Orbit Management Framework (OMF) Demonstration

Thierry Rakotoarivelo
Max Ott
OMF - User View

- Experiment Description
- Results

- Deploy & Configure

- Control
- Experimental Platform(s)

- Measure
Simple Multi-path Experiment

Overview
- Ad-Hoc Mesh Network + WiFi 802.11g
- Ad-Hoc routing with Multi-Path capability (OLSR from PhD NICTA)
- Constant traffic from one node to another

User at NICTA → OMF on Servers at WinLab → OMF on Wireless Testbed at NICTA
Simple Multi-path Experiment

Sender (n1) [1,4]
- Routing(OLSR)
- Traffic Source (OTG)

Relay (n2) [1,6]
- Routing(OLSR)

Relay (n3) [1,8]
- Routing(OLSR)

Relay (n4) [4,4]
- Routing(OLSR)

Receiver (n5) [4,8]
- Routing(OLSR)
- Traffic Sink (OTR)
- Pkt Capture (OTL)

MAC Addr: 00:60:b3:ac:2c:2c
MAC Addr: 00:60:b3:ac:2c:0d
Experiment Script

1. SETUP
   - Describe network Topology
   - Configure & Associate applications to nodes
   - Configure network Interfaces

2. OPERATION
   - Implement the Topology
   - List of Actions to perform
Experiment Script

- SETUP - Describe network Topology
  - See example script
  - Other available methods:

```ruby
defTopology('myTopo') { |t|
  baseTopo = Topology['system:topo:active']
  for count in 1..4
    # Draw a random node
    aNode = baseTopo.getUniqueRandomNode
    # Add it to this topology
    t.addNode(aNode)
  end
}

defTopology('aSubTopology') { |t|
  parentTopo = Topology['myTopo']
  aNode1 = parentTopo.getNodeByLabel("n_1")
  aNode2 = parentTopo.getNodeByLabel("n_4")
  t.addNode(aNode1)
  t.addNode(aNode2)
}

defTopology('myTopo') { |t|
  baseTopo = Topology['system:topo:active']
  someNodes = baseTopo.select( :method => :random,
                               :number => 4,
                               :name => "n_%i%",
                               :features => { :wifi => "atheros",
                                              :bt => "false",
                                              :mem => "1Gb",
                                              :channel => "6"})
  t.addNodes(someNodes)
  t.addLink("n_1","n_2", {:rate =>54, :per =>0.1, :asymmetric => true })
  t.addLink("n_2","n_3", {:rate =>12, :per =>0.2, :asymmetric => true })
```
**Experiment Script**

- **SETUP** - Configure and Associate applications to nodes
  - See example script - Other available methods:

```ruby
defApplication('app:myapp', 'myapp') { |a|
  a.version(1, 0, 0)
  a.shortDescription = "A Programmable traffic generator"
  a.defProperty('pkt-loss-rate', 'Packet Loss Rate in%')
  a.binaryRepository = "~/home/Alice/myArchive.tar"
  a.path = "~/usr/bin/traffic-gen"
}
defPrototype("proto:myApp") { |p|
  p.name = "MyApplication"
  p.description = "A simple network application"
  p.defProperty('lossRate', 'Packet Loss Rate in%, '0')
  p.addApplication(:myapp, "app:myapp") { |a|
    a.bindProperty('pkt-loss-rate', 'lossRate')
  }
}
Experiment.defProperty('myRate', 400, 'Rate in kBps')
defGroup("theSenderGroup", "anExistingTopology") { |node|
  node.prototype("proto:myApp", 'destinationHost'=>'192.168.255.255',
                 'packetSize' => 12,
                 'rate' => prop.myRate,
                 'protocol' => 'udp')
}
```
Experiment Script

• SETUP - Configure Network Interface
  - See example script

• OPERATION - Implement the Topology
  - See example script
  - Other available methods:

```ruby
allGroups.net.w0.enforce_link = {:topology => "mainTopology", :method => 'mackill'}
```

  or

```ruby
allGroups.net.w0.enforce_link = {:topology => "mainTopology", :method => 'ebtable'}
```

- Extendable, support for new technology = a new ‘method’
- Warn user if the Topology is not feasible (next OMF release)
Experiment Script

- OPERATION - List of Actions to perform
  - Available methods:

```python
whenAllUp { ... } # Execute actions when all nodes are UP
whenAllInstalled { ... } # Execute actions when all Apps on all nodes are ready to run
wait 30
info "Some messages..."
prop.myRate = 900
allGroups.exec("/usr/bin/someCommand -someParameter 123")
allGroups.startApplications
allGroups.stopApplications
Experiment.Done

## Same as above but for a specific group:

group("myGroup").startApplications

Etc...
```
Running the Experiment...

Sender (n1) [1,4]
- Routing(OLSR)
- Traffic Source (OTG)

Relay (n2) [1,6]
- Routing(OLSR)

Relay (n3) [1,8]
- Routing(OLSR)

Receiver (n5) [4,8]
- Routing(OLSR)
- Traffic Sink (OTR)
- Pkt Capture (OTL)

Relay (n4) [4,4]
- Routing(OLSR)

MAC Addr: 00:60:b3:ac:2c:2c
MAC Addr: 00:60:b3:ac:2c:0d

After 20sec...
The OMF Measurement Library (OML) - 2 available methods
A. Add measurement hooks in your App code + Compile App with OML library
B. Use built-in packet tap OTL, which is based on existing libpcap (….)
Define filters

```c
// - 1
omlc_init(name, &argc, argv, o_log);

// - 2
static OmlMPDef oml_def[] = {
    {"ts", OML_DOUBLE_VALUE},
    {"pkt_length", OML_LONG_VALUE},
    {"dst_host", OML_STRING_PTR_VALUE},
    {"dst_port", OML_LONG_VALUE},
    {NULL, (OmlValueT)0},
};

// - 3
static OmlMP* oml_mp;
oml_mp = omlc_add_mp("udp_out", oml_def);

// - 4 - before starting your app
omlc_start();

// - 5 - within your app’s execution
omlc_process(oml_mp, v);
```

// Libpcap specific filters
// (i.e. tcpdump 'expressions')
dst host 192.168.0.1
Each node: OML Measurement Point(s) & Filter(s)

Server: OML Collection & Storage Services

defApplication('app:myapp', 'myapp') {
  ...
  a.defMeasurement("mpl") {
    m.defMetric('pkt_seqno', 'long')
    m.defMetric('pkt_size', 'long')
  }
}

defPrototype("proto:myApp") {
  p.addApplication(:myApp, "app:myapp") {
    ...
    a.measure('mpl', :interval => 1.sec) {
      m.add('pkt_seqno')
      m.add('pkt_size', :filter => 'builtin:avg')
    }
  }
}
Thank you

Max.Ott@nicta.com.au
Thierry.Rakotoarivelo@nicta.com.au