Describing Experiments in Orbit

Max Ott

NICTA

Vision

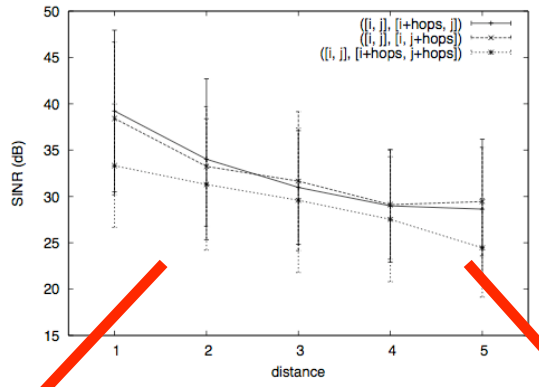
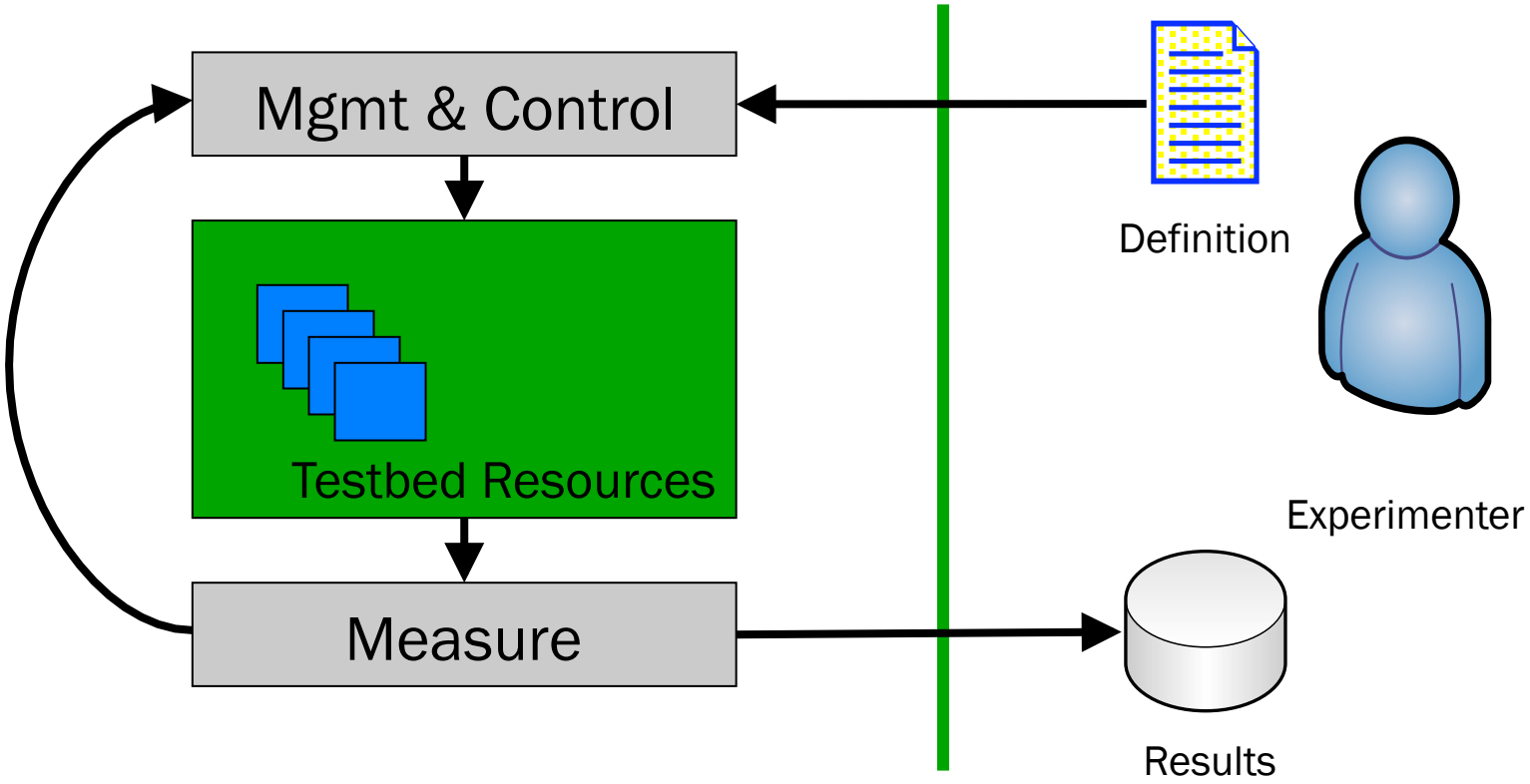


Figure 8: The average link quality measurement in SINR between a pair of nodes in ORBIT.

Experiment Description

Results

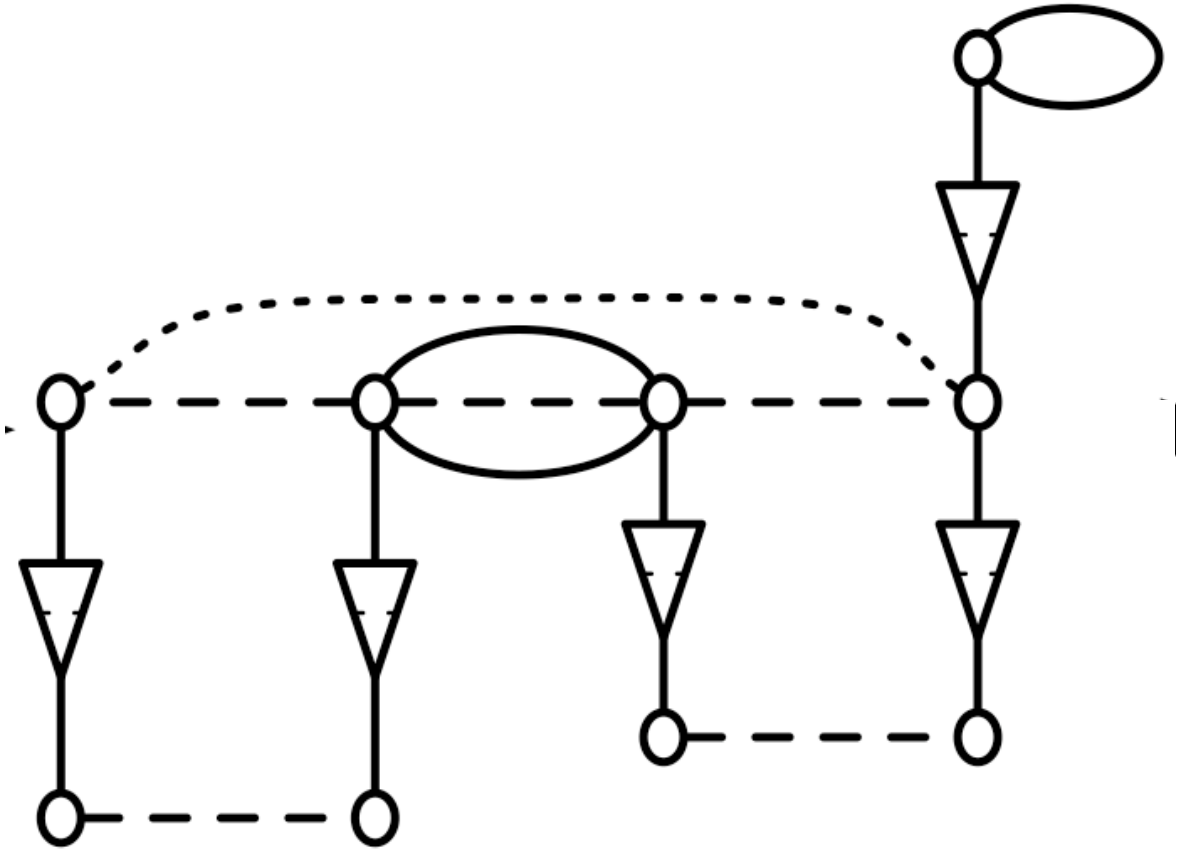
OMF – Experimenter's View



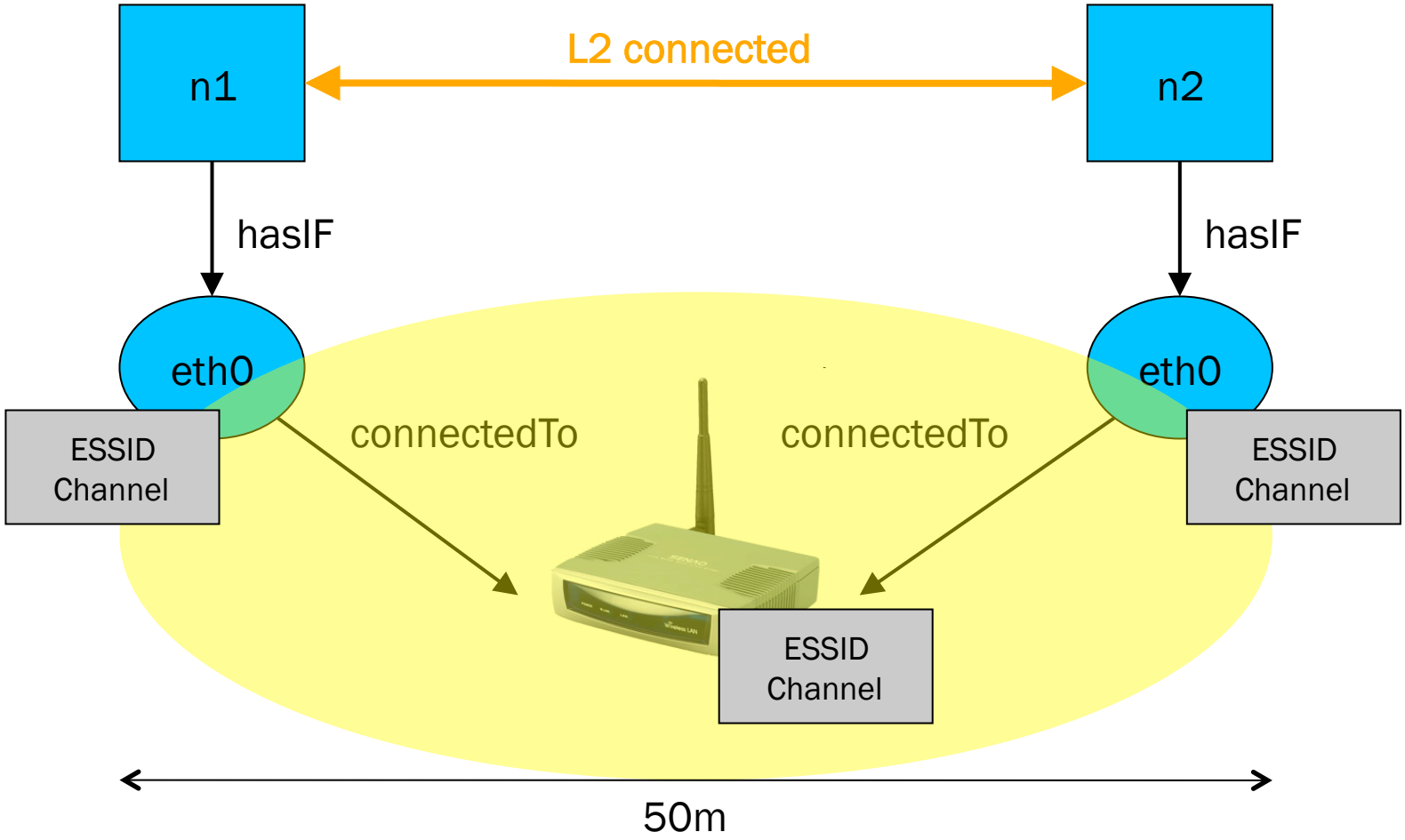
What do we need to describe experiments?

- A Resource Description Model
  - What do we need for the experiment
  - What are their properties and constraints
  - What do we want to measure (and how it ties back to resources)
- A Programming Model
  - How do we define the orchestration of the experiment
  - How do we deal with errors

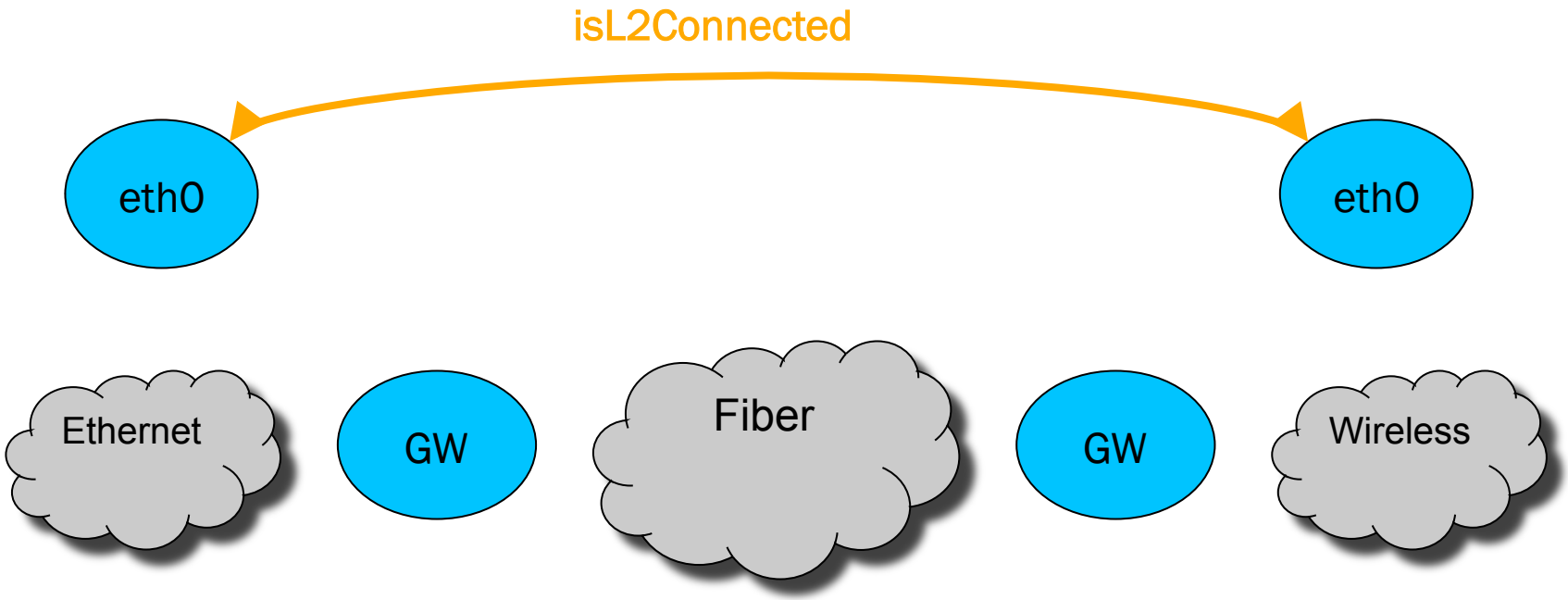
G.800 – ITU Recommendation to describe networks



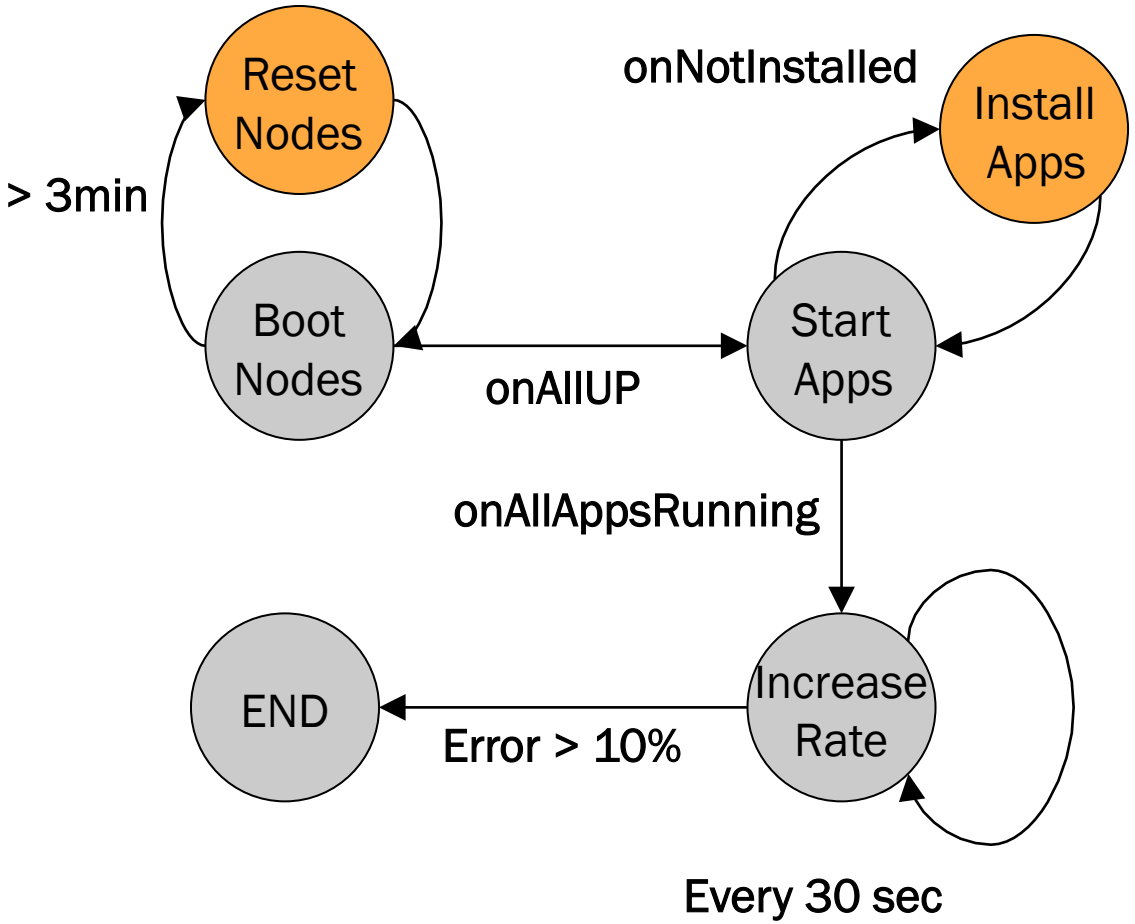
Relationships + Conditions/Rules



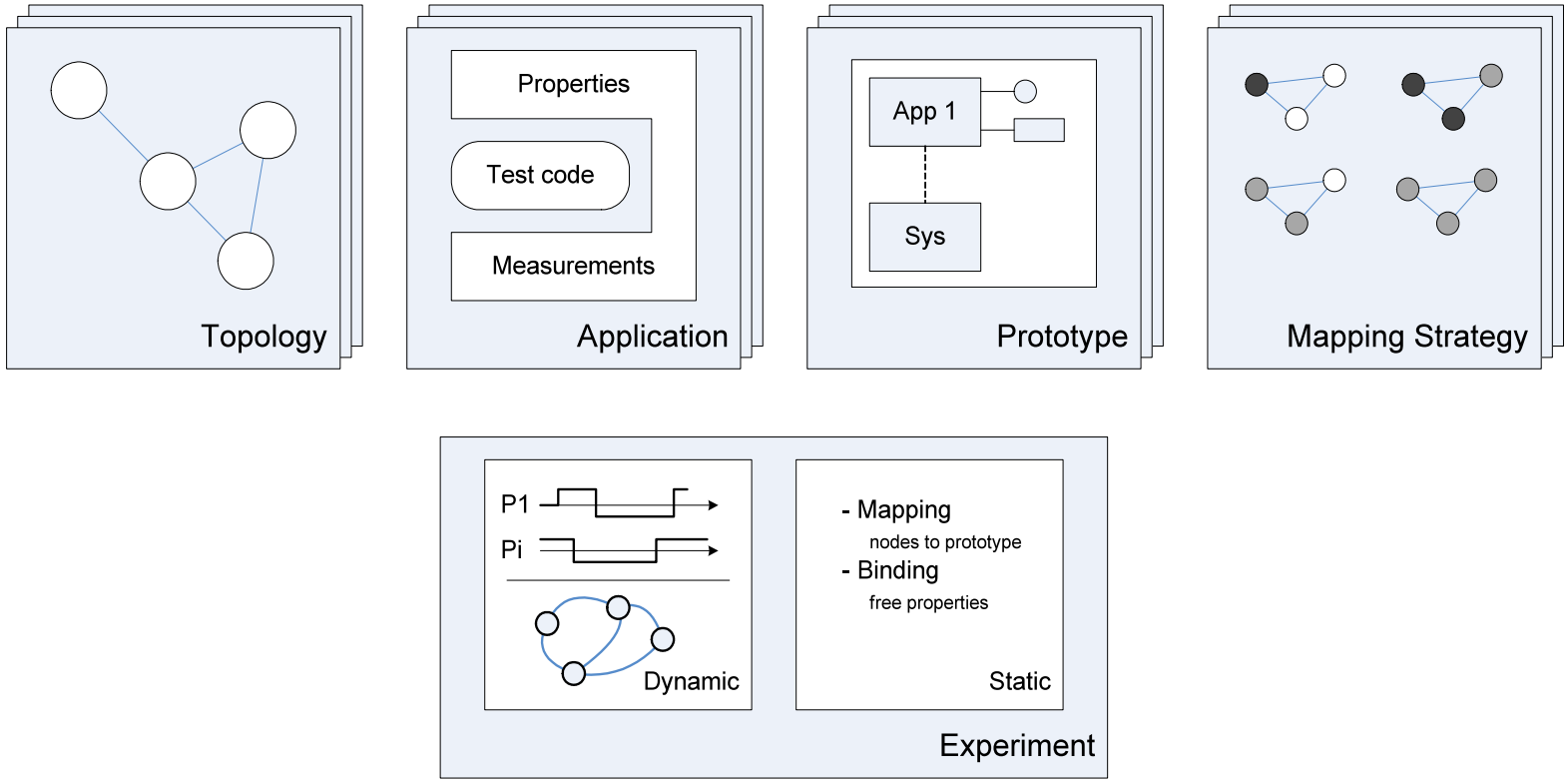
Hiding Details



Programming Model



So how do we really do it?



Orbit's Hello World Experiment

```
defNodes('sender', [1,2]) {|node|
  node.prototype("test:proto:sender", {
    'destinationHost' => '192.168.1.4',
    'packetSize' => 1024,
    'rate' => 100,
    'protocol' => 'udp'
  })
  node.net.w0.mode = "master"
}

defNodes('receiver', [1,4]) {|node|
  node.prototype("test:proto:receiver" , {
    'protocol' => 'udp'
  })
  node.net.w0.mode = "managed"
}
```

```
allNodes.net.w0 { |w|
  w.type = 'b'
  w.essid = "helloworld"
  w.ip = "%192.168.%x.%y"
}

#
# Now, start the application
#
whenAllInstalled() {|node|
  wait 30 # time to associate
  allNodes.startApplications
  wait 30
  Experiment.done
}
```

DescribingResources

```
defGroup('sender‘ , [1,2]) {|g|
  g.prototype("test:proto:sender", {
    'destinationHost' => '192.168.1.4',
    'packetSize' => 1024,
    'rate' => 100,
    'protocol' => 'udp'
  })
  g.net.w0.mode = "master"
}
```

Defining Experiment Timeline

```
whenAllInstalled() {
  wait 30 # time to associate
  allNodes.startApplications
  wait 30
  Experiment.done
}
```

Defining Prototypes

```
defPrototype("test:proto:sender") { |p|  p.name = "Sender"
    p.defProperty('protocol', 'Protocol to use', 'udp')
    ...
    p.addApplication(:otg, "test:app:otg") {|otg|
        otg.bindProperty('protocol')
        ...
        otg.measure('senderport') {|m|
            m.add('pkt_size', Filter::SUM)
            ...
        }
    }
}
```

Defining Applications

```
defApplication('test:app:otg', 'otg') { |a|
    a.shortDescription = "Programmable traffic ...”
    ...
    a.defProperty("rate", "Data rate of the flow [kbps]",
        {:dynamic => true, :type => :integer})
    ...
    a.defMeasurement("senderport") { |m|
        m.defMetric('pkt_seqno', 'long', 'Packet ...')
    }
    a.path = "/usr/bin/otg”
}
```

Experiment Properties …

```
defProperty('packetSize', 256, 'Size of packets ...')
defProperty('maxPktSize', 1280, 'Max. size of packets ...')

defNodes([1, 1], 'sender') {|node|
  node.prototype("test:proto:sender",
    'destinationHost' => '192.168.8.2',
    'packetSize' => prop.packetSize,
    'rate' => prop.rate)
  )
}
```

… Experiment Properties …

```
whenAllInstalled() {|nodes|
  prop.packetSize = 256
  ...
  prop.packetSize = 1024
  ...
  wait 30
  nodes.stopApplications
  ...
}
```

Summary

- Resource Description Model
  - Based on standard – G.800
  - Formulated through OWL/RDF
  - Allows for reasoning and constraint-based resource mapping
  - Includes measurements
- Programming Model
  - State-machine
    - Allows for easy separation of experiment & error handling
  - Events created from measurements

OIDL: Describing Experiments in Orbit

Max Ott

NICTA

Australian Government

Department of Communications, Information Technology and the Arts

Australian Research Council

NICTA Members

NICTA Partners