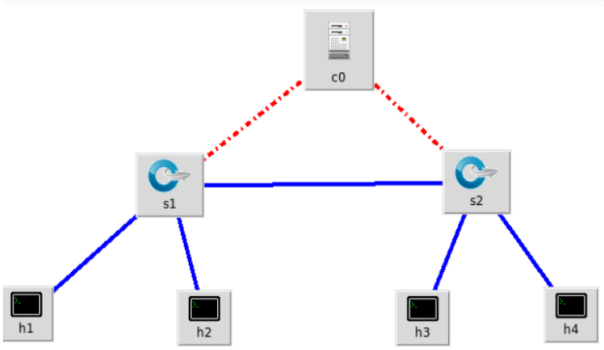
الشكل 33. واجهة الأداة miniedit.

ويبين الجدول التالي دلالات الرموز ضمن الواجهة

من أجل إضافة متحكم إلى الشبكة	
لإضافة وصلات بين عناصر الشبكة	
من أجل إضافة موجه شبكة تقليدي	
من أجل مبدل شبكة تقليدي	
من أجل إضافة مبدل شبكة SDN	
من أجل إضافة مضيفين للشبكة	

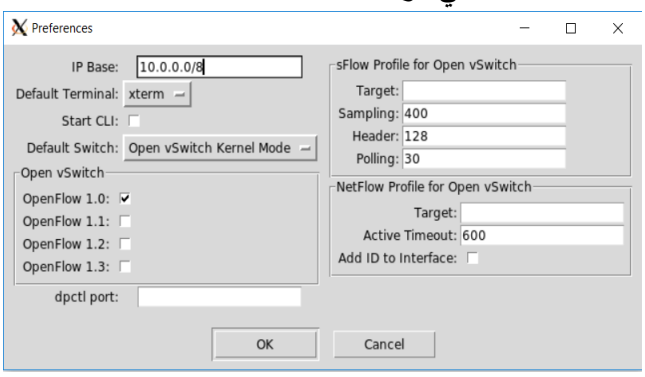
الجدول 1: انواع الرموز ضمن miniedit

بفرض نريد إنشاء الشبكة التالية بحيث يكون المتحكم افتراضي مع تشغيل موجه الأوامر CLI وبحيث تكون الوصلات بعرض حزمة 10Mbit وتأخير 14ms وخسارة 2%



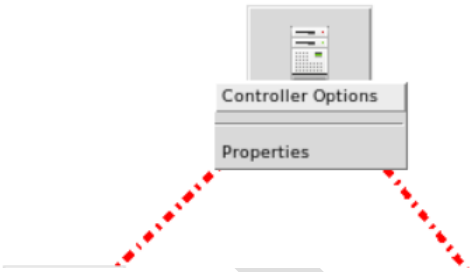
الشكل 34. تطبيق على miniedit.

بعد إنشاء الشبكة من الخيار edit نختار preferences لضبط إعدادات الشبكة كما في الواجهة التالية



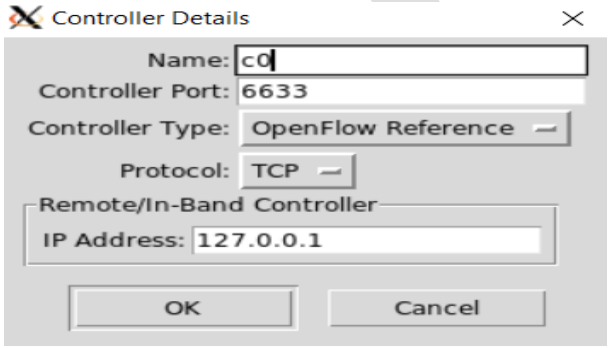
الشكل 35. ضبط إعدادت الشبكة.

حيث يمكن تحديد مجال عنونة IP الخاص بالشبكة والـ terminal التي ستستخدم إضافة لإصدار البروتوكول OF والمبدل الافتراضي وبارمترات أخرى يمكن ضبطها. نقوم بتفعيل CLI ثم نضغط ok لضبط خصائص المتحكم ضمن الشبكة نضغط عليه بالزر اليمين كما في الشكل ونختار properties



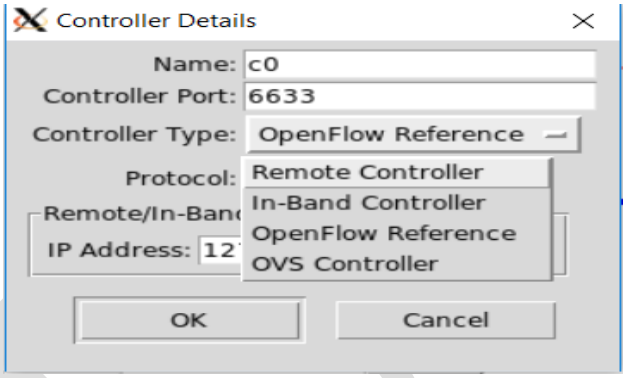
الشكل 36. كيفية الوصول لخصائص المتحكم.

حيث نحصل على الواجهة التالية



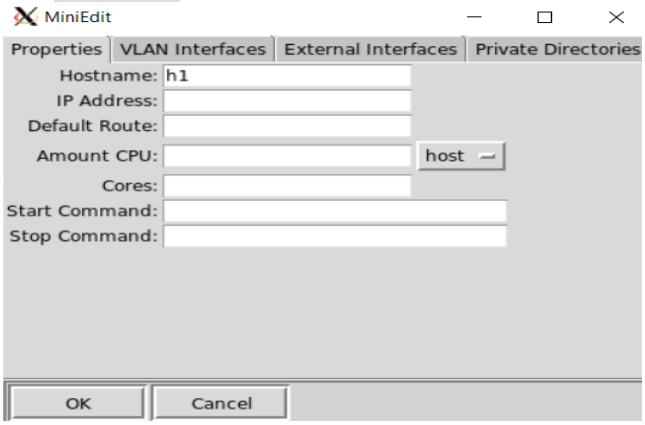
الشكل 37. ضبط خصائص المتحكم.

حيث من خلال هذه الواجهة يتم تحديد اسم المتحكم والمنفذ الذي يتم من خلاله الاتصال مع هذا المتحكم بالاضافة الى بروتوكول الاتصال و IP في حال اختيار متحكم remote ويوجد عدة أنماط للمتحكمات يمكن الاختيار بينها:



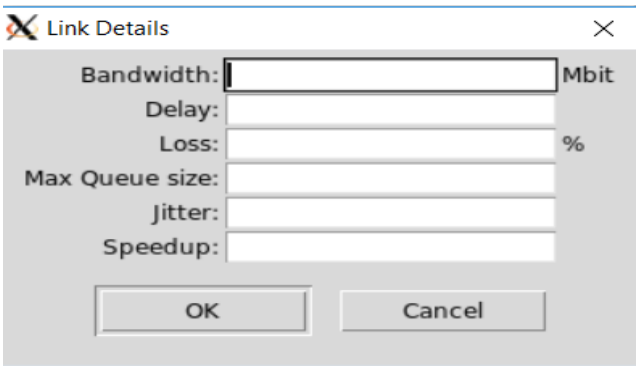
الشكل 38. أنواع المتحكمات المتاحة

بعد الانتهاء من ضبط الخصائص الخاصة بالمتحكم يمكن الضغط على ok. من أجل ضبط خصائص المضيفين بالزر الأيمن على عقدة المضيف ومن ثم الخيار Properties حيث يوجد مجموعة واسعة من الخيارات والتي يمكن التحكم بها.



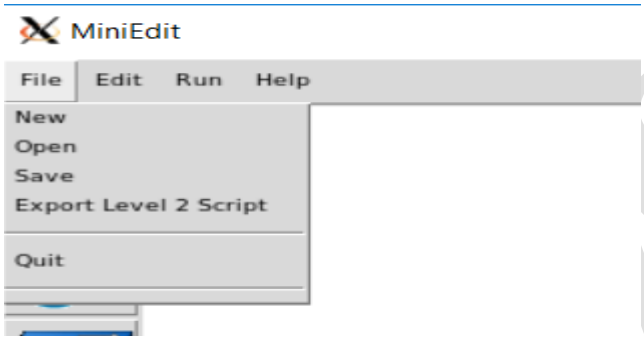
الشكل 39. ضبط المضيفين ضمن الشبكة.

يمكن أيضاً ضبط خصائص الوصلات من خلال الضغط بالزر الأيمن على أي وصلة ومن ثم Properties فنحصل على الواجهة



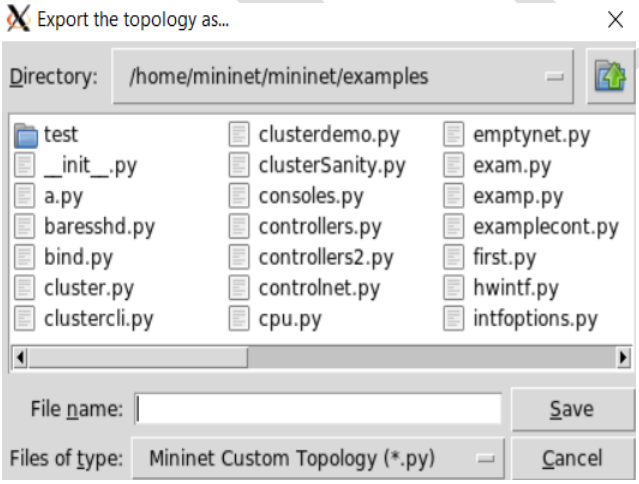
الشكل 40. ضبط خصائص الوصلات ضمن الشبكة.

بعد الانتهاء من ضبط الشبكة وخصائصها يمكن الحفظ من خلال الخيار File حيث يظهر خيارين



الشكل 41. حفظ الملف.

في حال اختيار Save سوف يتم الحفظ بامتداد mn أي يتم حفظ الطوبولوجيا التي تم اعدادها أما في حال اختيار export level2 scripts يتم توليد كود Python يصف هذه الشبكة.



الشكل 42. توليد كود برمجي يحاكي الشبكة.

حيث نقوم بإدراج الاسم المراد حفظ الشبكة به وليكن basic.py.

الآن بالعودة الى واجهة mininet الأساسية في حال استعراض الملفات ضمن هذا المسار فإننا سنحصل على basic.py بين الخيارات الموجودة.

```
mininet@mininet-vm:~/mininet/examples$ ls
a.py            example1.mn       miniedit.py       README.md         test1111.py
baresshd.py     example1.py       mobility.py       rectangle.py      test1.mn
basic.py        examplecont.py    mohammed.py       resukt            test1.py
bind.py         examp.py          multilink.py      result            test1.py.save
clustercli.py   exam.py           multiping.py      router1.py        tree1024.py
clusterdemo.py  first1.mn         multipoll.py      router.py         treeping64.py
cluster.py      first.py          multitest.py      scenario.py       try.py
clusterSanity.py hwintf.py        natnet.py         scratchnet.py     tsmem.py
consoles.py     __init__.py       nat.py            scratchnetuser.py vlanex.py
controllers2.py intfoptions.py    networkscript.py  second.py         vlanhost.py
controllers.py  limit.py          new_result        shaza.py
controlnet.py   linearbandwidth.py numberedports.py simpleperf.py
cpu.py          linuxrouter.py    popenpoll.py      sshd.py
emptynet.py     maalomatieh.py    popen.py          test
```

الشكل 43. استعراض الملف ضمن المسار examples.

لاستعراض محتوى الملف يمكن استخدام الأداة nano من خلال الأمر nano [file-name.py] حيث file-name هي basic (حسب المثال) فنحصل على

```
mininet@mininet-vm: ~/mininet/examples
  GNU nano 2.2.6                File: basic.py

#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController, OVSController
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' )
                [ Read 60 lines (Warning: No write permission) ]
^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text^T To Spell
```

الشكل 44. استعراض الكود المكتوب بلغة Python.

حيث نحصل على كود Python الذي تم توليده لهذه الشبكة.

وللخروج من خلال ctrl + x.

لتشغيل الشبكة من خلال الأمر

sudo python basic.py

```
mininet@mininet-vm:~/mininet/examples$ sudo python basic.py
*** Adding controller
*** Add switches
*** Add hosts
*** Add links
*** Starting network
*** Configuring hosts
h1 h2 h3 h4
*** Starting controllers
*** Starting switches
*** Post configure switches and hosts
*** Starting CLI:
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3 h4
h2 -> h1 h3 h4
h3 -> h1 h2 h4
h4 -> h1 h2 h3
*** Results: 0% dropped (12/12 received)
mininet>
```

الشكل 45. تشغيل الشبكة.

حيث نلاحظ كيف أن الشبكة تعمل بشكل سليم من خلال الأمر pingall.

V. حقن قواعد التدفق في المبدلات

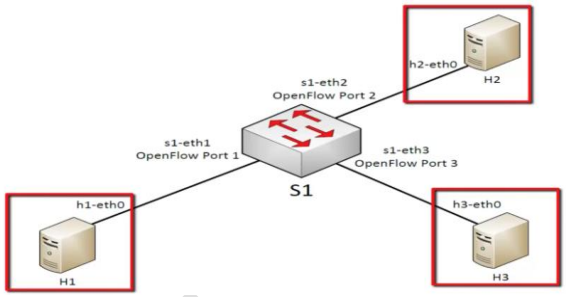
تستخدم الأداة ovs-ofctl من أجل حقن قواعد التدفق في المبدلات لكي تتمكن العقد من الاتصال مع بعضها البعض. يمكن استخدامها للقيام بالعمليات التالية:

1) عرض جداول تدفق المبدل.
2) إضافة مدخل (قواعد) جديدة لجدول التدفق.
3) حذف مدخل محدد من جدول التدفق أو حذفها جميعاً.
4) مراقبة الحركية على واجهات المبدل.

يمكن إضافة قواعد التدفق اعتماداً على 4 مستويات (طبقات) من مستويات الشبكة وهي:

1) الطبقة الفيزيائية: حيث تعتمد قاعدة التدفق فيها على رقم المنفذ الفيزيائي للمبدل.
2) طبقة وصلة المعطيات: تعتمد قاعدة التدفق فيها على عنوان MAC.
3) طبقة الشبكة: تعتمد قاعدة التدفق فيها على عنوان IP.
4) طبقة النقل: تعتمد قاعدة التدفق فيها على رقم منفذ طبقة النقل (منافذ TCP و UDP).

سنقوم بدراسة الشبكة ذات الشكل:



الشكل 46. الشبكة المدروسة.

للحصول عليها نستخدم الأمر المبين في الشكل:

```
mininet@mininet-vm:~$ sudo mn --topo=single,3 --controller=none --mac
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1) (h3, s1)
*** Configuring hosts
h1 h2 h3
*** Starting controller

*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

الشكل 47. تشغيل الشبكة.

حيث بما أننا نضيف قواعد التدفق يدوياً فيجب إلغاء وجود المتحكم عن طريق وضع قيمة none أو remote في تعليمة إنشاء الشبكة لذلك تم استخدام --controller=none.

يمكن عرض تخطيط واجهات المبدل مع أرقام منافذ المبدل من خلال الأمر المبين في الشكل:

sh ovs-ofctl show s1

```
mininet> sh ovs-ofctl show s1
OFPT_FEATURES_REPLY (xid=0x2): dpid:0000000000000001
n_tables:254, n_buffers:256
capabilities: FLOW_STATS TABLE_STATS PORT_STATS QUEUE_STATS ARP_MATCH_IP
actions: OUTPUT SET_VLAN_VID SET_VLAN_PCP STRIP_VLAN SET_DL_SRC SET_DL_DST SET_N
W_SRC SET_NW_DST SET_NW_TOS SET_TP_SRC SET_TP_DST ENQUEUE
 1(s1-eth1): addr:52:60:fe:e0:13:7e
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 2(s1-eth2): addr:72:49:3a:2d:49:17
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 3(s1-eth3): addr:f2:fe:a7:d0:2d:3b
     config:     0
     state:      0
     current:    10GB-FD COPPER
     speed: 10000 Mbps now, 0 Mbps max
 LOCAL(s1): addr:f2:e4:cb:b7:60:06
     config:     PORT_DOWN
     state:      LINK_DOWN
     speed: 0 Mbps now, 0 Mbps max
OFPT_GET_CONFIG_REPLY (xid=0x4): frags=normal miss_send_len=0
```

الشكل 48. عرض إعدادات المبدل s1.

بما أنه لا يوجد أي قواعد تدفق ضمن المبدل فإنه في حال قمنا باختبار الاتصال بين أجهزة الشبكة نلاحظ أنه لا يوجد أي اتصال كما في الشكل:

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
```

الشكل 49. عدم نجاح الاتصال.

في حال أردنا المبدل ضمن الشبكة أن يعمل كمبدل شبكة تقليدية نستخدم الخاصية normal

sh ovs-ofctl add-flow s1 action=normal

حيث باستخدام الأمر السابق يتم حقن قاعدة تدفق ضمن المبدل s1 تخبره أن آلية عمله هي الحالة الطبيعية للمبدل normal وبالتالي يعمل كمبدل عادي.

وبعدها نختبر الاتصال ونلاحظ نجاحه كما هو مبين في الشكل:

```
mininet> sh ovs-ofctl add-flow s1 action=normal
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

الشكل 50. تشغيل المبدل كمبدل شبكة تقليدي.

يمكن عرض جدول التدفق الخاص بالمبدل s1 وفق الأمر المبين في الشكل:

sh ovs-ofctl show s1

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=74.898s, table=0, n_packets=24, n_bytes=1680, idle_age=63,
 actions=NORMAL
```

الشكل 51. عرض جدول تدفق s1

حيث نلاحظ أن قيمة actions=normal أي أن المبدل يعمل كمبدل شبكة تقليدية وليس كمبدل SDN.

يمكن حذف مداخل جدول التدفق باستخدام الأمر المبين في الشكل

sh ovs-ofctl del-flows s1

وعندها في حال قمنا باختبار الاتصال فإنه لن ينجح مجدداً.

```
mininet> sh ovs-ofctl del-flows s1
mininet> pingall
*** Ping: testing ping reachability
h1 -> X X
h2 -> X X
h3 -> X X
*** Results: 100% dropped (0/6 received)
mininet>
```

الشكل 52. حذف مداخل جدول التدفق.

إدارة جداول التدفق باستخدام قواعد الطبقة الأولى:

عند إدارة جدول التدفق عن طريق الطبقة الفيزيائية فإنه يتم تحديد المنفذ الذي ستطبق عليه القاعدة أي يتم التعامل مع المضيف من خلال المنفذ الذي يربطه مع المبدل. سيتم العمل على نفس الشبكة في الشكل (43).

في حال كنا نريد فتح اتصال بين المضيفين h1 و h2 فإنه يجب إضافة القاعدتين المبينتين بالشكل:

sh ovs-ofctl add-flow s1

priority=500,in_port=1,action=output:2

والذي يعني إضافة مدخل تدفق إلى المبدل s1 ذو رقم أولوية 500 بحيث الرسالة الواردة عبر المنفذ ذو الرقم 1 (منفذ h1) فإن الحدث الذي سيتم عليها هو إخراجها عبر المنفذ ذو الرقم 2 (منفذ h2).

وليكون الاتصال ثنائي الاتجاه يجب حق قاعدة تدفق معاكسة بحيث مايرد عبر منفذ h2 يتم إخراجه للعقدة h1.

sh ovs-ofctl add-flow s1

priority=500,in_port=2,action=output:1

```
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=1,action=output:2
mininet> sh ovs-ofctl add-flow s1 priority=500,in_port=2,action=output:1
```

الشكل 53. حقن قواعد التدفق على مستوى المنافذ.

في حال قمنا بعرض جدول التدفق:

sh ovs-ofctl del-flows s1

```
mininet> sh ovs-ofctl dump-flows s1
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=246.621s, table=0, n_packets=0, n_bytes=0,
 idle_age=246, priority=500,in_port=1 actions=output:2
 cookie=0x0, duration=239.912s, table=0, n_packets=0, n_bytes=0,
 idle_age=239, priority=500,in_port=2 actions=output:1
```

الشكل 54. محتويات جدول التدفق

نلاحظ أنه توجد ضمنه القواعد التي حقنت ضمن المبدل. في حال قمنا باختبار اتصال بين h1 و h2 فإنه سينجح كما هو مبين في الشكل (52) أما الاتصال بين h1 و h3 سيعاني من الفشل لأنه لا توجد أي قاعدة ستسمح بهذا الاتصال.

```
mininet> h1 ping -c 2 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=0.684 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.036 ms

--- 10.0.0.2 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1000ms
rtt min/avg/max/mdev = 0.036/0.360/0.684/0.324 ms
mininet> h1 ping -c 2 h3
PING 10.0.0.3 (10.0.0.3) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable

--- 10.0.0.3 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1009ms
pipe 2
```

الشكل 55. فتح الاتصال المطلوب.

في حال إضافة تدفق ذو أولوية أعلى فإنه سيتم تجاهل التدفقات الأخرى ضمن المبدل فمثلاً بعد إضافة القاعدة المبينة في الشكل:

sh ovs-ofctl add-flow s1

priority=32768,action=drop

والتي تعني إضافة قاعدة تدفق ذات رقم أولوية 32768 والحدث هو إهمال الرزم.

```
mininet> sh ovs-ofctl add-flow s1 priority=32768,action=drop
```

الشكل 56. حقن تدفق بأولوية عليا.

فإن أي محاولة إجراء للاتصال بين h1 و h2 ستعاني من الفشل لأن المبدل يملك قاعدة ذات أولوية عليا الحدث فيها هو drop أي إهمال أي رزمة ترد إلى المتحكم.

A. إدارة جداول التدفق باستخدام قواعد الطبقة الثانية:

ضمن هذا الجزء سنقوم بتكرار العملية السابقة ولكن بدلاً من اعتماد رقم المنفذ سنستخدم عناوين MAC للمضيفين علماً أن عنونة MAC هي العنونة المستخدمة على مستوى الطبقة الثانية. قبل البدء بالتعامل مع قواعد التدفق نقوم بإلغاء أي قواعد حقنت مسبقاً بالمبدلات باستخدام del–flows. من أجل فتح اتصال بين h1 و h2 سنقوم باستخدام القواعد التالية:

sh ovs–ofctl add–flow s1

dl_src=00:00:00:00:00:02,

dl_dst=00:00:00:00:00:01, actions=output:1

والأمر

sh ovs–ofctl add–flow s1

dl_src=00:00:00:00:00:01,

dl_dst=00:00:00:00:00:02, actions=output:2

حيث نلاحظ أن استخدام عناوين MAC متاح بهذا الشكل بسبب استخدام الخاصية mac–– في عملية إنشاء الشبكة في الشكل (44) كما أنه يتم تحديد عنوان MAC للمصدر من خلال dl_src وعنوان MAC للهدف من خلال dt_dst. ويتم في كل أمر تحديد المنفذ الذي ستخرج منه الرسالة:

```
mininet> sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:02,
dl_dst=00:00:00:00:00:01,actions=output:1
mininet> sh ovs-ofctl add-flow s1 dl_src=00:00:00:00:00:01,
dl_dst=00:00:00:00:00:02,actions=output:2
```

الشكل 57. حقن قواعد التدفق على مستوى عناوين MAC.

في حال إجراء اختبار للاتصال بين h1 وh2 فإن الاتصال يفشل:

```
mininet> h1 ping -c 2 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
From 10.0.0.1 icmp_seq=1 Destination Host Unreachable
From 10.0.0.1 icmp_seq=2 Destination Host Unreachable

--- 10.0.0.2 ping statistics ---
2 packets transmitted, 0 received, +2 errors, 100% packet loss, time 1000ms
pipe 2
```

الشكل 58. فشل الاتصال بين المضيفين.

السبب وراء ذلك أن الأمر ping يتبع البروتوكول ICMP والذي يعمل على مستوى IP وأول ما يقوم به هو إرسال رزمة ARP لإيجاد عنوان MAC للمضيفين المختلفين وإن المبدل المستخدم مع القواعد المضافة له يقوم بترشيح أي حركية خاصة بـ ARP وبما أن بروتوكول ARP يعتمد البث العام فسوف نضيف القاعدة المبينة في الشكل:

sh ovs–ofctl add–flow s1

dl_type=0x806,nw_proto=1,actions=flood

```
mininet> sh ovs-ofctl add-flow s1 dl_type=0x806,nw_proto=1,actions=flood
```

الشكل 59. حقن قاعدة التدفق الخاصة بـ ARP

حيث أن القاعدة ستضيف تدفق على جميع إطارات Ethernet من النوع 0x806 (النوع ARP) حيث يضاف التدفق على جميع منافذ المبدل وإن nw_proto=1 تحدد أن الرزمة هي رزمة طلب ARP أي ARP request والحدث هو إجراء غمر لهذه الرزمة flood. في حال قمنا باختبار الاتصال نلاحظ أنه ينجح بين h1 و h2.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 X
h2 -> h1 X
h3 -> X X
*** Results: 66% dropped (2/6 received)
```

الشكل 60. نجاح الاتصال بين العقد المطلوبة.

B. إدارة جداول التدفق باستخدام قواعد الطبقة الثالثة

ضمن هذا الجزء سنقوم بتكرار العملية السابقة ولكن بدلاً من اعتماد رقم المنفذ أو عنوان MAC سنستخدم عناوين IP للمضيفين. قبل البدء بالتعامل مع قواعد التدفق نقوم بإلغاء أي قواعد حقنت مسبقاً بالمبدلات باستخدام del–flows.

من أجل فتح اتصال بين h1 و h2 سنستخدم القواعد الآتية:

sh ovs–ofctl add–flow s1 priority=500,

dl_type=0x800,nw_src=10.0.0.0/24,nw_dst=

10.0.0.0/24,

actions=normal

```
mininet> sh ovs-ofctl add-flow s1 priority=500,dl_type=0x800,nw_src=10.0.0.0/24,
nw_dst=10.0.0.0/24,actions=normal
```

الشكل 61. حقن قواعد التدفق على مستوى عناوين IP.

حيث يحدد رقم أولوية لقاعدة التدفق و dl_type=0x800 يعني استخدام البروتوكولIP ويحدد عنوان الشبكة المصدر وعنوان الشبكة الهدف.

يجب أيضاً تفعيل ARP ولكن سنتعامل معه بطريقة مختلفة

sh ovs-ofctl add-flow s1 arp
nw_dst=10.0.0.1,actions=output:1
sh ovs-ofctl add-flow s1 arp
nw_dst=10.0.0.2,actions=output:2
sh ovs-ofctl add-flow s1 arp
nw_dst=10.0.0.3,actions=output:3

```
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.1,actions=output:1
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.2,actions=output:2
mininet> sh ovs-ofctl add-flow s1 arp,nw_dst=10.0.0.3,actions=output:3
```

الشكل 62. حقن قاعدة التدفق الخاصة بـ ARP

حيث تم حقن 3 قواعد تدفق ضمن المبدل والتي تحدد عنوان IP الهدف والمنفذ الذي سيتم توجيه الرسالة من خلاله.

في حال قمنا باختبار الاتصال بين عقد الشبكة فنلاحظ نجاح الاتصال.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2 h3
h2 -> h1 h3
h3 -> h1 h2
*** Results: 0% dropped (6/6 received)
```

الشكل 63. نجاح الاتصال بين عقد الشبكة

C. إدارة جداول التدفق باستخدام قواعد الطبقة الرابعة

في هذه الحالة سنقوم بتفعيل أحد المضيفين كمخدم web وليكن h3 وسيتم الاتصال معه من قبل المضيفين h1 و h2 عبر المنفذ رقم 80 (المنفذ الخاص بخدمة http). ولكن يجب في البداية حذف جميع القواعد السابقة باستخدام الأمر del–flows. نقوم في البداية بتفعيل h3 كمخدم وفق الشكل:

h3 python –m SimpleHTTPServer 80 &

```
mininet> h3 python -m SimpleHTTPServer 80 &
```

الشكل 64. تشغيل المضيف h3 كمخدم ويب.

ونقوم بتفعيل ARP بطريقة مبسطة:

sh ovs–ofctl add–flow s1 arp,actions=normal

```
mininet> sh ovs-ofctl add-flow s1 arp,actions=normal
```

الشكل 65. حقن قاعدة التدفق الخاصة بـ ARP

سنقوم بإضافة قاعدة والتي تمرر حركية TCP (nw_proto=6) مع منفذ هدف رقمه 80 إلى المنفذ رقم 3 في المبدل:

sh ovs-ofctl add-flow s1
priority=500,dl_type=0x800,nw_proto=6,tp_dst=80,actions=output:3

```
mininet> sh ovs-ofctl add-flow s1 priority=500,dl_type=0x800,nw_proto=
6,tp_dst=8,actions=output:3
```

الشكل 66. حقن قاعدة التدفق الخاصة ب TCP ورقم المنفذ

أخيراً نضيف قاعدة بسيطة تمكن المخدم من الرد على اتصالات الزبائن

sh ovs-ofctl add-flow s1 arp,actions=normal
sh ovs-ofctl add-flow s1
priority=800,ip,nw_src=10.0.0.3,actions=normal

```
mininet> sh ovs-ofctl add-flow s1 priority=800,ip,nw_src=10.0.0.3,acti
ons=normal
```

الشكل 67. حقن قاعدة خاصة لرد المخدم

وأخيراً نختبر الاتصال باستخدام الأمر wget

```
mininet> h1 wget h3
--2022-05-23 11:27:50--  http://10.0.0.3/
Connecting to 10.0.0.3:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1428 (1.4K) [text/html]
Saving to: 'index.html.7'

 0% [                                       ] 0           --.-K/s
100%[======================================>] 1,428       --.-K/s   in
 0.02s

2022-05-23 11:27:50 (62.8 KB/s) - 'index.html.7' saved [1428/1428]
```

الشكل 68. نجاح الاتصال

VI. قياس التأخير والإنتاجية ضمن الشبكة

يمكن قياس التأخير في شبكة SDN وبرنامج mininet من خلال رسائل Ping التي تتبع البروتوكول ICMP، مثلا في حال لدينا الشبكة:

sudo mn

وقمنا باستخدام الأمر:

h1 ping –c 4 h2

فيظهر الشكل قيم التأخير للرزم ضمن الشبكة.

```
mininet> h1 ping -c 4 h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=1.75 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.461 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.031 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.026 ms

--- 10.0.0.2 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 2999ms
rtt min/avg/max/mdev = 0.026/0.568/1.755/0.707 ms
```

الشكل 69. حساب التأخير ضمن الشبكة

حيث نحصل على القيمة العظمى للتأخير 1.755 والقيمة الدنيا 0.026 والقيمة الوسطية 0.568 وتكون عادة الرزمة الأولى بأعلى تأخير لأنها ترسل إلى المتحكم من أجل أن يقوم بحقن قاعدة تدفق خاصة برزمة ping بين الجهازين.

يمكن قياس **الإنتاجية** باستخدام الأداة iperf من أجل القيام بقياس عرض الحزمة المتاح من أجل إرسال معين. عندما نريد استخدام إرسال TCP أو UDP يجب أن يكون أحد المضيفين هو مخدم والآخر مضيف.

سنقوم بعملية التحليل من خلال المثال التالي والذي يمثل شبكة بمبدل وحيد مع مضيفين متصلين معه:

sudo mn --topo=single,2

```
mininet@mininet-vm:~$ sudo mn --topo=single,2
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
```

الشكل 70. تشغيل الشبكة

نقوم بالدخول إلى المستخدمين h1 و h2 من خلال الأمر:

xterm h1 h2

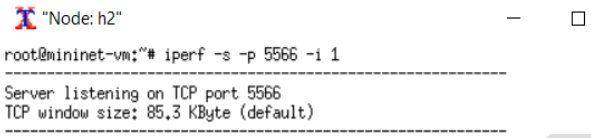
ونقوم بتشغيل أحد المضيفين وليكن h2 كمخدم من خلال الأمر:

iperf –s –p 5566 –i 1

حيث : s للدلالة على أنه مخدم Server

P: رقم المنفذ الخاص بالاتصال

i: الزمن الفاصل بين عرض الرسائل



الشكل 71. تشغيل الجهاز كمخدم

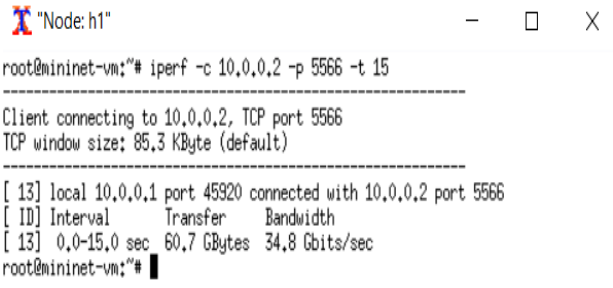
ونقوم بتشغيل المضيف الأخر h1 كزبون من خلال الأمر

iperf –c 10.0.0.2 –p 5566 –t 15

حيث : c للدلالة على أنه زبون client

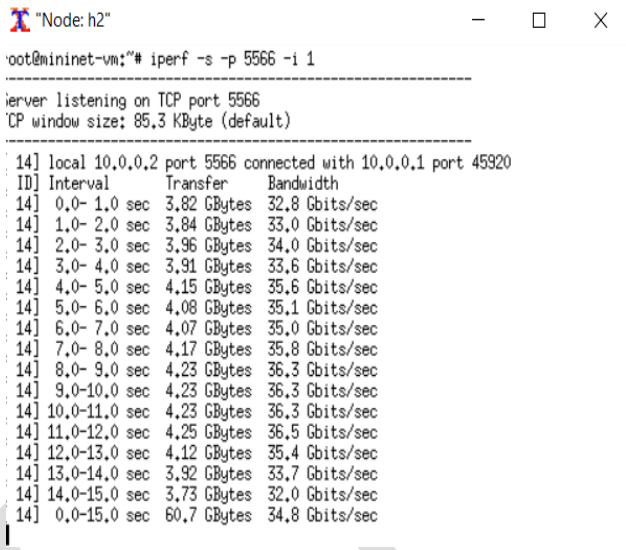
P: رقم المنفذ الخاص بالاتصال

t: زمن الاتصال مع المخدم



الشكل 72. تشغيل الجهاز كزبون

وبعد انتهاء الاتصال تصبح واجهة المخدم



الشكل 73. الرسائل الواردة للمخدم

في حال كنا نريد رسم مخطط الانتاجية سنعيد التقييم ولكن نقوم بحفظ النتائج في ملف. نقوم ببعض المعالجة ومن ثم تستخدم الأداة gnuplot للرسم. نقوم بتشغيل h2 كمخدم من خلال الأمر (تحفظ النتيجة في الملف result)

iperf –s –p 5566 –i 1 > result

نقوم بتشغيل المضيف الأخر h1 كزبون من خلال الأمر

iperf –c 10.0.0.2 –p 5566 –t 15

عملية المعالجة ضمن المخدم بعد إنهاء الاتصال من خلال الأمر:

cat result | grep sec | head –15 | tr – “ “ |

awk ‘{print $4,$8}’ > new_result

حيث: cat من أجل الاقتطاع من الملف.

grep sec من أجل التعامل مع الثواني.

head: من أجل تحديد عدد الأسطر المراد أخذها.

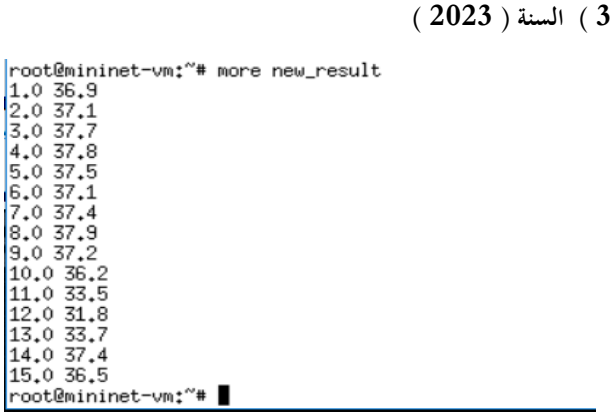
tr: من أجل استبدال الرمز – بفراغ.

Print: من أجل الحصول على العمودين الرابع والثامن فقط.

new_result: اسم الملف بعد الاقتطاع.

يمكن عرض الملف الجديد من خلال الأمر:

more new_result

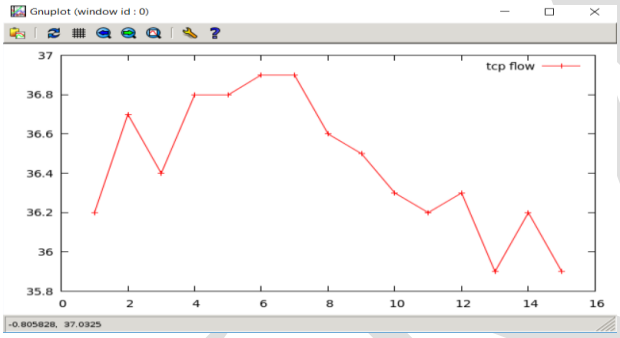


الشكل 74. النتائج بعد الاقتصاص

من أجل الرسم ندخل إلى الأداة gnuplot ضمن واجهة المخدم بذكر اسمها فقط ومن ثم نستخدم الأمر:

plot “new_result” title “tcp flow” with linespoints

نحصل على

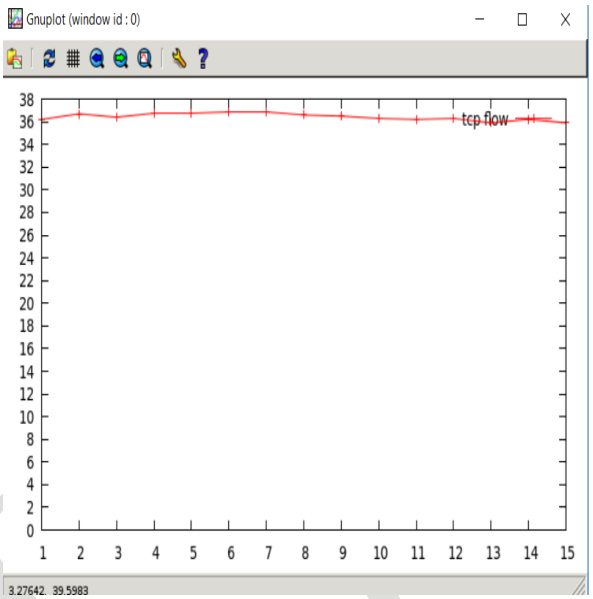


الشكل 75. مخطط الانتاجية

يمكن استخدام أوامر لتحسين شكل المخطط

```
gnuplot> set xrange[1:15]
gnuplot> set xtics 1,1,15
gnuplot> set yrange[0:38]
gnuplot> set ytics 0,2,38
gnuplot> replot
gnuplot> set xlabel “time (sec)”
gnuplot> set ylabel “throughput (sec)”
gnuplot>replot
```

فنحصل على



الشكل 76. مخطط الانتاجية بعد التعديل

المراجع:

[1]. http://mininet.org/, last visit, 14, October, 2022
[2]. https://www.virtualbox.org/, last visit, 14 October, 2022
[3]. https://www.putty.org/, last visit, 14 October, 2022
[4]. https://sourceforge.net/projects/xming/, last visit 14, October, 2022