

Part I: Configuring the Environment

1. Once you log in the mobile node, enter the folder geniclosure. You can see a list of scripts there for you to setup the handover experiment.
2. Run script “configInterfaces.sh” to bring up the WiMAX interface and the WiFi Interface, as shown as eth2 and wlan0 in Figure (1). The WiMAX is connected through a dongle. Thus, the WiMAX connection is shown as an Ethernet interface.

```
eth1      Link encap:Ethernet  HWaddr 00:03:1d:0c:d3:4b
          inet addr:10.1.0.1  Bcast:10.1.0.255  Mask:255.255.255.0
          inet6 addr: fe80::203:1dff:fe0c:d34b/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2525 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3496 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:221789 (221.7 KB)  TX bytes:295287 (295.2 KB)
          Interrupt:16 Memory:fa600000-fa620000

eth2      Link encap:Ethernet  HWaddr 00:1e:42:02:0c:b9
          inet addr:192.168.0.8  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::21e:42ff:fe02:cb9/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1018 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:29244 (29.2 KB)  TX bytes:1414 (1.4 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:11 errors:0 dropped:0 overruns:0 frame:0
          TX packets:11 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:104 (104.0 B)  TX bytes:104 (104.0 B)

wlan0     Link encap:Ethernet  HWaddr 00:15:6d:84:df:10
          UP BROADCAST MULTICAST  MTU:1500  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:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Figure (1) Bring up Interfaces

3. Run script “startOVS.sh” to bring up the OVS used for transparent handover, as shown in Figure (2)

```
root@node1:~/geniclosure# sudo sh startOVS.sh
Loading OVS modules
OVS: Creating database
2015-07-16T14:29:24Z|00001|vlog|INFO|opened log file /usr/local/var/log/openvswitch/ovsdb-server.log
OVS: Initializing OVS...
OVS: Starting OVS...
2015-07-16T14:29:24Z|00001|reconnect|INFO|unix:/usr/local/var/run/openvswitch/db.sock: connecting...
2015-07-16T14:29:24Z|00002|reconnect|INFO|unix:/usr/local/var/run/openvswitch/db.sock: connected
```

Figure (2) Bring up OVS

4. Run script “onesystem_setup.sh” to configure the OVS and connect the WiFi interface to our test AP. Note that we have disabled the network-manager on the machine. Thus we use iwconfig to build the WiFi connection in the script. After this, we can run “sudo ovs-vsctl show” to check the configuration of the OVS.

```
root@node1:~/geniclosure# sudo ovs-vsctl show
50477e69-5da2-4588-ad4a-31b32ef04465
    Bridge br_tap
        Controller "tcp:127.0.0.1:6653"
        fail_mode: secure
        Port br_tap
            Interface br_tap
                type: internal
        Port "wlan0"
            Interface "wlan0"
        Port wmaxtun
            Interface wmaxtun
                type: gre
                options: {remote_ip="130.127.38.133"}
```

Figure (3) OVS Configuration

As shown in Figure (3), the OVS has three ports. Port “br_tap” is the local interface of the OVS Bridge. It will be used as the default interface for all application traffic. Port “wlan0” connects to the WiFi interface. Port “wmaxtun” is a gre tunnel to eth5 of the handover root node (130.127.38.133) built upon the WiMAX interface. Basically, application packets received at port “br_tap” will be forwarded out through either port “wlan0” or port “wmaxtun”. Forwarding out through “wlan0”/“wmaxtun” means that your application is using WiFi/WiMAX.

5. Run script “initial_flows_wifi.sh” (“initial_flows_flow.sh”) to indicate that your application will use WiFi (WiMAX) in the beginning. This will insert corresponding flow entry into the OVS. Figure (4) shows the flow entries after running “initial_flows_wifi.sh”

```
root@node1:~/geniclosure# sudo ovs-ofctl dump-flows br_tap
NXST_FLOW reply (xid=0x4):
 cookie=0x0, duration=24.517s, table=0, n_packets=0, n_bytes=0, idle_age=24, priority=20000,in_port=LOCAL actions=output:2
 cookie=0x0, duration=24.520s, table=0, n_packets=4, n_bytes=276, idle_age=0, priority=20000,in_port=2 actions=LOCAL
```

Figure (4) Initial Flow Entries for WiFi in OVS

6. Run script “dhclient_tap.sh” to get an IP for br_tap and meanwhile, notify the root node which network is currently being used (The root node needs to know which network is currently being used so that it knows which way it should use to send packets back to you). After this, you can see that br_tap has an IP address of “192.168.4.8”, as shown in Figure (5).

```
root@node1:~/geniclosure# ifconfig
br_tap      Link encap:Ethernet  HWaddr 12:51:16:90:8f:ee
            inet addr:192.168.4.8  Bcast:192.168.4.255  Mask:255.255.255.0
            UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
            RX packets:209  errors:0  dropped:67  overruns:0  frame:0
            TX packets:34  errors:0  dropped:0  overruns:0  carrier:0
            collisions:0 txqueuelen:0
            RX bytes:14611 (14.6 KB)  TX bytes:11628 (11.6 KB)
```

Figure (5) IP of br_tap

7. Run script “add_default_route.sh” to set br_tap as the default interface for all application packets. The correct routing table setting is shown in Figure (6).

```
root@node1:~/geniclosure# route -n
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
0.0.0.0 192.168.4.128 0.0.0.0 UG 0 0 0 br_tap
10.1.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth1
130.127.38.133 192.168.0.1 255.255.255.255 UGH 0 0 0 eth2
192.168.0.0 0.0.0.0 255.255.255.0 U 0 0 0 eth2
192.168.4.0 0.0.0.0 255.255.255.0 U 0 0 0 br_tap
```

Figure (6) IP Routing Table

8. Now you are all set for the configuration. As mentioned above, we set WiFi as the networking being used in the beginning. Then, to switch to WiMAX network, you only need to run the script “switch_from_wifi_to_wmax.sh”. After this, to switch back to WiFi again, you can run script “switch_from_wmax_to_wifi.sh”. Figure (7) illustrates two transparent handover. First, at icmp_req=17, we switch to WiMAX. After this, we see that the RTT

increases to around 100ms, which represents the WiMAX. Second, at icmp_req=27, we switch back to WiFi. After this, the RTT falls to around 25ms, which represents the WiFi.

```
64 bytes from 8.8.8.8: icmp_req=13 ttl=41 time=25.5 ms
64 bytes from 8.8.8.8: icmp_req=14 ttl=41 time=25.8 ms
64 bytes from 8.8.8.8: icmp_req=15 ttl=41 time=25.6 ms
64 bytes from 8.8.8.8: icmp_req=16 ttl=41 time=25.6 ms
64 bytes from 8.8.8.8: icmp_req=17 ttl=41 time=25.8 ms
64 bytes from 8.8.8.8: icmp_req=19 ttl=41 time=119 ms
64 bytes from 8.8.8.8: icmp_req=20 ttl=41 time=234 ms
64 bytes from 8.8.8.8: icmp_req=21 ttl=41 time=106 ms
64 bytes from 8.8.8.8: icmp_req=22 ttl=41 time=91.2 ms
64 bytes from 8.8.8.8: icmp_req=23 ttl=41 time=94.7 ms
64 bytes from 8.8.8.8: icmp_req=24 ttl=41 time=108 ms
64 bytes from 8.8.8.8: icmp_req=25 ttl=41 time=101 ms
64 bytes from 8.8.8.8: icmp_req=26 ttl=41 time=107 ms
64 bytes from 8.8.8.8: icmp_req=27 ttl=41 time=99.7 ms
64 bytes from 8.8.8.8: icmp_req=28 ttl=41 time=27.1 ms
64 bytes from 8.8.8.8: icmp_req=29 ttl=41 time=25.8 ms
64 bytes from 8.8.8.8: icmp_req=30 ttl=41 time=26.2 ms
64 bytes from 8.8.8.8: icmp_req=31 ttl=41 time=25.5 ms
64 bytes from 8.8.8.8: icmp_req=32 ttl=41 time=26.3 ms
```

Figure (7) Switch to WiMax and then back to WiFi