Provisioning experiments in NCL testbed

NCL's testbed provisioning system is built based on DETERLab (https://www.deterlab.net/) and there are numerous tutorials and recipes online that provide a comprehensive guide on how one can create experiments in DETERLab platform.

For a very quick tutorial on creation of an experiment in NCL please refer to NCL tutorial page at https://ncl.sg/tutorials. For further guide with comprehensive details a good place to start is DETERLab's documentation (http://docs.deterlab.net/)

Note that in order to create an experiment with SDN switches involved, you will need to substitute your NS file with one that is written with regards to the following explanations.

Including SDN switches to NCL testbed experiments

In order to define an OpenFlow(v1.3) enabled switch using BNVirt you will need to set the hardware of your switch node as "ofswitch" like:

```
set switch1 [$ns node]
tb-set-hardware $switch1 ofswitch
```

This way the provisioning system (DETER in this case) will understand that you have an OF-enabled switch in your topology and will kickstart the BNVirt module to carry out the SDN provisioning processes.

In the same way you will need to declare a controller node and use tb-set-hardware to declare that particular node as an SDN controller as shown below:

```
set controller [$ns node]
tb-set-hardware $controller ofcontrol
```

Please note that you do not need to connect the controller to the SDN switches. The BNVirt will carry out the out-of-band control connections between the controller and the switches automatically. Currently, an experiment can have only a single SDN Controller node (Multi-Controller topologies are currently not supported)

DETER uses TCL (the language used for writing NS files that describe the simulation scenario) so you basically have all the power that TCL has to offer. A simple topology example can further elaborate steps needed for creating an experiment with SDN switches.

Please note that you will need to assign IP addresses to the nodes connected to SDN switches manually by logging into those nodes via SSH. Otherwise the nodes that are connected to an SDN switch will be assigned IP addresses in different subnets by default. You can login to the controller node, and run any standard OpenFlow controller of your choice to control your SDN Network.

What is not supported as of now?

1) Currently, we do not support having multiple controllers in a single experiment. However, we will consider adding this based on demand.
2) We do not support nsfile having "for" or "while" or similar loop constructs in SDN.

Example 1: Single SDN switch connected to two nodes.

Topology schema:

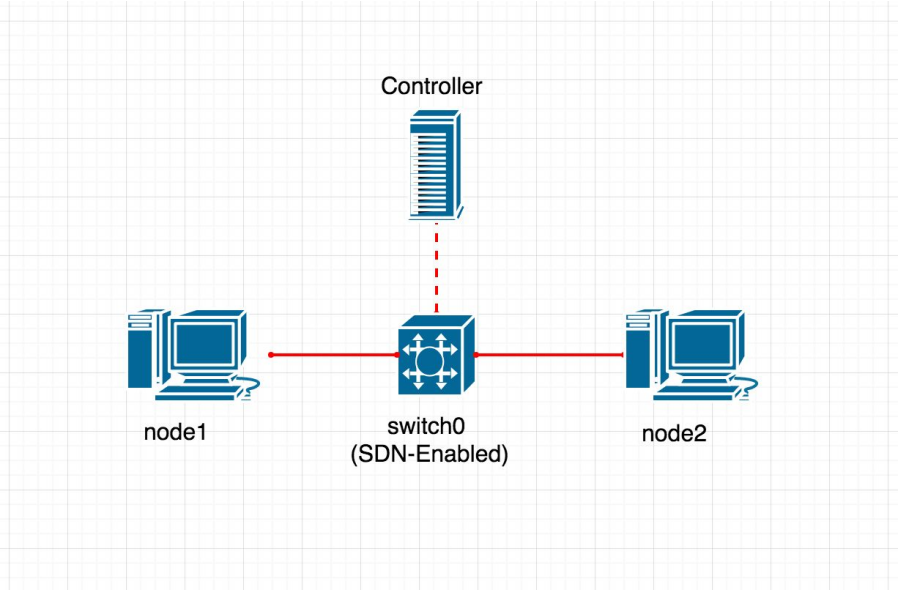

Figure 1: Single SDN switch connected to two nodes

Input NS script:

```
### Start of the script

# Create a new experiment
set ns [new Simulator]
# Enable tb commands (an extension to TCL)
source tb_compat.tcl

# Define a node (a server node)
set node1 [$ns node]
# Same as the above
set node2 [$ns node]

# Define a switch
set switch0 [$ns node]
# Declare that the switch0 is an OpenFlow Enabled switch
tb-set-hardware $switch0 ofswitch

# Connect node1 to switch0
set link1 [$ns duplex-link $switch0 $node1 1Gb 0ms DropTail]
# Connect node2 to switch0
```

```
set link2 [$ns duplex-link $switch0 $node2 1Gb 0ms DropTail]

# [Optional] Define a node named “controller”
# If you define a controller you can set a custom image
# otherwise the default image for the controller will be loaded
set controller [$ns node]
# If you have defined a controller you have to declare it as below
# Declare that “controller” is an SDN Controller
tb-set-hardware $controller ofcontrol

# Boiler plates
$ns rtproto Static
$ns run
### End of the script
```

Example 2: Tree Topology

To further elaborate, here is a another example. In this example we have a tree topology with 2 levels and a branching factor of 2 and four (4) server nodes that are connected to the leaf switches.

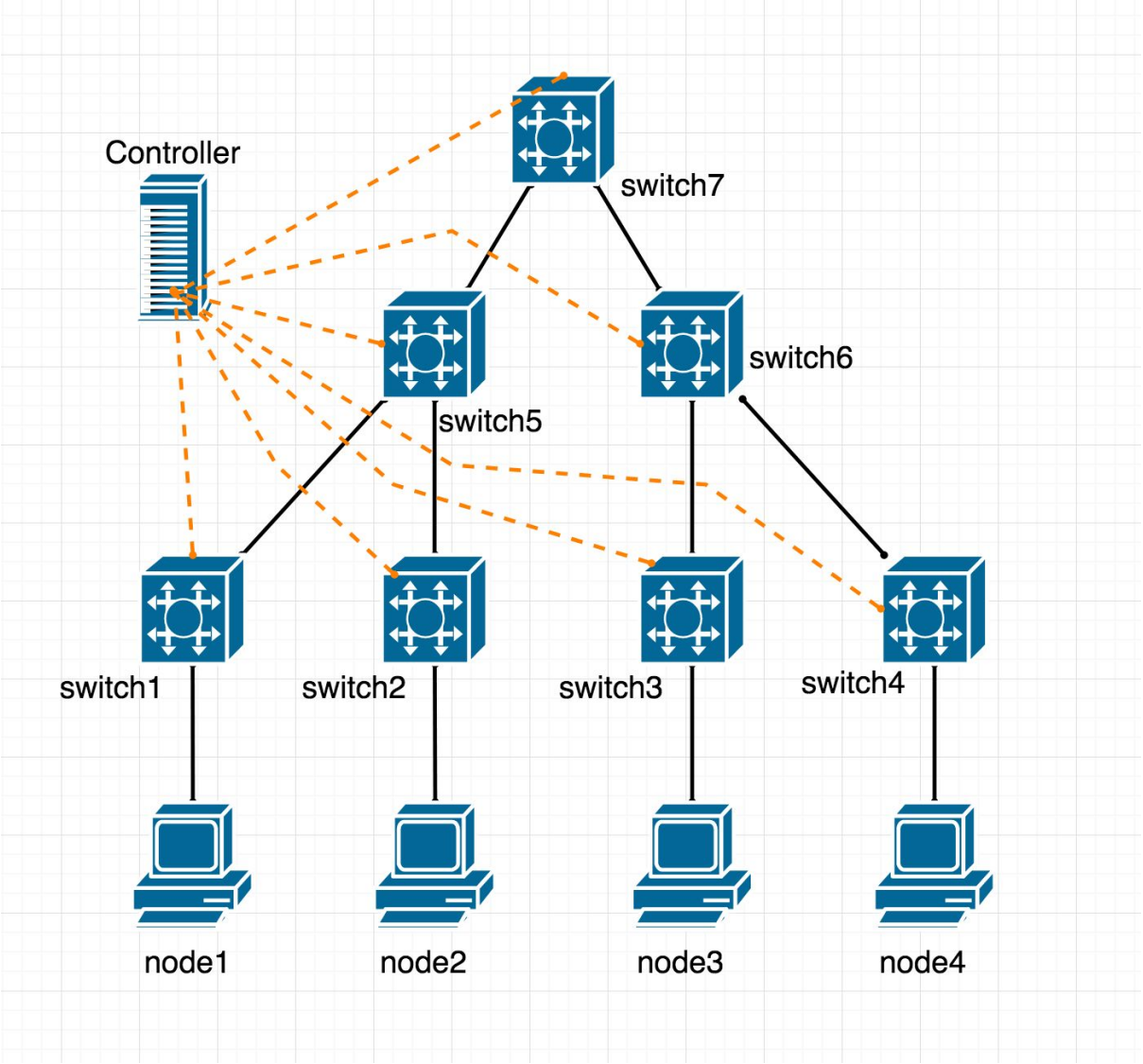

Figure 2: A tree topology consisting SDN switches and nodes connected to leaf switches

The required NS file for such a topology will be as follows:

```
### Start of the script

# Starting a simulator
set ns [new Simulator]
source tb_compat.tcl

# Defining nodes
```

```
set node1 [$ns node]
set node2 [$ns node]
set node3 [$ns node]
set node4 [$ns node]

# Defining Switches
set switch1 [$ns node]
tb-set-hardware $switch1 ofswitch
set switch2 [$ns node]
tb-set-hardware $switch2 ofswitch
set switch3 [$ns node]
tb-set-hardware $switch3 ofswitch
set switch4 [$ns node]
tb-set-hardware $switch4 ofswitch
set switch5 [$ns node]
tb-set-hardware $switch5 ofswitch
set switch6 [$ns node]
tb-set-hardware $switch6 ofswitch
set switch7 [$ns node]
tb-set-hardware $switch7 ofswitch

# Connecting Links between leaf switches and server nodes 1-4
set link1 [$ns duplex-link $switch1 $node1 1Gb 0ms DropTail]
set link2 [$ns duplex-link $switch2 $node2 1Gb 0ms DropTail]
set link3 [$ns duplex-link $switch3 $node3 1Gb 0ms DropTail]
set link4 [$ns duplex-link $switch4 $node4 1Gb 0ms DropTail]

# Connecting switches together
set link5 [$ns duplex-link $switch1 $switch5 1Gb 0ms DropTail]
set link6 [$ns duplex-link $switch2 $switch5 1Gb 0ms DropTail]
set link7 [$ns duplex-link $switch3 $switch6 1Gb 0ms DropTail]
set link8 [$ns duplex-link $switch4 $switch6 1Gb 0ms DropTail]
set link9 [$ns duplex-link $switch5 $switch7 1Gb 0ms DropTail]
set link10 [$ns duplex-link $switch6 $switch7 1Gb 0ms DropTail]

# [Optional] Define a node named “controller”
# and declare it as an SDN Controller
set controller [$ns node]
tb-set-hardware $controller ofcontrol

$ns rtproto Static
$ns run

### End of the script
```