دانشگاه صنعتی اصفهان

دانشکده مهندسی برق و کامپیوتر

عنوان: پروژه‌ی اول درس شبکه‌های کامپیوتری ۲

نام و نام خانوادگی: علیرضا ابره فروش

شماره دانشجویی: ۹۸۱۶۶۰۳

نیم سال تحصیلی: بهار ۱۴۰۰/۱۴۰۱

مدرّس: دکتر مسعودرضا هاشمی

۱

جهت نصب این نرم‌افزار در لینوکس به ترتیب زیر عمل می‌کنیم:
ابتدا گیت را نصب می‌کنیم و سپس مخزن شامل نرم‌افزار در گیت‌هاب را کلون می‌کنیم.

```
$ sudo apt install git
$ sudo git clone https://github.com/mininet/mininet
```

سپس وارد پوشه‌ی mininet/util شده و به ترتیب زیر نرم‌افزار را نصب می‌کنیم.

```
$ ./install.sh -a
$ sudo mn –test pingall
```

۲

برای این کار دستور زیر را وارد می‌کنیم.

```
$ sudo mn –topo single,3
```

خروجی زیر نشان می‌دهد که توپولوژی Single با ۳ میزبان ایجاد شد.

```
alireza@alireza-VirtualBox:~/Documents/Projects/CN2_PRJ1$ sudo mn --topo single,3
*** 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 ...
*** Starting CLI:
mininet>
```

شکل ۱: خروجی دستور *sudo mn –topo single,3*

۱.۲ دستور **nodes**

دستور nodes، نودهای موجود در شبکه را نشان می‌دهد.

```
mininet> nodes
available nodes are:
c0 h1 h2 h3 s1
mininet>
```

شکل ۲: خروجی دستور *nodes*

همانطور که می‌بینیم، دستور nodes، همه‌ی نودهای حاضر در شبکه‌ی ساخته شده اعم از کنترلر، سوئیچ و میزبان را لیست می‌کند. در اینجا یک کنترلر (c0)، سه میزبان (h1، h2، h3) و یک سوئیچ (s1) در شبکه موجودند.

۲.۲ دستور **net**

دستور net، لینک‌های موجود بین دیوایس‌های Mininet در شبکه را برای درک بهتر توپولوژی شبکه نشان می‌دهد.

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
h3 h3-eth0:s1-eth3
s1 lo:  s1-eth1:h1-eth0 s1-eth2:h2-eth0 s1-eth3:h3-eth0
c0
mininet>
```

شکل ۳: خروجی دستور *net*

همانطور که می‌بینیم، دستور net، همه‌ی لینک‌های موجود در شبکه را لیست می‌کند. در اینجا به ترتیب اینترفیسِ eth0 از میزبانِ h1 به eth1 از سوئیچِ s1، اینترفیسِ eth0 از میزبانِ h2 به eth2 از سوئیچِ s1، اینترفیسِ eth0 از میزبانِ h3 به eth3 از سوئیچِ s1 وصل است. سوئیچِ s1 دارای یک اینترفیسِ loopback به نام lo است. سوئیچِ s1 به اینترفیسِ h1-eth0 از طریق اینترفیسِ s1-eth1 متصل است. سوئیچِ s1 به اینترفیسِ h2-eth0 از طریق اینترفیسِ s1-eth2 متصل است. سوئیچِ s1 به اینترفیسِ h3-eth0 از طریق اینترفیسِ s1-eth3 متصل است. کنترلرِ c0 مغز شبکه است که یک دانش جامع از درباره‌ی شبکه دارد. یک کنترلر سوئیچ‌ها را در نحوه‌ی forward یا drop کردن بسته‌ها در شبکه راهنمایی می‌کند.

۳.۲ دستور **dump**

دستور dump، اطلاعات از قبیل نوع دیوایس، آدرس آی‌پی‌ها و شناسه پروسس‌های نودهای موجود در شبکه را نشان می‌دهد.

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=4141>
<Host h2: h2-eth0:10.0.0.2 pid=4143>
<Host h3: h3-eth0:10.0.0.3 pid=4145>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None,s1-eth3:None pid=4150>
<Controller c0: 127.0.0.1:6653 pid=4134>
mininet>
```

شکل ۴: خروجی دستور *dump*

همانطور که می‌بینیم، دستور dump، اطلاعات دیوایس‌های موجود در شبکه را لیست می‌کند. در اینجا به ترتیب میزبانِ h1 دارای آدرس آی‌پیِ 10.0.0.1 و پروسس آی‌دیِ ۴۱۴۱، میزبانِ h2 دارای آدرس آی‌پیِ 10.0.0.2 و پروسس آی‌دیِ ۴۱۴۳، میزبانِ h3 دارای

آدرس آی‌پیِ 10.0.0.3 و پروسس آی‌دیِ ۴۱۴۵، سوئیچِ s1 دارای آدرس آی‌پیِ لوپ‌بک 127.0.0.1 و پروسس آی‌دیِ ۴۱۵۰، کنترلرِ c0 دارای آدرس آی‌پیِ 127.0.0.1 و پروسس آی‌دیِ ۴۱۳۴،

۳

۱.۳

Tree ۱.۱.۳

توپولوژیِ Tree که تحت عنوان star bus نیز شناخته می‌شود، یک ساختار درخت‌گونه دارد که در آن همه‌ی کامپیوترها به یکدیگر همچون شاخه‌های درخت به درخت متصل شده‌اند. در شبکه‌های کامپیوتری، این توپولوژی ترکیبی از توپولوژی‌های Star و Bus است. مزیت اصلی این توپولوژی انعطاف‌پذیری (flexibility) و مقیاس‌پذیری (Scalability) است. این توپولوژی به عنوان ساده‌ترین توپولوژی در میان همه توپولوژی‌های دارای تنها یک روتر بین هر دو نود داخل شبکه است. الگوی اتصال شبیه درختی است که در آن همه شاخه ها از یک ریشه سرچشمه می گیرند و از این رو توپولوژی Tree یکی از محبوب ترین توپولوژی‌ها در بین پنج توپولوژی شبکه است.

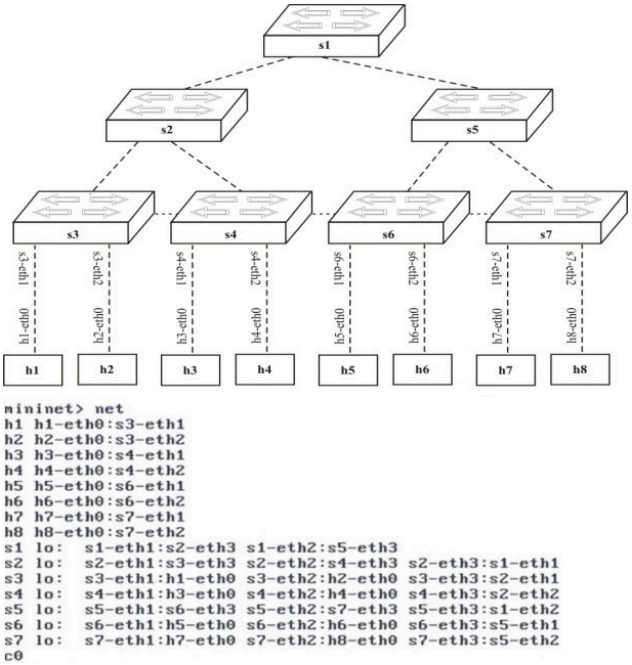
مزایا:

- این توپولوژی ترکیبی از توپولوژی Star و Bus است.
- این توپولوژی چینش داده های سلسله مراتبی و مرکزی گره ها را فراهم می‌کند.
- از آنجایی که نودهای برگ می‌توانند یک یا چند گره را در زنجیره سلسله مراتبی اضافه کنند، این توپولوژی مقیاس‌پذیری بالایی را فراهم می‌کند.
- اگر یکی از نودهای آن‌ها آسیب ببیند یا کار نکند، نودهای دیگر در یک شبکه تحت تأثیر قرار نمی‌گیرند.
- توپولوژی Tree تعمیر و نگهداری آسان را فراهم می‌کند و به آسانی می‌توان عیب‌یابی را انجام داد.
- توپولوژی قابل فراخوانی (نودهای برگ می‌توانند نودهای بیشتری را نگه دارند.)
- توسط چندین وندورِ سخت‌افزار و نرم‌افزار پشتیبانی می‌شود.
- سیم کشی نقطه‌به‌نقطه برای بخش‌های جداگانه.

معایب:

- پیکربندی این شبکه در مقایسه با سایر توپولوژی‌های شبکه بسیار دشوار است.
- طول یک بخش محدود است و محدودیت سگمنت به نوع کابل‌کشی استفاده شده بستگی دارد.
- به دلیل وجود تعداد زیادی نود، عملکرد شبکه توپولوژی Tree کمی کند می‌شود.
- اگر کامپیوتر سطح اول خطا داشته باشد، کامپیوتر سطح بعدی نیز دچار مشکل خواهد شد.
- در مقایسه با توپولوژی Star و Ring، به تعداد زیادی کابل نیاز دارد.

- از آنجایی که داده‌ها باید از کابل مرکزی منتقل شوند، ترافیک شبکه متراکمی ایجاد می‌کند.
- Backbone به عنوان نقطه شکست کل بخش شبکه ظاهر می‌شود.
- تعمیر این توپولوژی بسیار پیچیده است.
- هزینه تاسیسات نیز افزایش می‌یابد.
- اگر بخش عمده‌ای از نودها در این شبکه اضافه شوند، تعمیر و نگهداری پیچیده خواهد شد.



شکل ۵: ساختار توپولوژیِ *Tree*

۲.۱.۳ **Linear**

توپولوژی Linear از یک کابل اصلی با یک پایانه در هر انتها تشکیل شده است. تمام نودها (سرور فایل، ورک‌استیشن‌ها و تجهیزات جانبی) به کابل خطی متصل هستند. توپولوژی خطی شامل k سوئیچ و k میزبان است. همچنین یک پیوند بین هر سوئیچ و هر میزبان در میان سوئیچ‌ها ایجاد می‌کند. این توپولوژی در واقع نوعی توپولوژی شبکه است که در آن هر دستگاه یکی پس از دیگری در یک زنجیره متوالی به هم متصل می‌شود. در این حالت Bus اتصال شبکه بین دستگاه‌ها است. اگر هر پیوندی در زنجیره شبکه قطع شود، تمام انتقال شبکه متوقف می‌شود. برای شبکه‌های کوچک کارآمد است. زیرا راه اندازی آن ساده است و از کابل‌های کوتاه‌تری استفاده می‌کند زیرا هر دستگاه به دستگاه بعدی متصل می‌شود. با این حال، این یک راه حل ضعیف برای شبکه‌های بزرگتر است، زیرا کل شبکه به هر اتصال متکی است و با اضافه شدن دستگاه‌های بیشتر، سرعت شبکه کاهش می‌یابد.

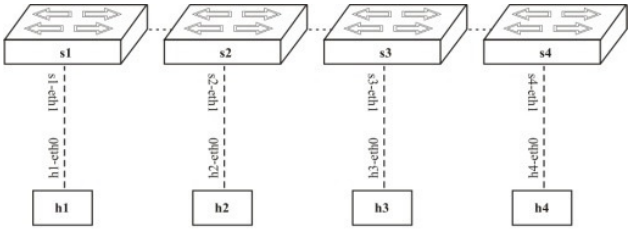
مزایا:

- اتصال کامپیوتر یا دستگاه جانبی به Bus خطی آسان است.

- نسبت به توپولوژی Star به طول کابل کمتری نیاز دارد.

معایب:

- در صورت قطع شدن کابل اصلی، کل شبکه قطع می‌شود.
- ترمیناتورها در هر دو انتهای کابل Backbone مورد نیاز هستند.
- اگر کل شبکه قطع شود، شناسایی مشکل دشوار است.
- قرار نیست به عنوان یک راه حل مستقل در یک ساختمان بزرگ استفاده شود.




```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2
s4 lo:  s4-eth1:h4-eth0 s4-eth2:s3-eth3
c0
```

شکل ۶: ساختار توپولوژیِ *Linear*

۳.۱.۳ Reversed

این یک توپولوژی ساده با یک سوئیچ Openflow و k میزبان است. همچنین یک پیوند بین سوئیچ و k میزبان ایجاد می‌کند. در واقع این توپولوژی همان توپولوژیِ Single است و فقط ترتیب اتصال اینترفیس‌ها برعکس است. در توپولوژی‌های ستاره‌ای (Single و Reversed) همه دستگاه‌ها از طریق یک کابل به یک سوئیچ واحد متصل می‌شوند. این سوئیچ نود مرکزی است و تمام نودهای دیگر به نود مرکزی متصل هستند.

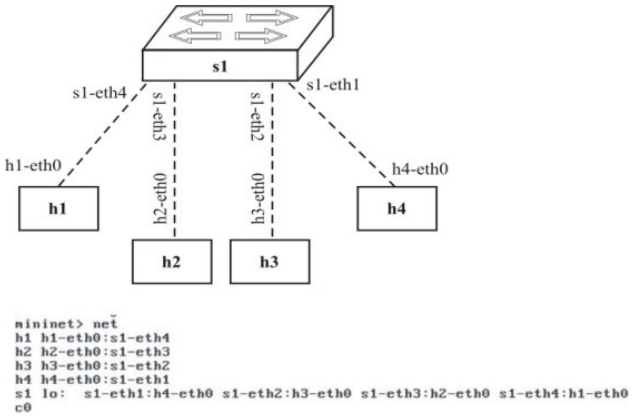
مزایا:

- اگر n دستگاه در این توپولوژی به یکدیگر متصل شده باشند، تعداد کابل های مورد نیاز برای اتصال آنها n است. بنابراین، تنظیم آن آسان است.
- هر دستگاه برای اتصال به سوئیچ فقط به ۱ پورت نیاز دارد، بنابراین تعداد کل پورت‌های مورد نیاز n است.

معایب:

- اگر سوئیچ که کل توپولوژی به آن متکی است از کار بیفتد، کل سیستم از کار می‌افتد.
- هزینه نصب بالاست.

- عملکرد بر اساس متمرکز کننده منفرد یعنی سوئیچ است.



شکل ۷: ساختار توپولوژیِ *Reversed*

۲.۳

۱.۲.۳ Tree

برای ساخت شبکه‌ای با توپولوژیِ Tree با عمقِ ۳ (۸ میزبان) دستور زیر را اجرا می‌کنیم.

```
$ sudo mn –topo tree,depth=3
```

```
alireza@alireza-VirtualBox:~$ sudo mn --topo tree,depth=3
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2 h3 h4 h5 h6 h7 h8
*** Adding switches:
s1 s2 s3 s4 s5 s6 s7
*** Adding links:
(s1, s2) (s1, s5) (s2, s3) (s2, s4) (s3, h1) (s3, h2) (s4, h3) (s4, h4) (s5, s6) (s5, s7) (s6, h5) (s6, h6) (s7, h7) (s7, h8)
*** Configuring hosts
h1 h2 h3 h4 h5 h6 h7 h8
*** Starting controller
c0
*** Starting 7 switches
s1 s2 s3 s4 s5 s6 s7 ...
*** Starting CLI:
mininet>
```

شکل ۸: خروجی دستورِ *sudo mn –topo tree,depth=3*

```
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 h5 h6 h7 h8 s1 s2 s3 s4 s5 s6 s7
```

شکل ۹: *nodes*

```
mininet> net
h1 h1-eth0:s3-eth1
h2 h2-eth0:s3-eth2
h3 h3-eth0:s4-eth1
h4 h4-eth0:s4-eth2
h5 h5-eth0:s6-eth1
h6 h6-eth0:s6-eth2
h7 h7-eth0:s7-eth1
h8 h8-eth0:s7-eth2
s1 lo:  s1-eth1:s2-eth3 s1-eth2:s5-eth3
s2 lo:  s2-eth1:s3-eth3 s2-eth2:s4-eth3 s2-eth3:s1-eth1
s3 lo:  s3-eth1:h1-eth0 s3-eth2:h2-eth0 s3-eth3:s2-eth1
s4 lo:  s4-eth1:h3-eth0 s4-eth2:h4-eth0 s4-eth3:s2-eth2
s5 lo:  s5-eth1:s6-eth3 s5-eth2:s7-eth3 s5-eth3:s1-eth2
s6 lo:  s6-eth1:h5-eth0 s6-eth2:h6-eth0 s6-eth3:s5-eth1
s7 lo:  s7-eth1:h7-eth0 s7-eth2:h8-eth0 s7-eth3:s5-eth2
c0
```

شکل ۱۰: *net*

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=10281>
<Host h2: h2-eth0:10.0.0.2 pid=10283>
<Host h3: h3-eth0:10.0.0.3 pid=10285>
<Host h4: h4-eth0:10.0.0.4 pid=10287>
<Host h5: h5-eth0:10.0.0.5 pid=10289>
<Host h6: h6-eth0:10.0.0.6 pid=10291>
<Host h7: h7-eth0:10.0.0.7 pid=10293>
<Host h8: h8-eth0:10.0.0.8 pid=10295>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=10300>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=10303>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None pid=10306>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None,s4-eth3:None pid=10309>
<OVSSwitch s5: lo:127.0.0.1,s5-eth1:None,s5-eth2:None,s5-eth3:None pid=10312>
<OVSSwitch s6: lo:127.0.0.1,s6-eth1:None,s6-eth2:None,s6-eth3:None pid=10315>
<OVSSwitch s7: lo:127.0.0.1,s7-eth1:None,s7-eth2:None,s7-eth3:None pid=10318>
<Controller c0: 127.0.0.1:6653 pid=10274>
mininet>
```

شکل ۱۱: *dump*

۲.۲.۳ **Linear**

برای ساخت شبکه‌ای با توپولوژیِ Linear با ۱ کنترلر، ۴ میزبان و ۴ سوئیچ دستور زیر را اجرا می‌کنیم.

```
$ sudo mn –topo linear,4
```

```
alireza@alireza-VirtualBox:~$ sudo mn --topo linear,4
[sudo] password for alireza:
*** 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:
mininet>
```

شکل ۱۲: خروجی دستورِ *sudo mn –topo linear,4*

```
mininet> nodes
available nodes are:
c0 h1 h2 h3 h4 s1 s2 s3 s4
```

شکل ۱۳: *nodes*

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s2-eth1
h3 h3-eth0:s3-eth1
h4 h4-eth0:s4-eth1
s1 lo:  s1-eth1:h1-eth0 s1-eth2:s2-eth2
s2 lo:  s2-eth1:h2-eth0 s2-eth2:s1-eth2 s2-eth3:s3-eth2
s3 lo:  s3-eth1:h3-eth0 s3-eth2:s2-eth3 s3-eth3:s4-eth2
s4 lo:  s4-eth1:h4-eth0 s4-eth2:s3-eth3
c0
```

شکل ۱۴: *net*

```
mininet> dump
<Host h1: h1-eth0:10.0.0.1 pid=10630>
<Host h2: h2-eth0:10.0.0.2 pid=10632>
<Host h3: h3-eth0:10.0.0.3 pid=10634>
<Host h4: h4-eth0:10.0.0.4 pid=10636>
<OVSSwitch s1: lo:127.0.0.1,s1-eth1:None,s1-eth2:None pid=10641>
<OVSSwitch s2: lo:127.0.0.1,s2-eth1:None,s2-eth2:None,s2-eth3:None pid=10644>
<OVSSwitch s3: lo:127.0.0.1,s3-eth1:None,s3-eth2:None,s3-eth3:None pid=10647>
<OVSSwitch s4: lo:127.0.0.1,s4-eth1:None,s4-eth2:None pid=10650>
<Controller c0: 127.0.0.1:6653 pid=10623>
mininet>
```

شکل ۱۵: *dump*

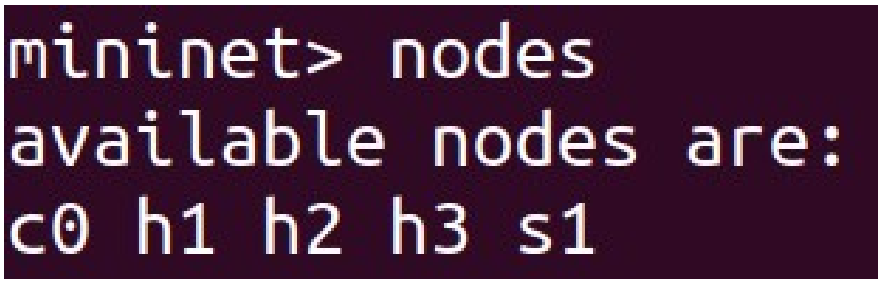
۳.۲.۳ **Reversed**

برای ساخت شبکه‌ای با توپولوژیِ Reveresed با ۱ کنترلر، ۴ میزبان و ۴ سوئیچ دستور زیر را اجرا می‌کنیم.

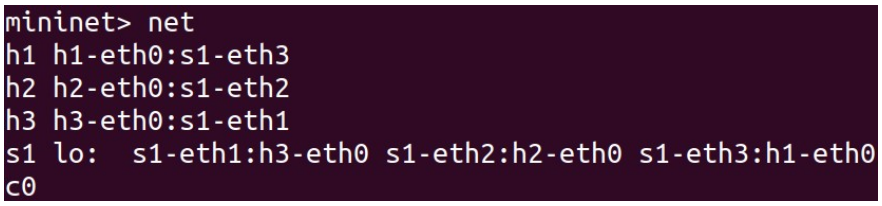
$ sudo mn –topo reversed,3

```
alireza@alireza-VirtualBox:~$ sudo mn --topo reversed,3
*** 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 ...
*** Starting CLI:
mininet>
```

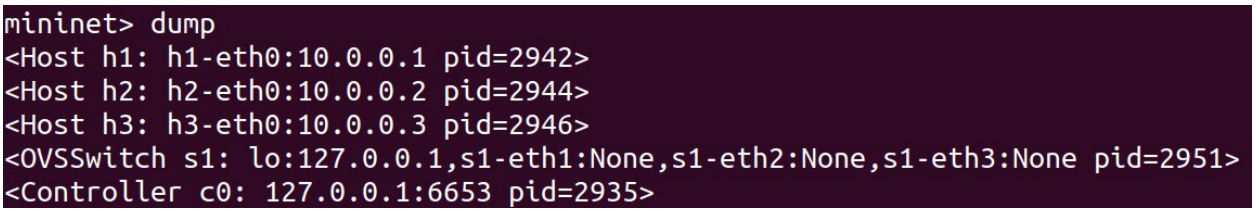
شکل ۱۶: خروجی دستورِ *sudo mn –topo reversed,3*



شکل ۱۷: *nodes*



شکل ۱۸: *net*



شکل ۱۹: *dump*

منابع

[1] http://mininet.org/overview/

[2] https://www.geeksforgeeks.org/advantages-and-disadvantages-of-tree-topology/

[3] https://www.guru99.com/type-of-network-topology.html

[4] https://slsdn.blogspot.com/2020/05/topologies-in-mininet.html

[5] https://fcit.usf.edu/network/chap5/chap5.htm