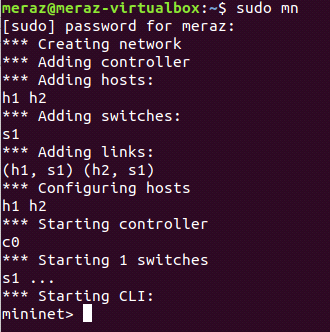
**MININET WALKTHROUGH**

**Start a minimal topology and enter the CLI:**

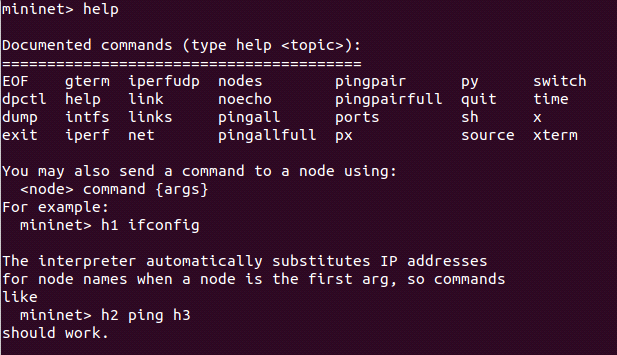


The default topology is the **minimal** topology, which includes one OpenFlow kernel switch connected to two hosts, plus the OpenFlow reference controller. This topology could also be specified on the command line with **--topo=minimal**. Other topologies are also available out of the box; see the **--topo** section in the output of **mn -h**.

All four entities (2 host processes, 1 switch process, 1 basic controller) are now running in the VM. The controller can be outside the VM, and instructions for that are at the bottom.

If no specific test is passed as a parameter, the Mininet CLI comes up.

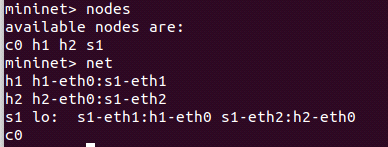
**Display Mininet CLI commands:**



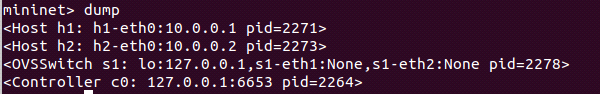
**Display nodes:**



**Display links:**



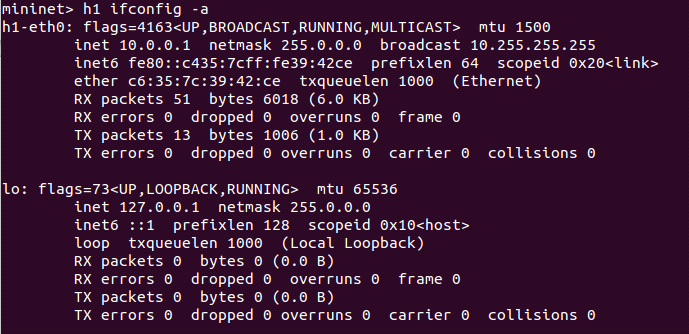
**Dump information about all nodes:**



You should see the switch and two hosts listed.

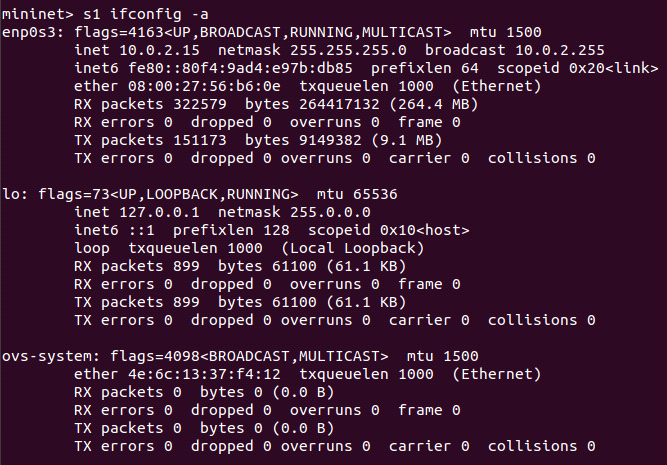
If the first string typed into the Mininet CLI is a host, switch or controller name, the command is executed on that node.

**Run a command on a host process:**

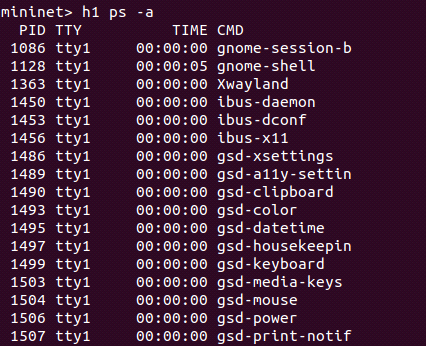


You should see the host’s **h1-eth0** and loopback (**lo**) interfaces. Note that this interface (**h1-eth0**) is not seen by the primary Linux system when **ifconfig** is run, because it is specific to the network namespace of the host process.

In contrast, the switch by default runs in the root network namespace, so running a command on the “switch” is the same as running it from a regular terminal:



Note that only the network is virtualized; each host process sees the same set of processes and directories. For example, print the process list from a host process:

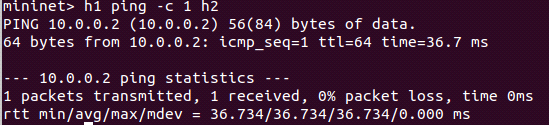


This should be the exact same as that seen by the root network namespace:

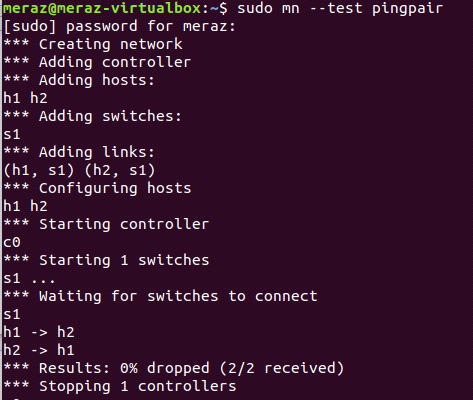
If a string appears later in the command with a node name, that node name is replaced by its IP address; this happened for h2.

You should see OpenFlow control traffic. The first host ARPs for the MAC address of the second, which causes a **packet\_in** message to go to the controller. The controller then sends a **packet\_out** message to flood the broadcast packet to other ports on the switch (in this example, the only other data port). The second host sees the ARP request and sends a reply. This reply goes to the controller, which sends it to the first host and pushes down a flow entry.

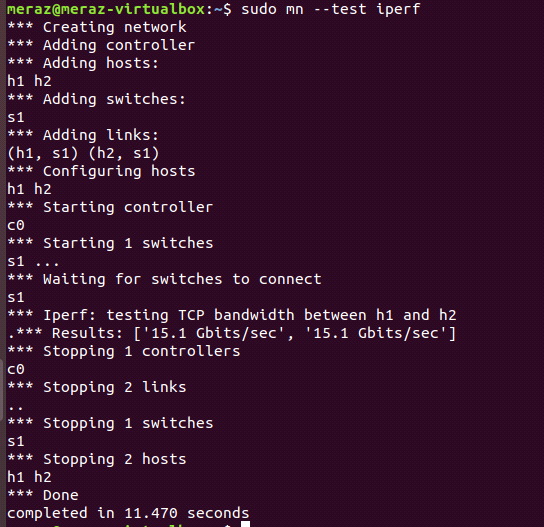
Now the first host knows the MAC address of the second, and can send its ping via an ICMP Echo Request. This request, along with its corresponding reply from the second host, both go the controller and result in a flow entry pushed down (along with the actual packets getting sent out).



**Run a regression test:**



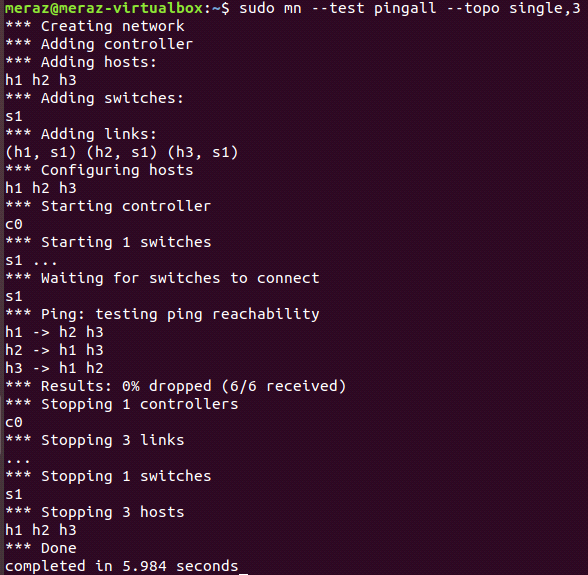
his command created a minimal topology, started up the OpenFlow reference controller, ran an all-pairs-**ping** test, and tore down both the topology and the controller.

ip

This command created the same Mininet, ran an iperf server on one host, ran an iperf client on the second host, and parsed the bandwidth achieved.

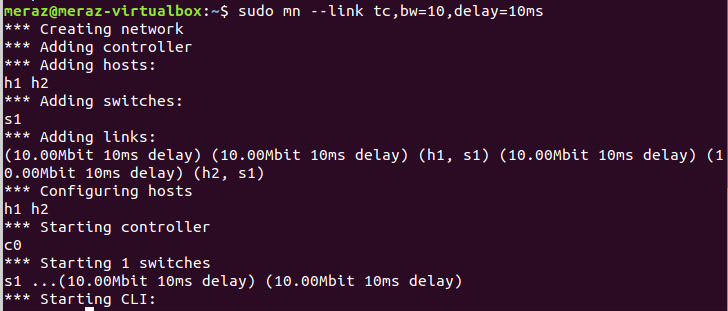
The default topology is a single switch connected to two hosts. You could change this to a different topo with **--topo**, and pass parameters for that topology’s creation. For example, to verify all-pairs ping connectivity with one switch and three hosts:

**Run a regression test:**

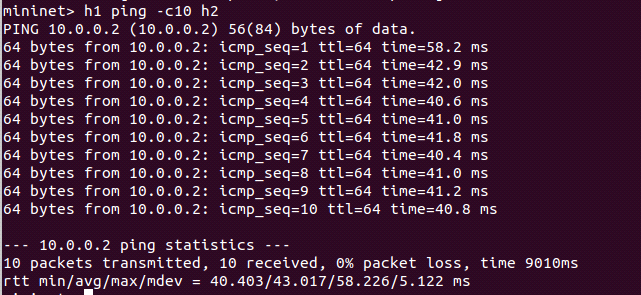


Parametrized topologies are one of Mininet’s most useful and powerful features.

Mininet 2.0 allows you to set link parameters, and these can even be set automatially from the command line:



If the delay for each link is 10 ms, the round trip time (RTT) should be about 40 ms, since the ICMP request traverses two links (one to the switch, one to the destination) and the ICMP reply traverses two links coming back.



**Python Interpreter**

If the first phrase on the Mininiet command line is **py**, then that command is executed with Python. This might be useful for extending Mininet, as well as probing its inner workings. Each host, switch, and controller has an associated Node object.

**At the Mininet CLI, run:**



### SSH daemon per host

One example that may be particularly useful runs an SSH daemon on every host:

