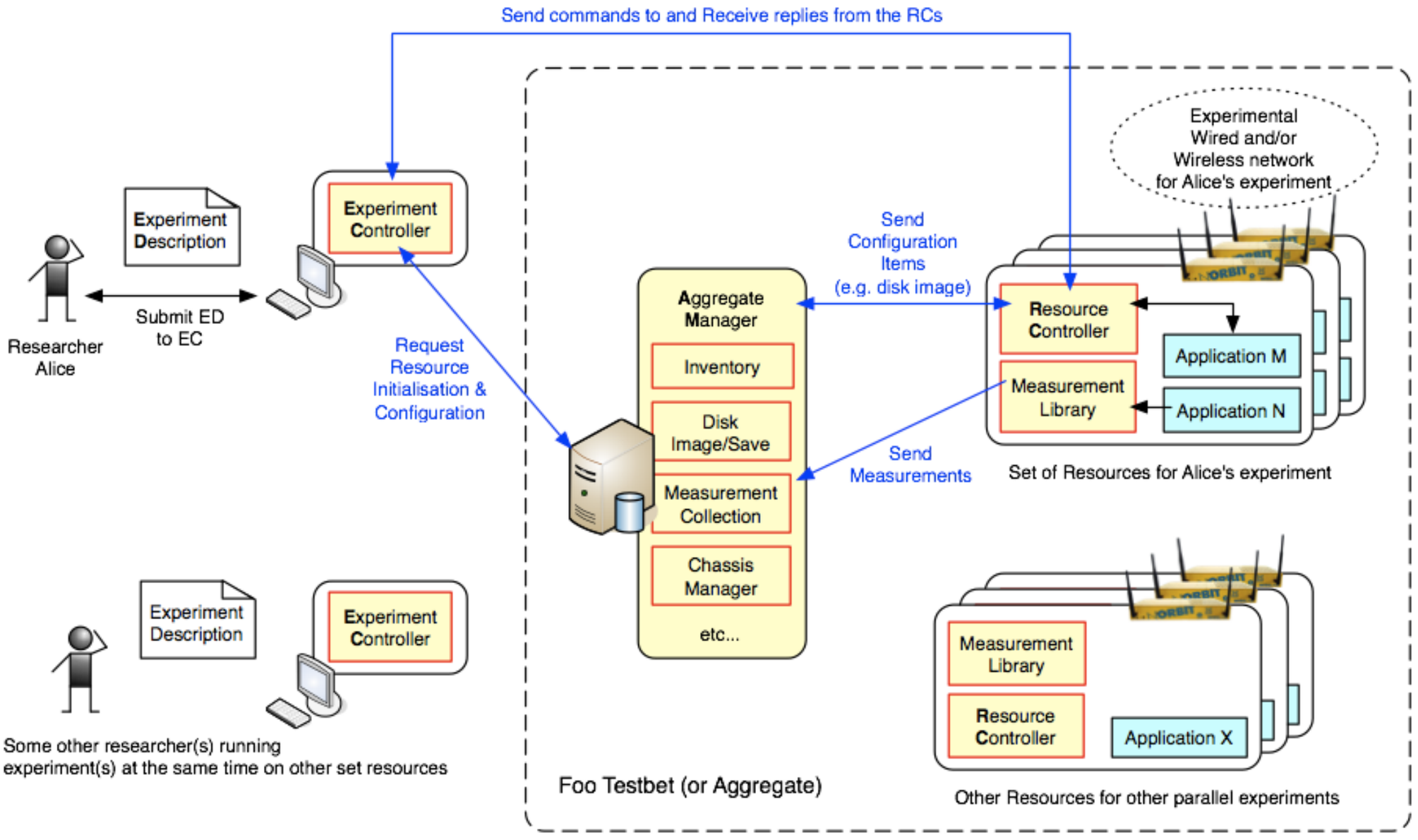
WiMax Experiments using OMF/OML

Hamed Soroush,
Ph.D Candidate,
University of Massachusetts, Amherst

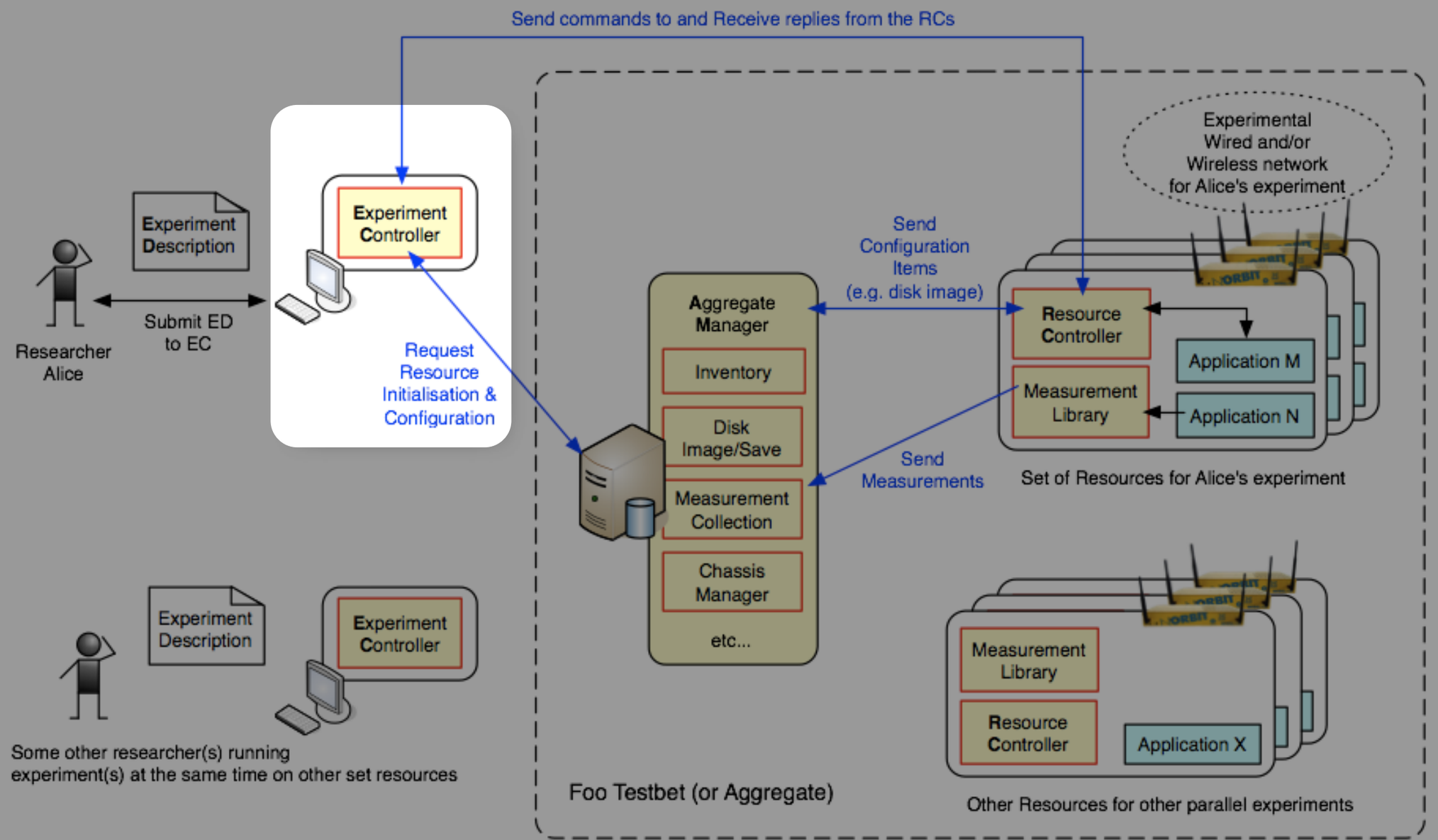
Raytheon
BBN Technologies

geni

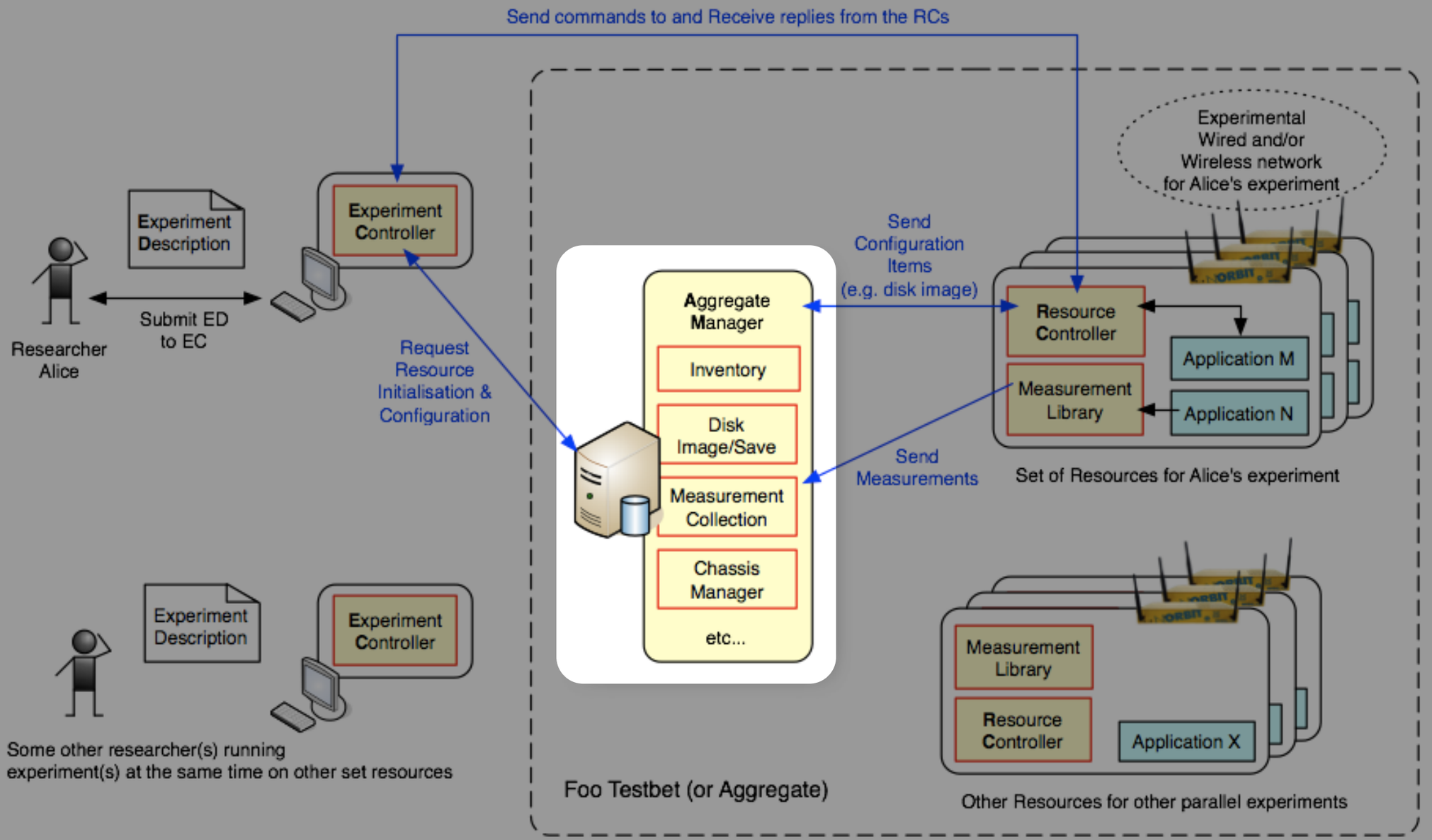
OMF System


Send commands to and Receive replies from the RCs
Experimental
Wired and/or
Wireless network
for Alice's experiment
Experiment
Description
Experiment
Controller
Researcher
Alice
Submit ED
to EC
Request
Resource
Initialisation &
Configuration
Send
Configuration
Items
(e.g. disk image)
Aggregate
Manager
Inventory
Disk
Image/Save
Measurement
Collection
Chassis
Manager
etc...
Resource
Controller
Measurement
Library
Application M
Application N
Send
Measurements
Set of Resources for Alice's experiment
Experiment
Description
Experiment
Controller
Some other researcher(s) running
experiment(s) at the same time on other set resources
Measurement
Library
Resource
Controller
Application X
Foo Testbet (or Aggregate)
Other Resources for other parallel experiments

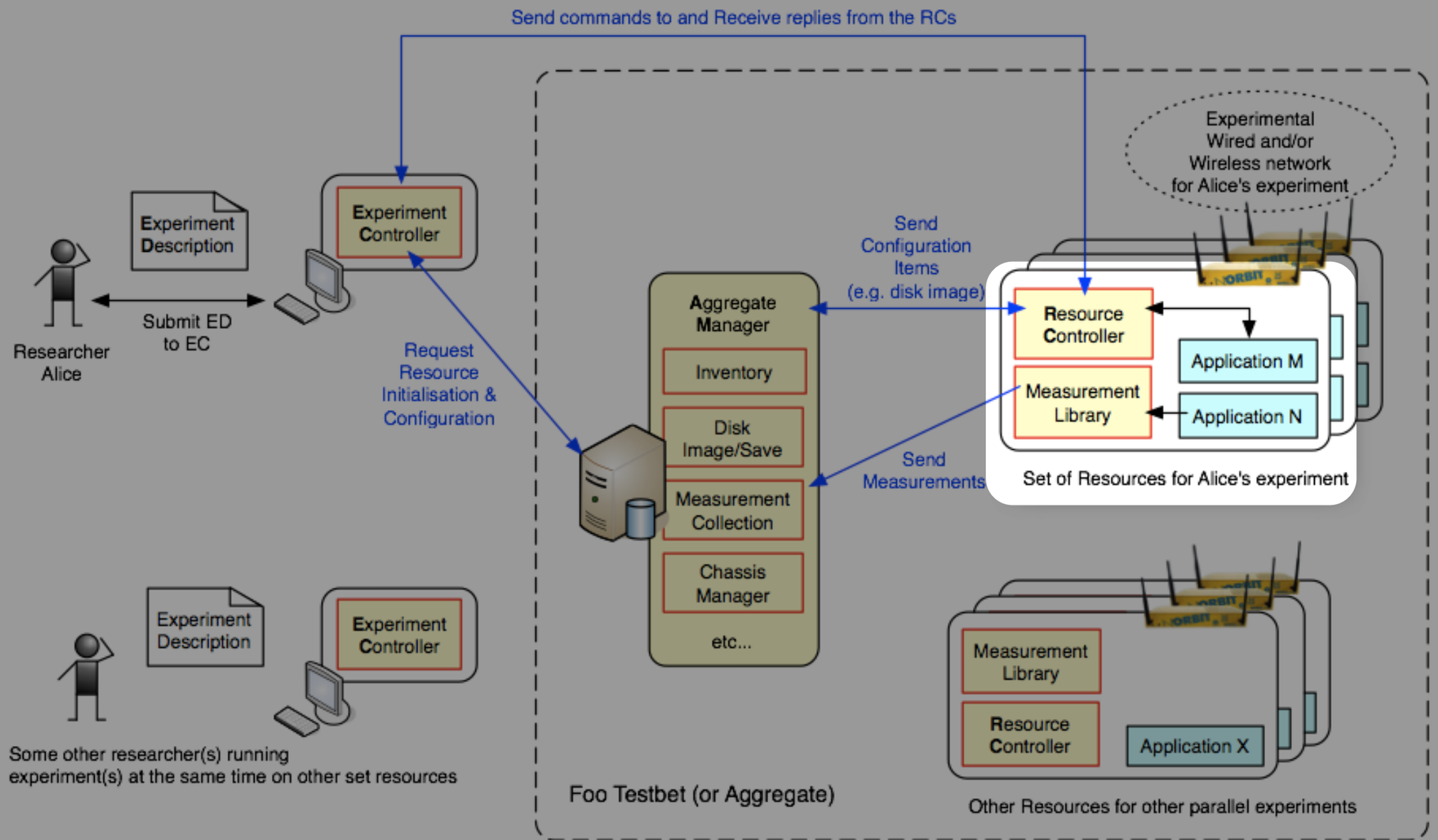
OMF System


Send commands to and Receive replies from the RCs
Experiment Description
Researcher Alice
Submit ED to EC
Experiment Controller
Request Resource Initialisation & Configuration
Experimental Wired and/or Wireless network for Alice's experiment
Send Configuration Items (e.g. disk image)
Aggregate Manager
Inventory
Disk Image/Save
Measurement Collection
Chassis Manager
etc...
Resource Controller
Application M
Measurement Library
Application N
Send Measurements
Set of Resources for Alice's experiment
Experiment Description
Experiment Controller
Some other researcher(s) running experiment(s) at the same time on other set resources
Measurement Library
Resource Controller
Application X
Other Resources for other parallel experiments
Foo Testbet (or Aggregate)

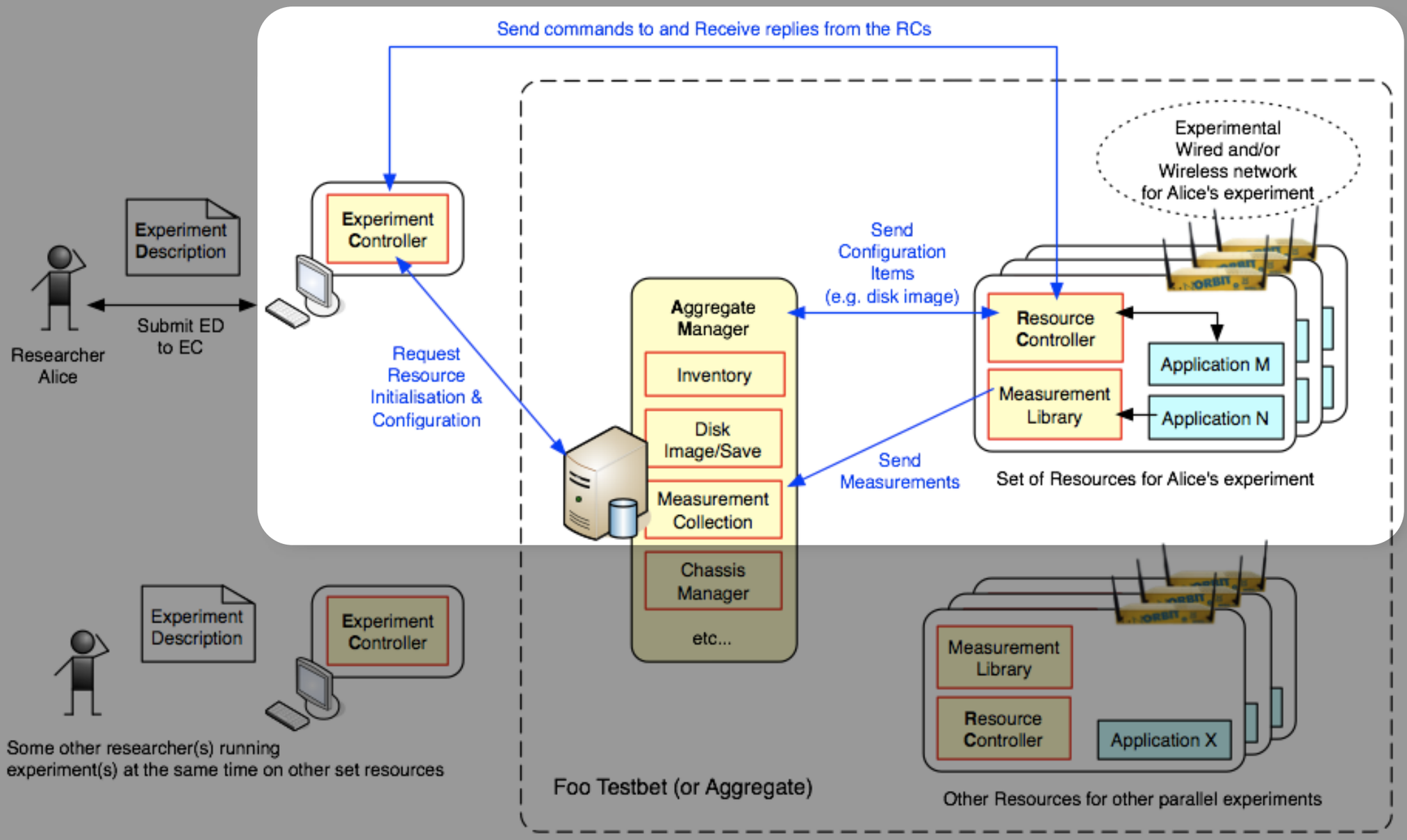
OMF System


Send commands to and Receive replies from the RCs
Experimental Wired and/or Wireless network for Alice's experiment
Experiment Description
Experiment Controller
Researcher Alice
Submit ED to EC
Request Resource Initialisation & Configuration
Send Configuration Items (e.g. disk image)
Aggregate Manager
Inventory
Disk Image/Save
Measurement Collection
Chassis Manager
etc...
Resource Controller
Application M
Application N
Measurement Library
Send Measurements
Set of Resources for Alice's experiment
Experiment Description
Experiment Controller
Some other researcher(s) running experiment(s) at the same time on other set resources
Measurement Library
Resource Controller
Application X
Other Resources for other parallel experiments
Foo Testbet (or Aggregate)

OMF System


Send commands to and Receive replies from the RCs
Experiment Description
Experiment Controller
Researcher Alice
Submit ED to EC
Request Resource Initialisation & Configuration
Send Configuration Items (e.g. disk image)
Aggregate Manager
Inventory
Disk Image/Save
Measurement Collection
Chassis Manager
etc...
Send Measurements
Experimental Wired and/or Wireless network for Alice's experiment
Resource Controller
Measurement Library
Application M
Application N
Set of Resources for Alice's experiment
Experiment Description
Experiment Controller
Some other researcher(s) running experiment(s) at the same time on other set resources
Measurement Library
Resource Controller
Application X
Other Resources for other parallel experiments
Foo Testbet (or Aggregate)

OMF System


Send commands to and Receive replies from the RCs
Experiment Description
Researcher Alice
Submit ED to EC
Experiment Controller
Request Resource Initialisation & Configuration
Aggregate Manager
Inventory
Disk Image/Save
Measurement Collection
Chassis Manager
etc...
Send Configuration Items (e.g. disk image)
Send Measurements
Experimental Wired and/or Wireless network for Alice's experiment
Resource Controller
Application M
Measurement Library
Application N
Set of Resources for Alice's experiment
Experiment Description
Experiment Controller
Some other researcher(s) running experiment(s) at the same time on other set resources
Measurement Library
Resource Controller
Application X
Other Resources for other parallel experiments
Foo Testbet (or Aggregate)

```
# Experiment Definition File.
# Performing Downlink Throughput Measurements

defProperty('res1', "omf.bbn.node1", "ID of a node")
```

```
# Adding support for GPS logging
defApplication('gpslogger','gps') do |app|
  app.path="gpslogger"
  app.appPackage = "gpslogger.tar"
  app.version(1,0,0)
  app.shortDescription = "GPS Logger"
  app.description = "Retrieves GPS fix data from gpsd
                     and feeds it into OML."
  app.defMeasurement('gps_data') do |mp|
    mp.defMetric('lat', :float)
    mp.defMetric('lon', :float)
    mp.defMetric('ele', :float)
    mp.defMetric('fix', :string)
    mp.defMetric('time', :string)
  end
end
```

```
# Adding support for logging link status
defApplication('linkstatus_app','linkstatus') do |app|
  app.path="wiwrapper.rb"
  app.appPackage = "wiwrapper.tar"
  app.version(1,0,0)
  app.shortDescription = "Wrapper around wimaxcu link status"
  app.description = <<TEXT
      This is a wrapper around wimaxcu link status command
TEXT
```

```
# Experiment Definition File.
# Performing Downlink Throughput Measurements

defProperty('res1', "omf.bbn.node1", "ID of a node")


# Adding support for GPS logging
defApplication('gpslogger','gps') do |app|
  app.path="gpslogger"
  app.appPackage = "gpslogger.tar"
  app.version(1,0,0)
  app.shortDescription = "GPS Logger"
  app.description = "Retrieves GPS fix data from gpsd
                     and feeds it into OML."
  app.defMeasurement('gps_data') do |mp|
    mp.defMetric('lat', :float)
    mp.defMetric('lon', :float)
    mp.defMetric('ele', :float)
    mp.defMetric('fix', :string)
    mp.defMetric('time', :string)
  end
end


# Adding support for logging link status
defApplication('linkstatus_app','linkstatus') do |app|
  app.path="wiwrapper.rb"
  app.appPackage = "wiwrapper.tar"
  app.version(1,0,0)
  app.shortDescription = "Wrapper around wimaxcu link status"
  app.description = <<TEXT
      This is a wrapper around wimaxcu link status command
TEXT
```

```
    mp.defMetric('time', :string)
  end
end


# Adding support for logging link status
defApplication('linkstatus_app','linkstatus') do |app|
  app.path="wiwrapper.rb"
  app.appPackage = "wiwrapper.tar"
  app.version(1,0,0)
  app.shortDescription = "Wrapper around wimaxcu link status"
  app.description = <<TEXT
      This is a wrapper around wimaxcu link status command
TEXT
  app.defProperty('sampling','Sampling interval in sec', 's',
  {
    :type => :integer,
    :dynamic => false
  })
  app.defMeasurement('wimaxstat') do |m|
    m.defMetric('CenterFrequency', :long)
    m.defMetric('RSSI', :long)
    m.defMetric('CINR', :long)
    m.defMetric('TXPWR', :long)
  end
end


# Defining a group with one node, that utilizes above applications
defGroup('Actor', property.res1) { |node|
  node.prototype("test:proto:iperftcpreceiver",
    { 'report_interval' => 1, 'time' => 40})
  node.addApplication("gpslogger") { |app|
    app.measure('gps_data', :interval => 3)
```

```
    mp.defMetric('time', :string)
  end
end
```

```
# Adding support for logging link status
defApplication('linkstatus_app','linkstatus') do |app|
  app.path="wiwrapper.rb"
  app.appPackage = "wiwrapper.tar"
  app.version(1,0,0)
  app.shortDescription = "Wrapper around wimaxcu link status"
  app.description = <<TEXT
      This is a wrapper around wimaxcu link status command
TEXT
  app.defProperty('sampling','Sampling interval in sec', 's',
  {
    :type => :integer,
    :dynamic => false
  })
  app.defMeasurement('wimaxstat') do |m|
    m.defMetric('CenterFrequency', :long)
    m.defMetric('RSSI', :long)
    m.defMetric('CINR', :long)
    m.defMetric('TXPWR', :long)
  end
end
```

```
# Defining a group with one node, that utilizes above applications
defGroup('Actor', property.res1) { |node|
  node.prototype("test:proto:iperftcpreceiver",
    { 'report_interval' => 1, 'time' => 40})
  node.addApplication("gpslogger") { |app|
    app.measure('gps_data', :interval => 3)
```

```
    mp.defMetric('time', :string)
  end
end


# Adding support for logging link status
defApplication('linkstatus_app','linkstatus') do |app|
  app.path="wiwrapper.rb"
  app.appPackage = "wiwrapper.tar"
  app.version(1,0,0)
  app.shortDescription = "Wrapper around wimaxcu link status"
  app.description = <<TEXT
      This is a wrapper around wimaxcu link status command
TEXT
  app.defProperty('sampling','Sampling interval in sec', 's',
  {
    :type => :integer,
    :dynamic => false
  })
  app.defMeasurement('wimaxstat') do |m|
    m.defMetric('CenterFrequency', :long)
    m.defMetric('RSSI', :long)
    m.defMetric('CINR', :long)
    m.defMetric('TXPWR', :long)
  end
end


# Defining a group with one node, that utilizes above applications
defGroup('Actor', property.res1) { |node|
  node.prototype("test:proto:iperftcpreceiver",
    { 'report_interval' => 1, 'time' => 40})
  node.addApplication("gpslogger") { |app|
    app.measure('gps_data', :interval => 3)
```

```
  app.description = <<TEXT
      This is a wrapper around wimaxcu link status command
TEXT
  app.defProperty('sampling','Sampling interval in sec', 's',
  {
    :type => :integer,
    :dynamic => false
  })
  app.defMeasurement('wimaxstat') do |m|
    m.defMetric('CenterFrequency', :long)
    m.defMetric('RSSI', :long)
    m.defMetric('CINR', :long)
    m.defMetric('TXPWR', :long)
  end
end


# Defining a group with one node, that utilizes above applications
defGroup('Actor', property.res1) { |node|
  node.prototype("test:proto:iperftcpreceiver",
    { 'report_interval' => 1, 'time' => 40})
  node.addApplication("gpslogger") { |app|
    app.measure('gps_data', :interval => 3)
  }
  node.addApplication("linkstatus_app") { |app|
    app.setProperty('sampling', 5)
    app.measure('wimaxstat')
  }
  node.net.x0.profile = '51'
}


# Start up the experiment when all nodes are ready
onEvent(:ALL_UP_AND_INSTALLED) do |event|
```

```
  app.description = <<TEXT
      This is a wrapper around wimaxcu link status command
TEXT
  app.defProperty('sampling','Sampling interval in sec', 's',
  {
    :type => :integer,
    :dynamic => false
  })
  app.defMeasurement('wimaxstat') do |m|
    m.defMetric('CenterFrequency', :long)
    m.defMetric('RSSI', :long)
    m.defMetric('CINR', :long)
    m.defMetric('TXPWR', :long)
  end
end
```

```
# Defining a group with one node, that utilizes above applications
defGroup('Actor', property.res1) { |node|
  node.prototype("test:proto:iperftcpreceiver",
    { 'report_interval' => 1, 'time' => 40})
  node.addApplication("gpslogger") { |app|
    app.measure('gps_data', :interval => 3)
  }
  node.addApplication("linkstatus_app") { |app|
    app.setProperty('sampling', 5)
    app.measure('wimaxstat')
  }
  node.net.x0.profile = '51'
}
```

```
# Start up the experiment when all nodes are ready
onEvent(:ALL_UP_AND_INSTALLED) do |event|
```

```
# Start up the experiment when all nodes are ready
onEvent(:ALL_UP_AND_INSTALLED) do |event|
  wait 5
  info "Press enter to continue"
  STDIN.gets
  group('Actor').startApplications
  wait 50
  group('Actor').stopApplications
  info "Experiment is done. Copying measurements to /home/geni/experiment_logs"
  info "Enter prefix: "
  prefix=STDIN.gets
  prefix.strip!
  FileUtils.cp "/tmp/#{Experiment.ID}.sq3" ,
               "/home/geni/experiment_logs/#{prefix}_#{Experiment.ID}.sq3"
  info "files copied."
  Experiment.done
end
```

```
# Adding support for visualization
addTab(:defaults)
  addTab(:graph2) do |tab|
  opts = { :postfix => %{This graph shows the throughput.},
           :updateEvery => 1
  }
  tab.addGraph("Throughput", opts) do |g|
    data = []
    mp = ms('iperf_TCP_Info')
    mp.project(:oml_ts_server, :Bandwidth).each do |sample|
      time, bw = sample.tuple
      data << [time, bw]
    end
    g.addLine(data, :label => "Bandwidth")
  end
end
```

```
# Start up the experiment when all nodes are ready
onEvent(:ALL_UP_AND_INSTALLED) do |event|
  wait 5
  info "Press enter to continue"
  STDIN.gets
  group('Actor').startApplications
  wait 50
  group('Actor').stopApplications
  info "Experiment is done. Copying measurements to /home/geni/experiment_logs"
  info "Enter prefix: "
  prefix=STDIN.gets
  prefix.strip!
  FileUtils.cp "/tmp/#{Experiment.ID}.sq3" ,
               "/home/geni/experiment_logs/#{prefix}_#{Experiment.ID}.sq3"
  info "files copied."
  Experiment.done
end
```

```
# Adding support for visualization
addTab(:defaults)
  addTab(:graph2) do |tab|
  opts = { :postfix => %{This graph shows the throughput.},
           :updateEvery => 1
  }
  tab.addGraph("Throughput", opts) do |g|
    data = []
    mp = ms('iperf_TCP_Info')
    mp.project(:oml_ts_server, :Bandwidth).each do |sample|
      time, bw = sample.tuple
      data << [time, bw]
    end
    g.addLine(data, :label => "Bandwidth")
  end
end
```

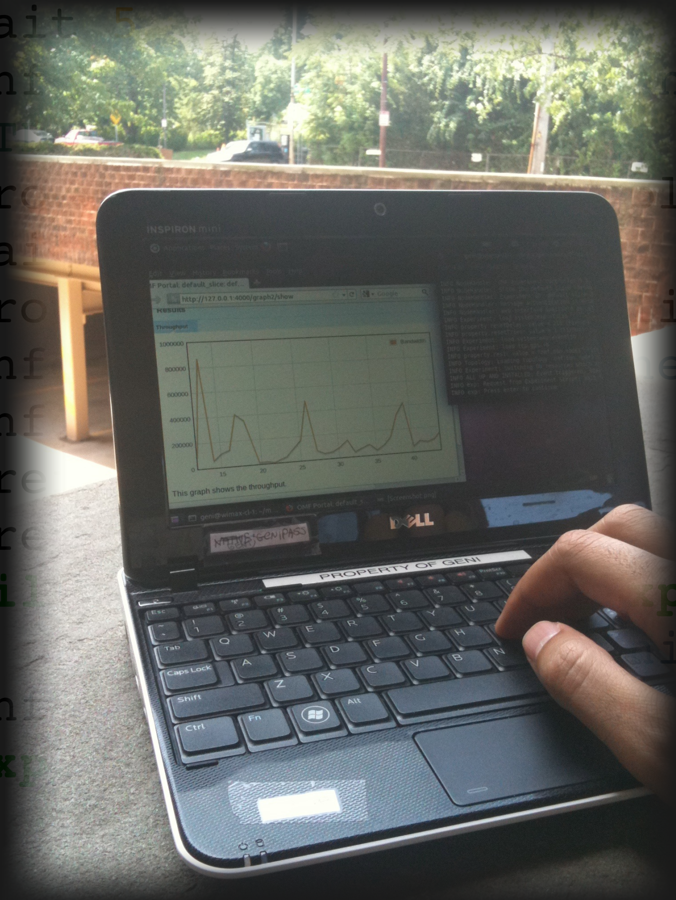
```
# Start up the experiment when all nodes are ready
onEvent(:ALL_UP_AND_INSTALLED) do |event|
  wait 5
  info "Press enter to continue"
  STDIN.gets
  group('Actor').startApplications
  wait 50
  group('Actor').stopApplications
  info "Experiment is done. Copying measurements to /home/geni/experiment_logs"
  info "Enter prefix: "
  prefix=STDIN.gets
  prefix.strip!
  FileUtils.cp "/tmp/#{Experiment.ID}.sq3" ,
               "/home/geni/experiment_logs/#{prefix}_#{Experiment.ID}.sq3"
  info "files copied."
  Experiment.done
end
```

```
# Adding support for visualization
addTab(:defaults)
  addTab(:graph2) do |tab|
  opts = { :postfix => %{This graph shows the throughput.},
           :updateEvery => 1
  }
  tab.addGraph("Throughput", opts) do |g|
    data = []
    mp = ms('iperf_TCP_Info')
    mp.project(:oml_ts_server, :Bandwidth).each do |sample|
      time, bw = sample.tuple
      data << [time, bw]
    end
    g.addLine(data, :label => "Bandwidth")
  end
end
```

```
# Start up the experiment when all nodes are ready
onEvent(:ALL_UP_AND_INSTALLED) do |event|
  wait 5
  info "Press enter to continue"
  STDIN.gets
  group('Actor').startApplications
  wait 50
  group('Actor').stopApplications
  info "Experiment is done. Copying measurements to /home/geni/experiment_logs"
  info "Enter prefix: "
  prefix=STDIN.gets
  prefix.strip!
  FileUtils.cp "/tmp/#{Experiment.ID}.sq3" ,
               "/home/geni/experiment_logs/#{prefix}_#{Experiment.ID}.sq3"
  info "files copied."
  Experiment.done
end
```

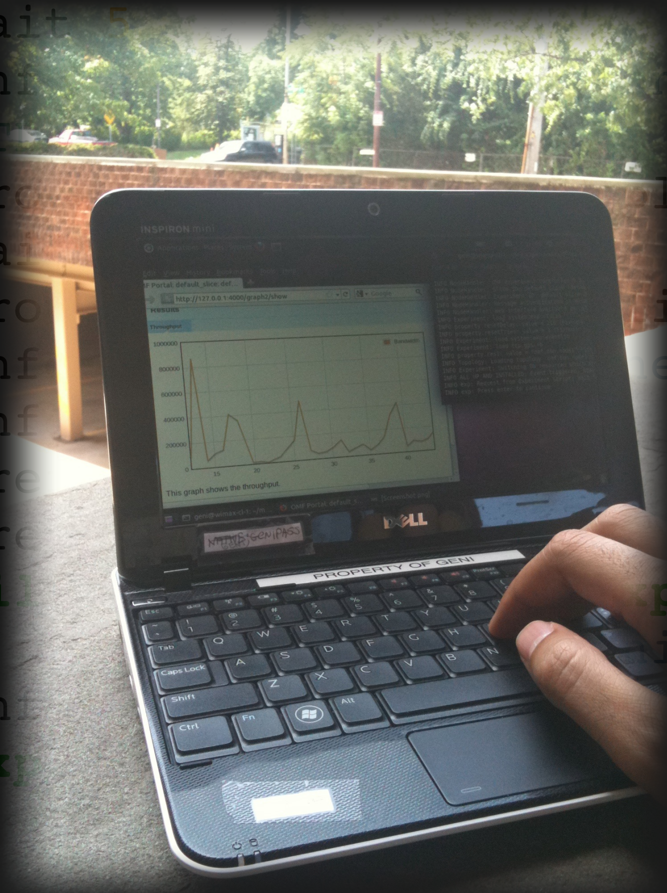
```
# Adding support for visualization
addTab(:defaults)
  addTab(:graph2) do |tab|
  opts = { :postfix => %{This graph shows the throughput.},
           :updateEvery => 1
  }
  tab.addGraph("Throughput", opts) do |g|
    data = []
    mp = ms('iperf_TCP_Info')
    mp.project(:oml_ts_server, :Bandwidth).each do |sample|
      time, bw = sample.tuple
      data << [time, bw]
    end
    g.addLine(data, :label => "Bandwidth")
  end
end
```

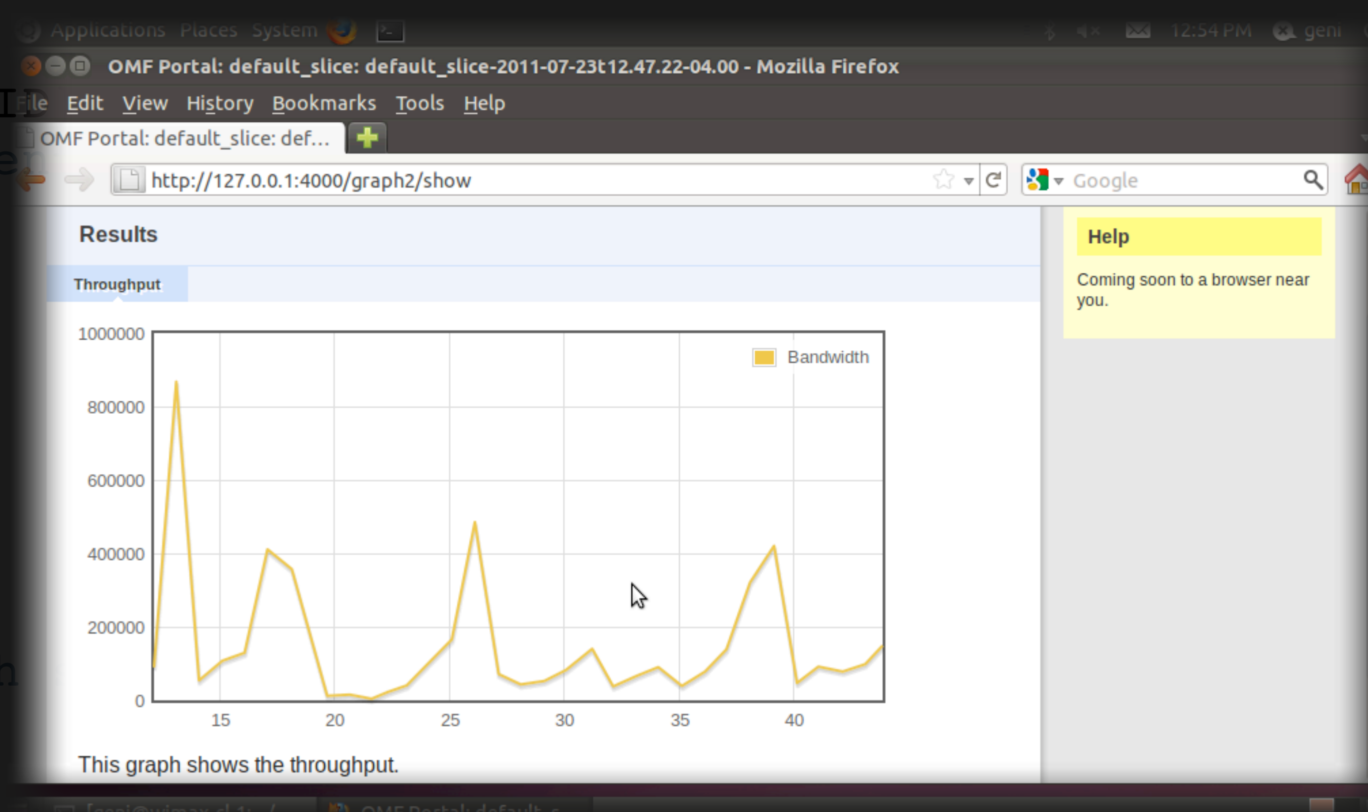
```
# Start up the experiment when all nodes are ready
onEvent(:ALL_UP_AND_INSTALLED) do |event|
  wait
  in                   ntinue"
  ST
  gr                   lications
  wa
  gr                   ications
  in                   e. Copying measurements to /home/geni/experiment_logs"
  in
  pr
  pr
  Fi                   periment.ID}.sq3" ,
                       i/experiment_logs/#{prefix}_#{Experiment.ID}.sq3"
  in
  Ex
end
```



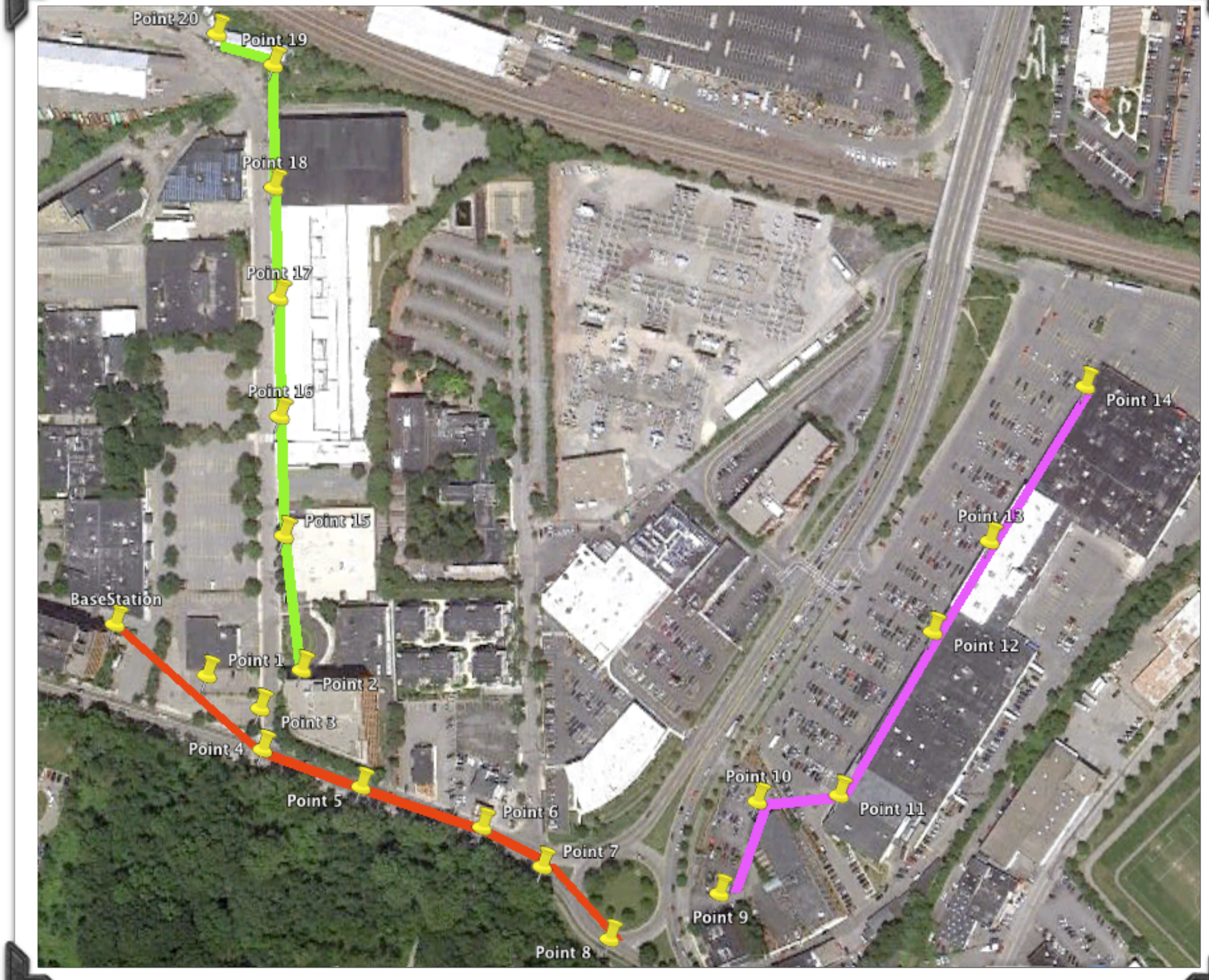

```
# Adding support for visualization
addTab(:defaults)
  addTab(:graph2) do |tab|
  opts = { :postfix => %{This graph shows the throughput.},
           :updateEvery => 1
  }
  tab.addGraph("Throughput", opts) do |g|
    data = []
    mp = ms('iperf_TCP_Info')
    mp.project(:oml_ts_server, :Bandwidth).each do |sample|
      time, bw = sample.tuple
      data << [time, bw]
    end
    g.addLine(data, :label => "Bandwidth")
  end
end
```

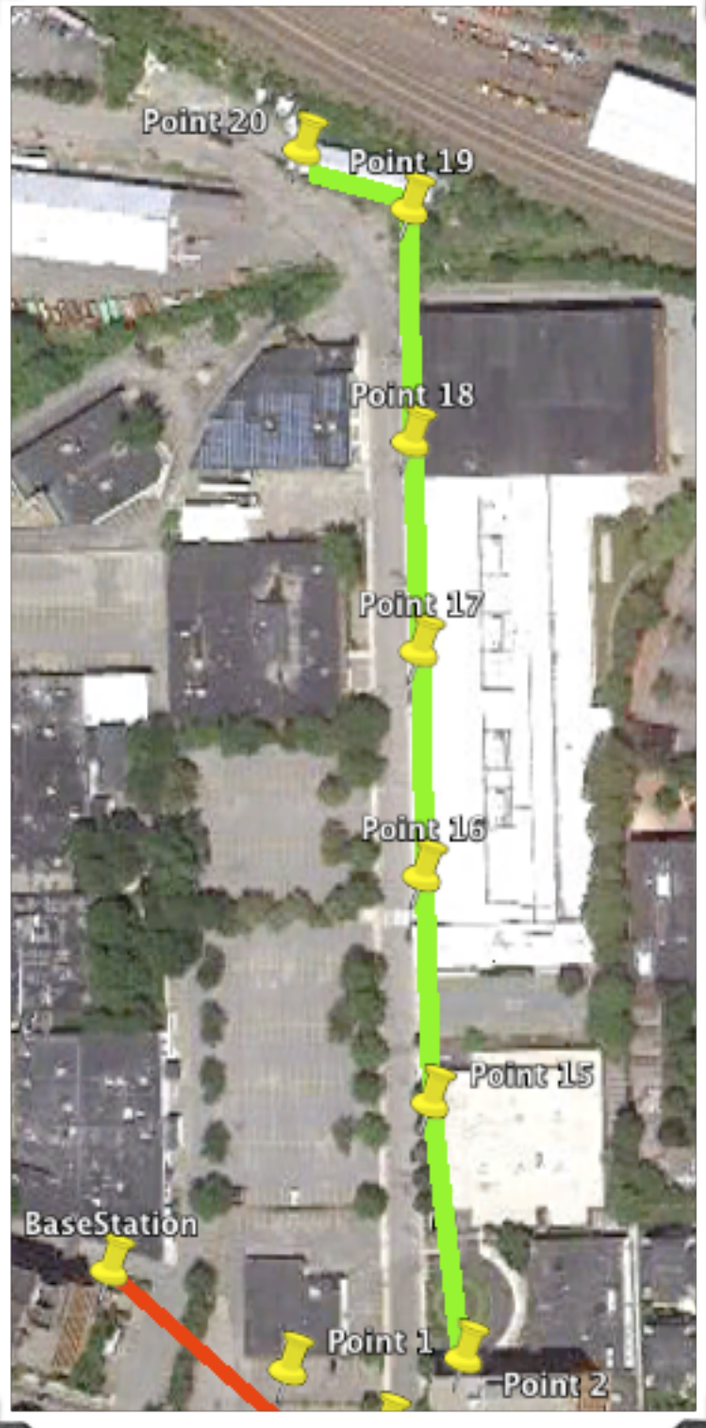
```
# Start up the experiment when all nodes are ready
onEvent(:ALL_UP_AND_INSTALLED) do |event|
  wai
  in                              ntinue"
  ST
  gr                             lications
  wa
  gro                            ications
  inf                           e. Copying measurements to /home/geni/experiment_logs"
  inf
  pr
  pr
  Fi                           periment.I
                                i/experime
  in
  Ex
end
```

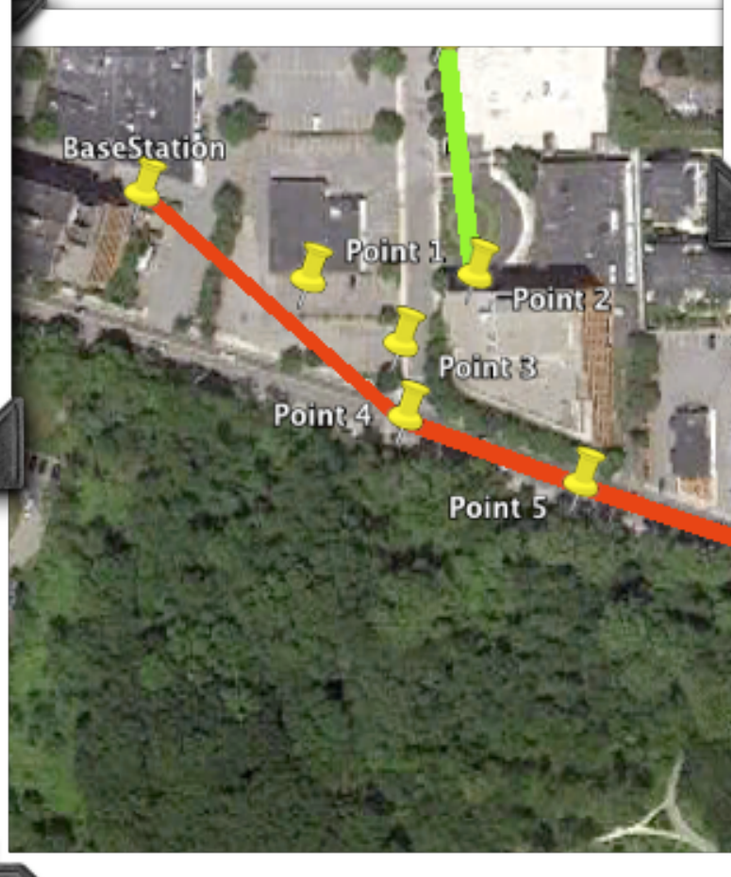


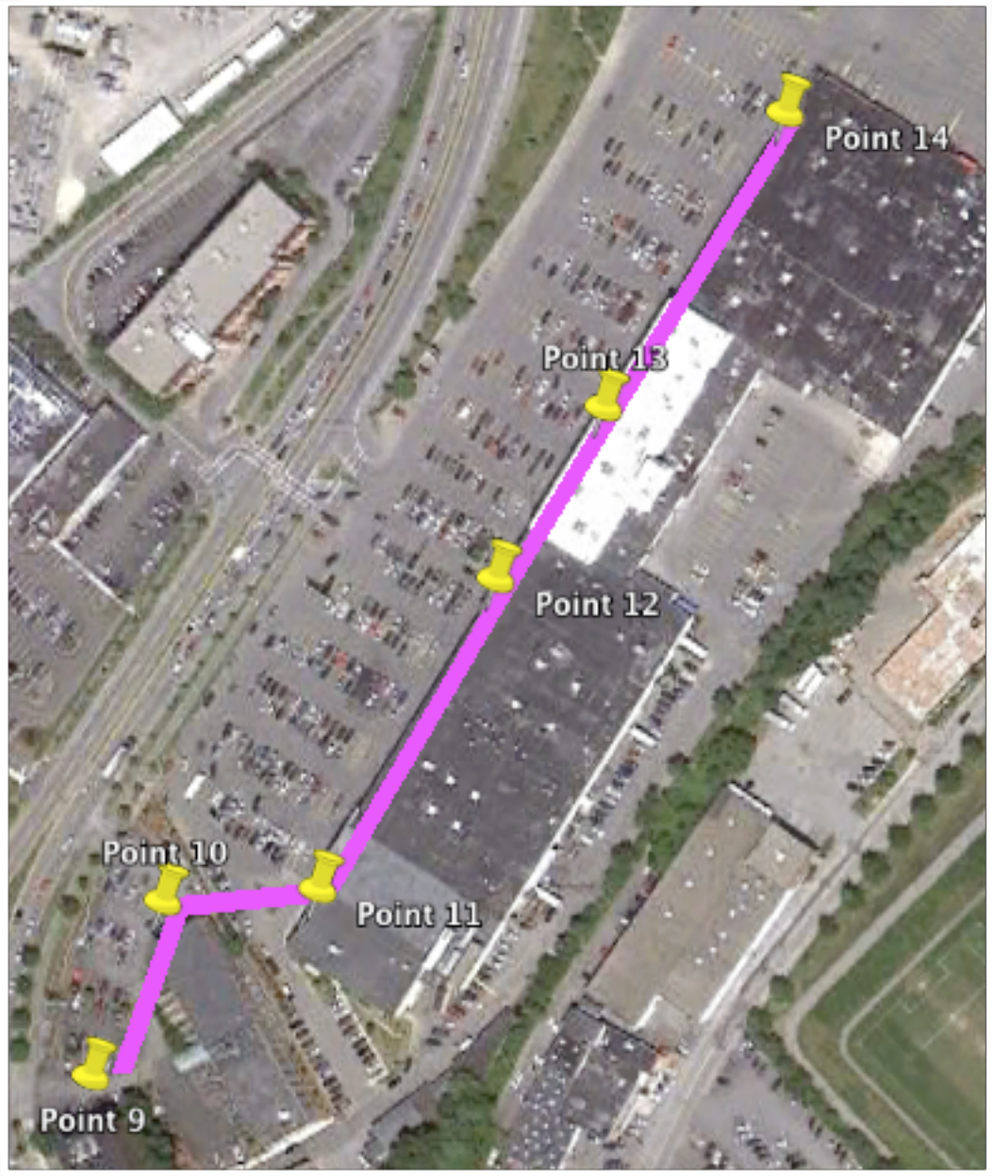


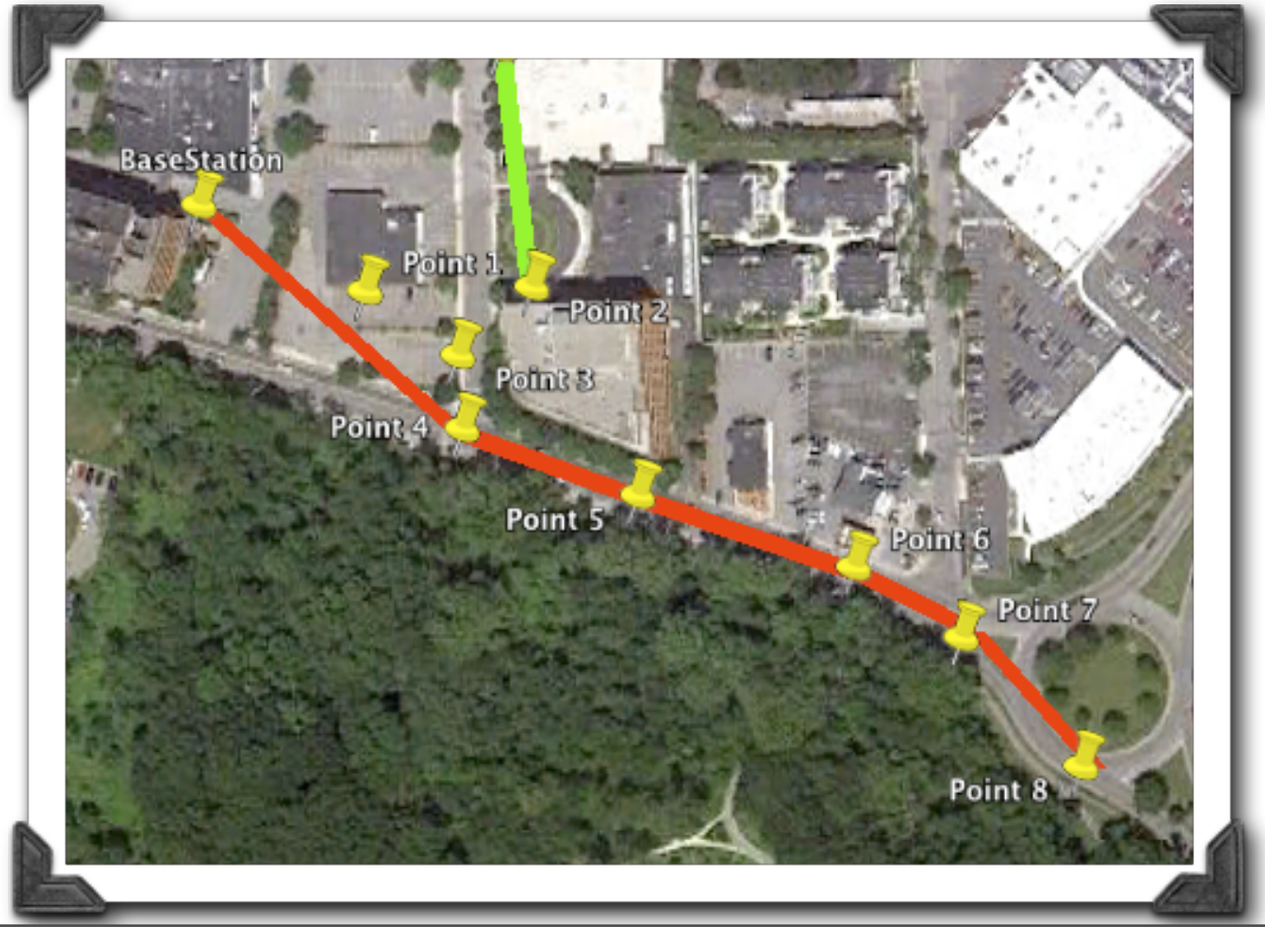

```
# Adding support for visualization
addTab(:defaults)
  addTab(:graph2) do |tab|
  opts = { :postfix => %{This graph
           :updateEvery => 1
  }
  tab.addGraph("Throughput", opts) do |g|
    data = []
    mp = ms('iperf_TCP_Info')
    mp.project(:oml_ts_server, :Bandwidth).each do |sample|
      time, bw = sample.tuple
      data << [time, bw]
    end
    g.addLine(data, :label => "Bandwidth")
  end
end
```

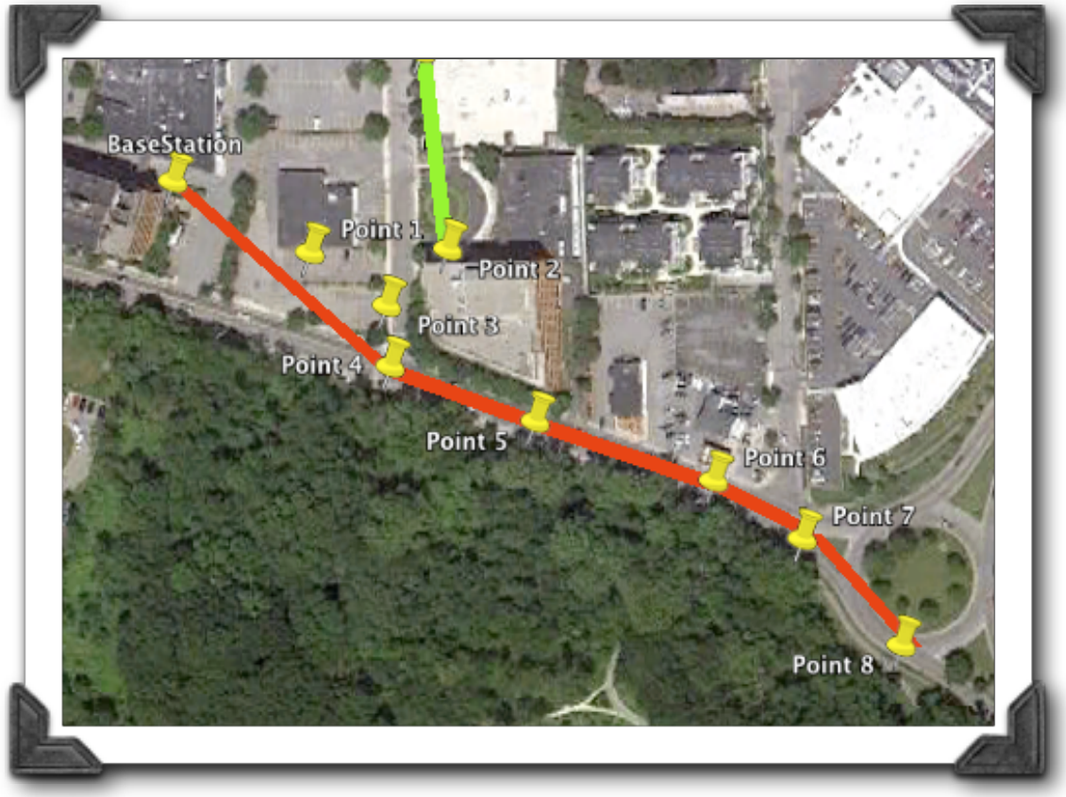

Point 20
Point 19
Point 18
Point 17
Point 16
Point 15
BaseStation
Point 1
Point 2
Point 3
Point 4
Point 5
Point 6
Point 7
Point 8
Point 9
Point 10
Point 11
Point 12
Point 13
Point 14

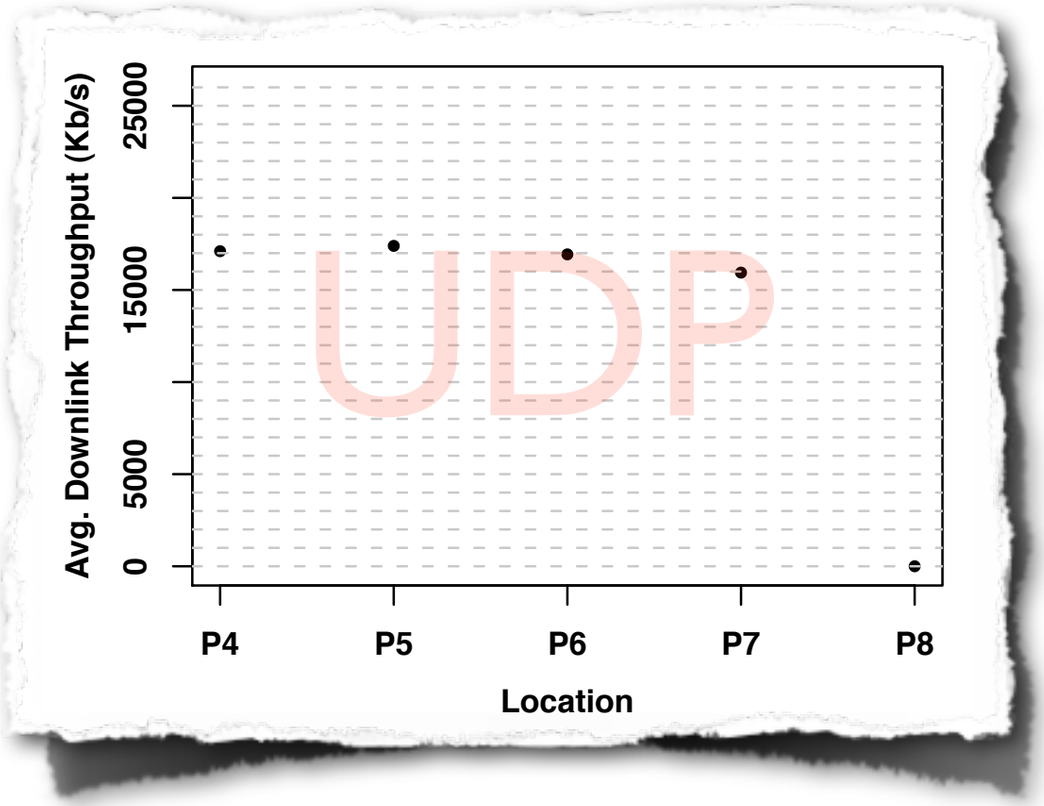

Point 20
Point 19
Point 18
Point 17
Point 16
Point 15
BaseStation
Point 1
Point 2

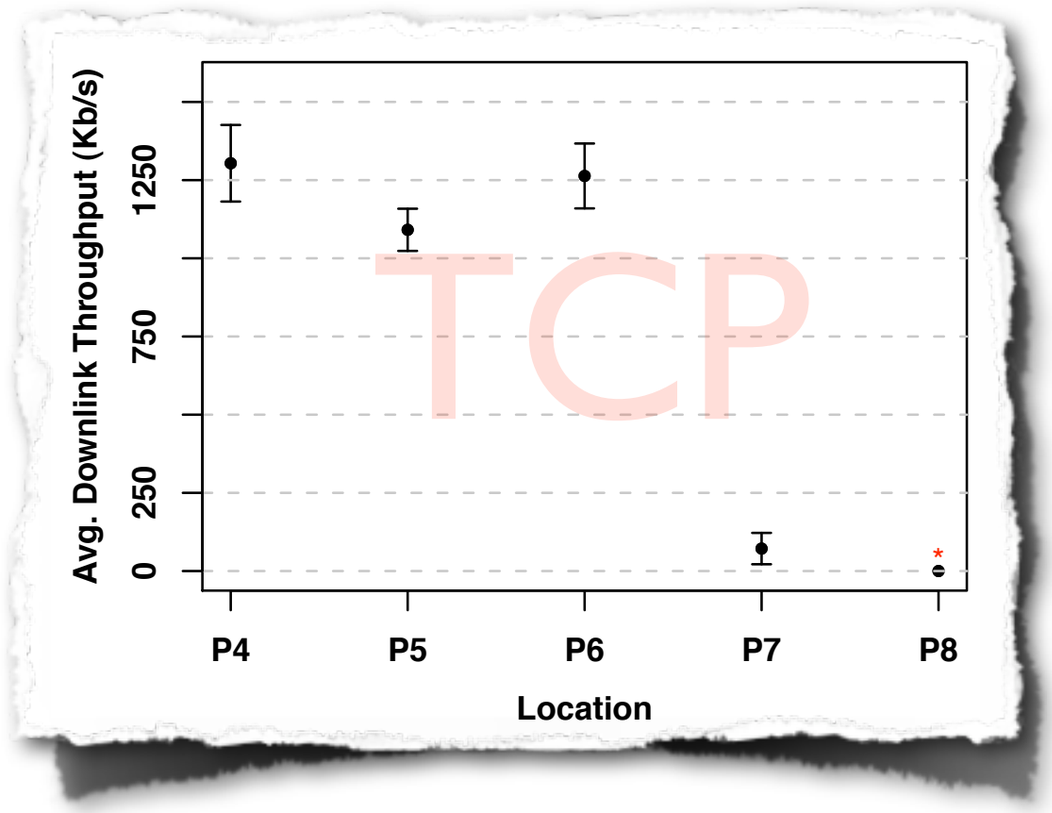

BaseStation
Point 1
Point 2
Point 3
Point 4
Point 5
Point 6
Point 7
Point 8

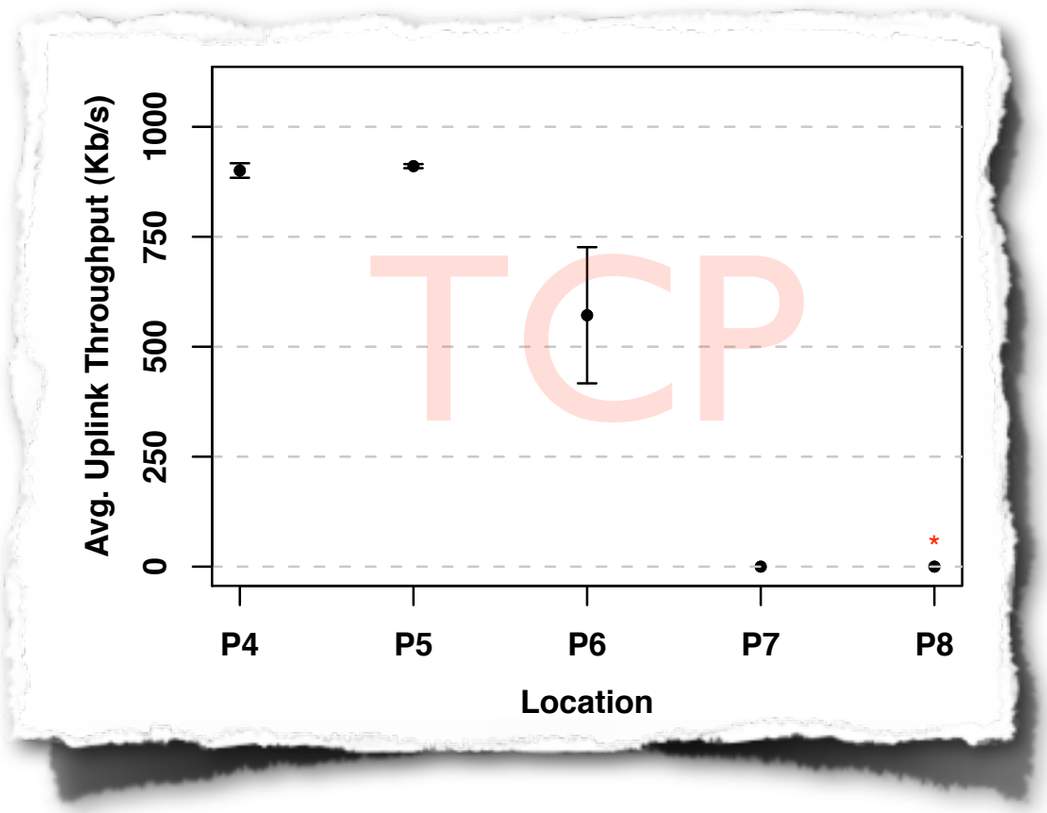

Point 14
Point 13
Point 12
Point 10
Point 11
Point 9

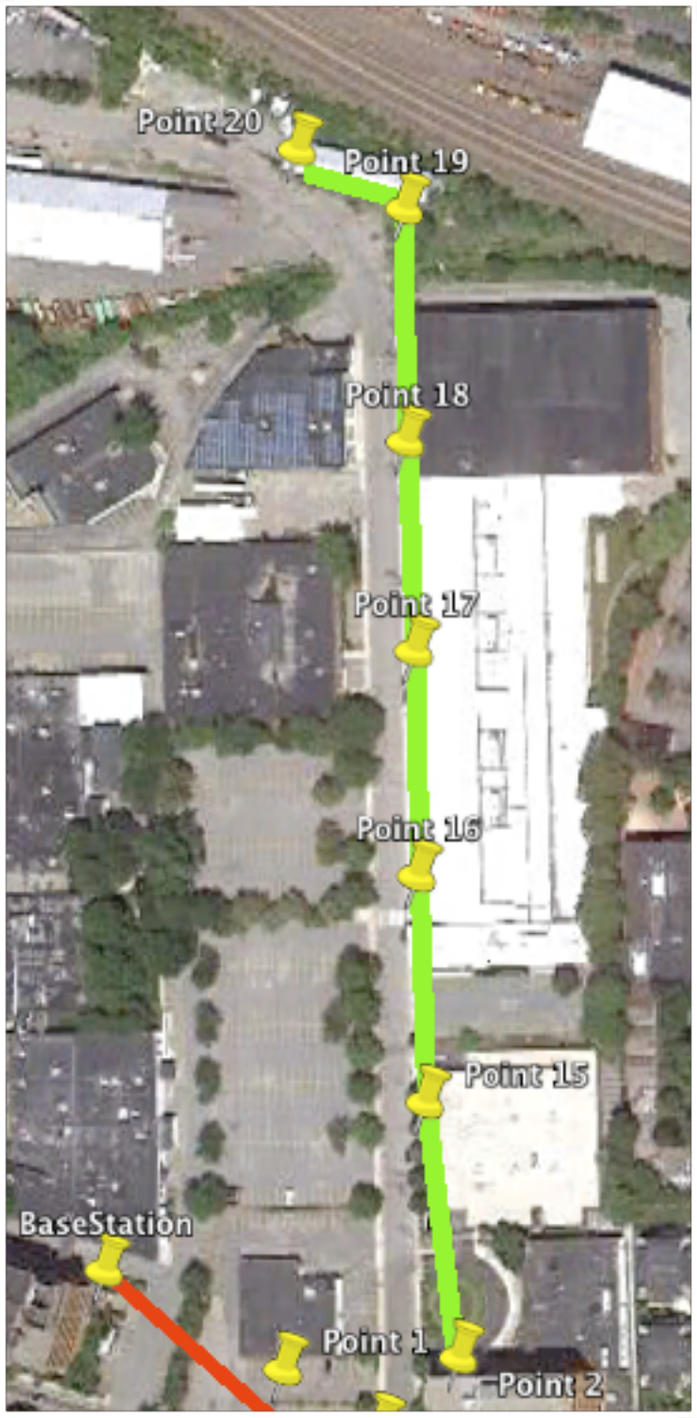

BaseStation
Point 1
Point 2
Point 3
Point 4
Point 5
Point 6
Point 7
Point 8

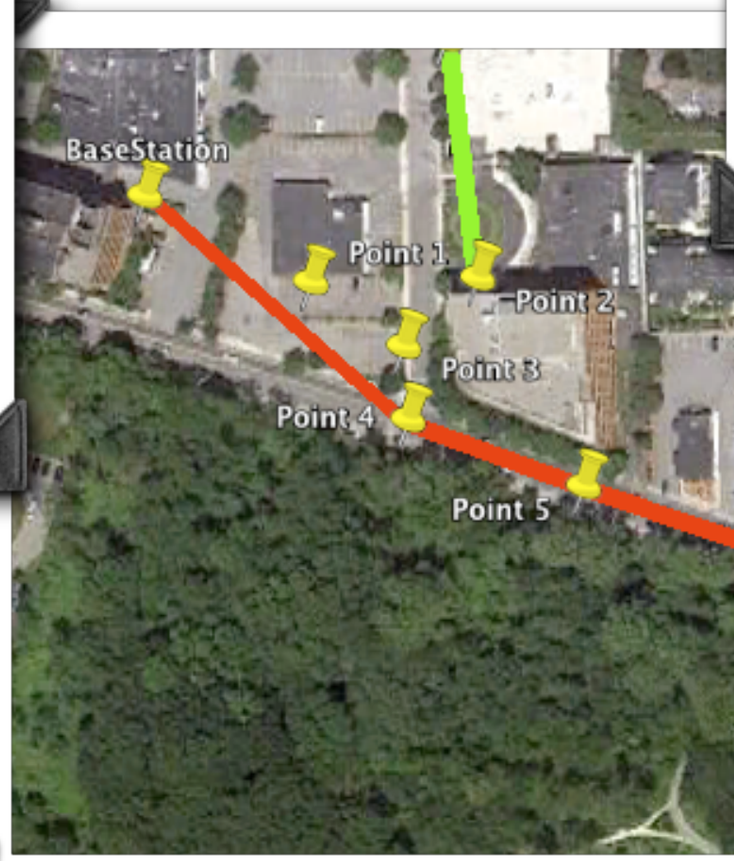

BaseStation
Point 1
Point 2
Point 3
Point 4
Point 5
Point 6
Point 7
Point 8

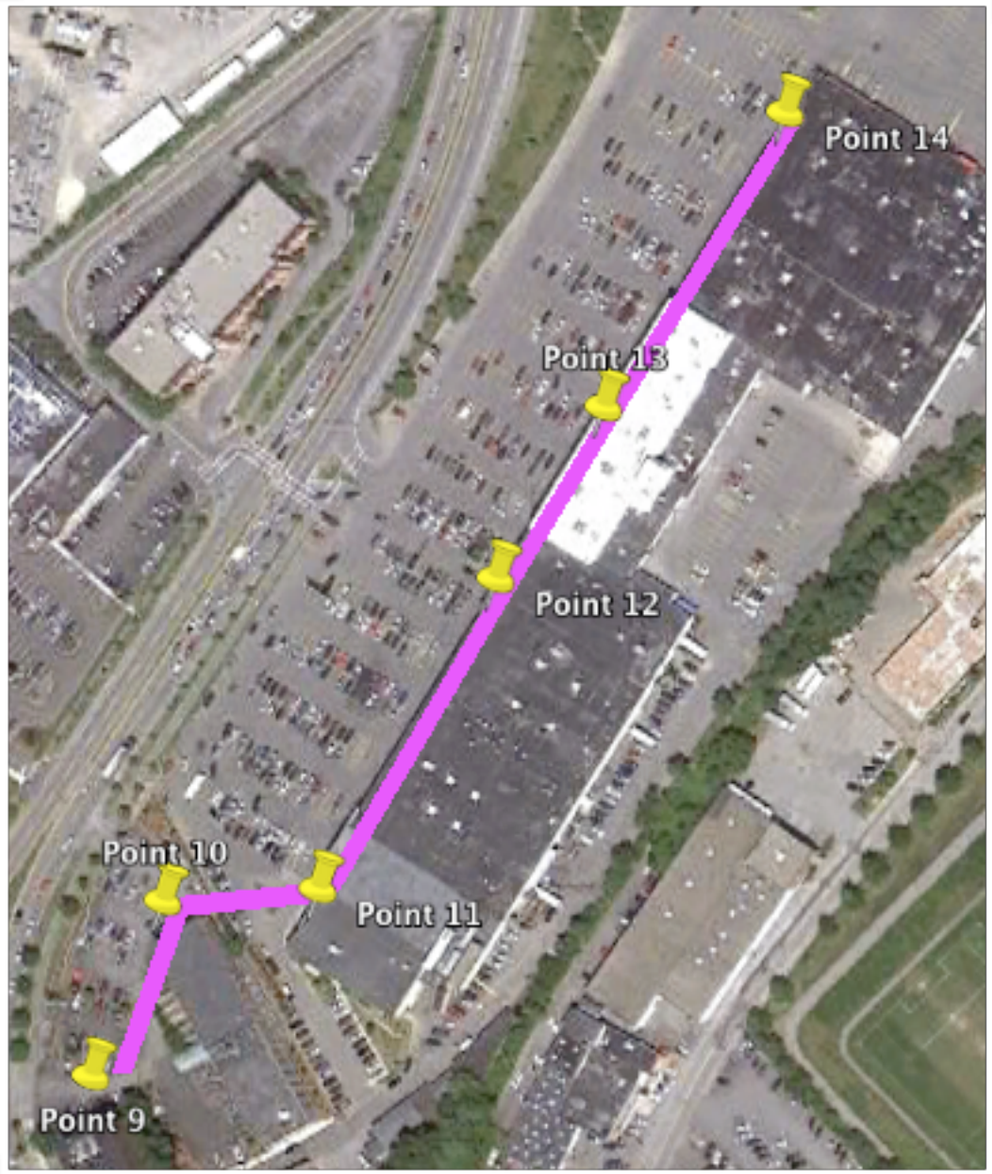

Avg. Downlink Throughput (Kb/s)
0
5000
15000
25000
P4
P5
P6
P7
P8
Location
UDP

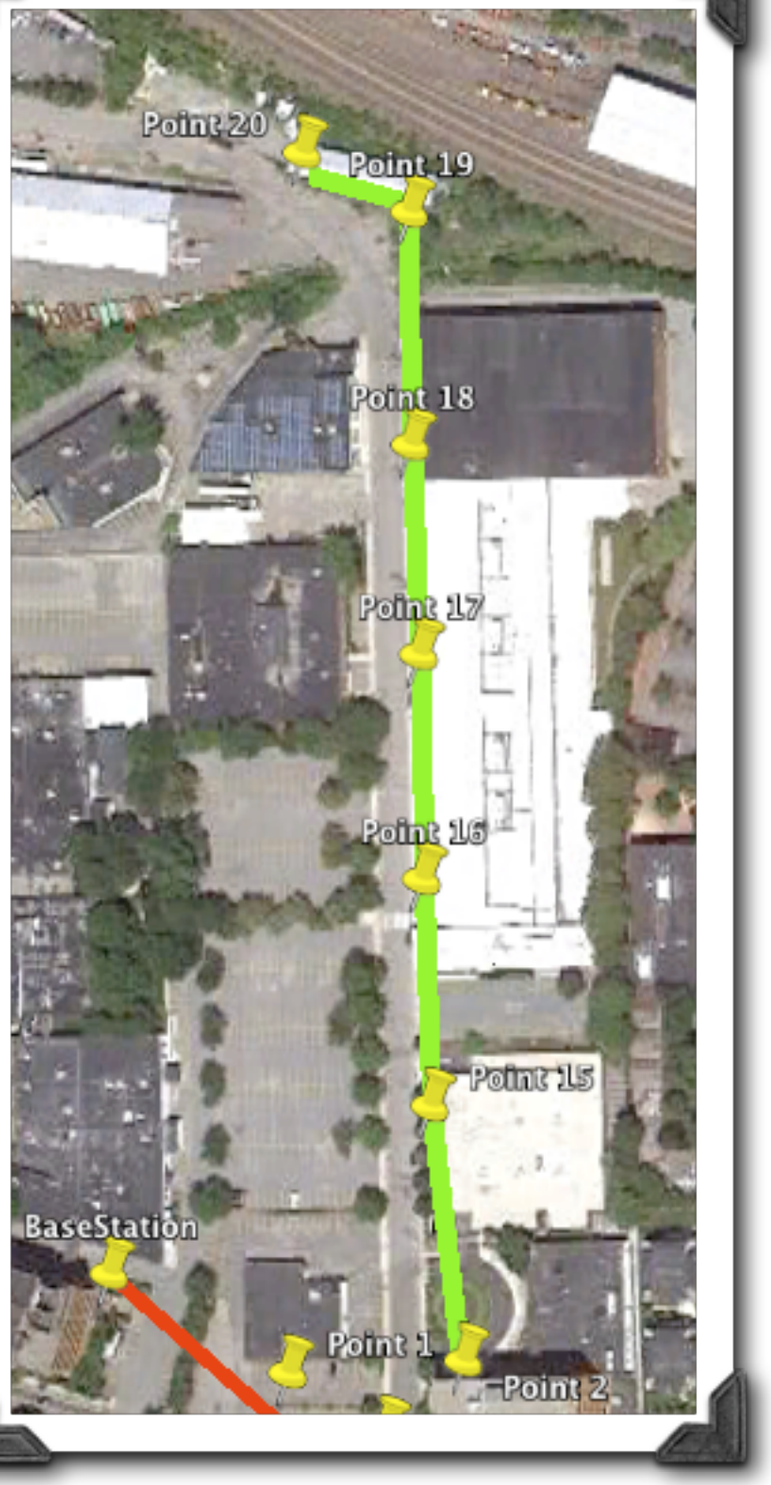

Avg. Downlink Throughput (Kb/s)
0
250
750
1250
P4
P5
P6
P7
P8
Location
TCP

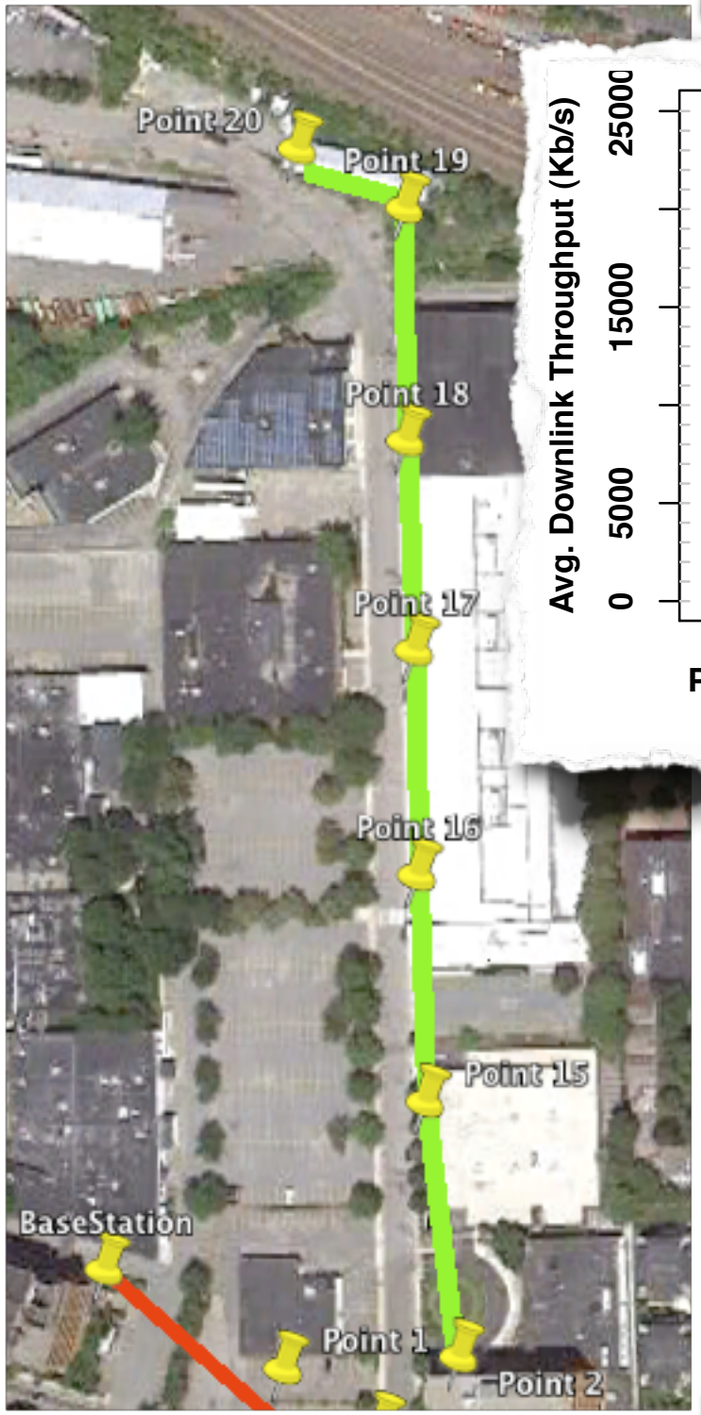

Avg. Uplink Throughput (Kb/s)
0
250
500
750
1000
P4
P5
P6
P7
P8
Location
TCP

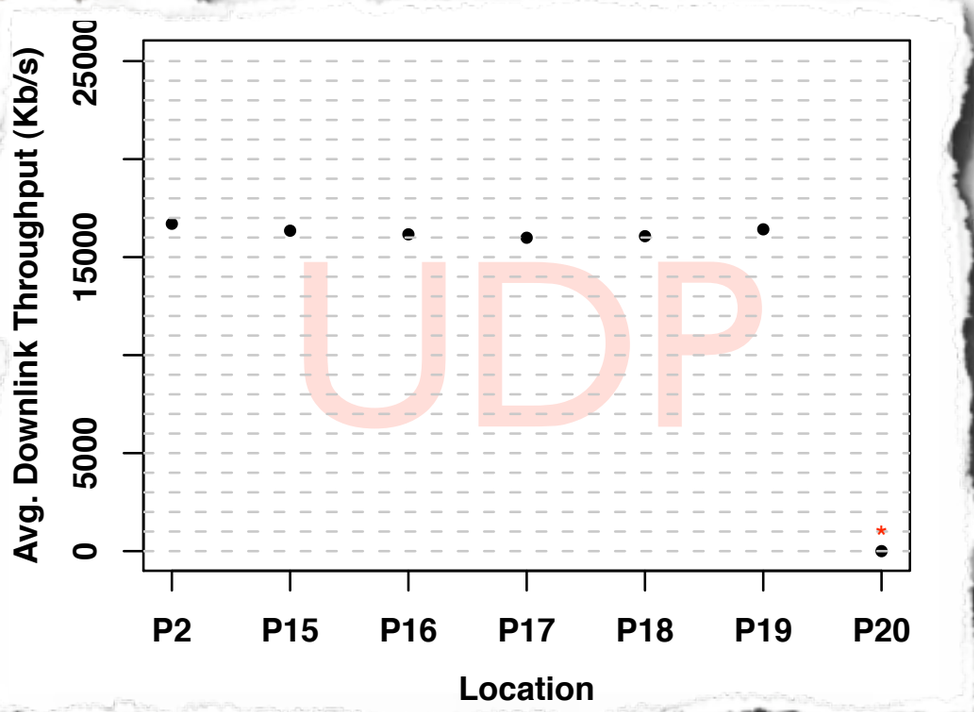

Point 20
Point 19
Point 18
Point 17
Point 16
Point 15
BaseStation
Point 1
Point 2

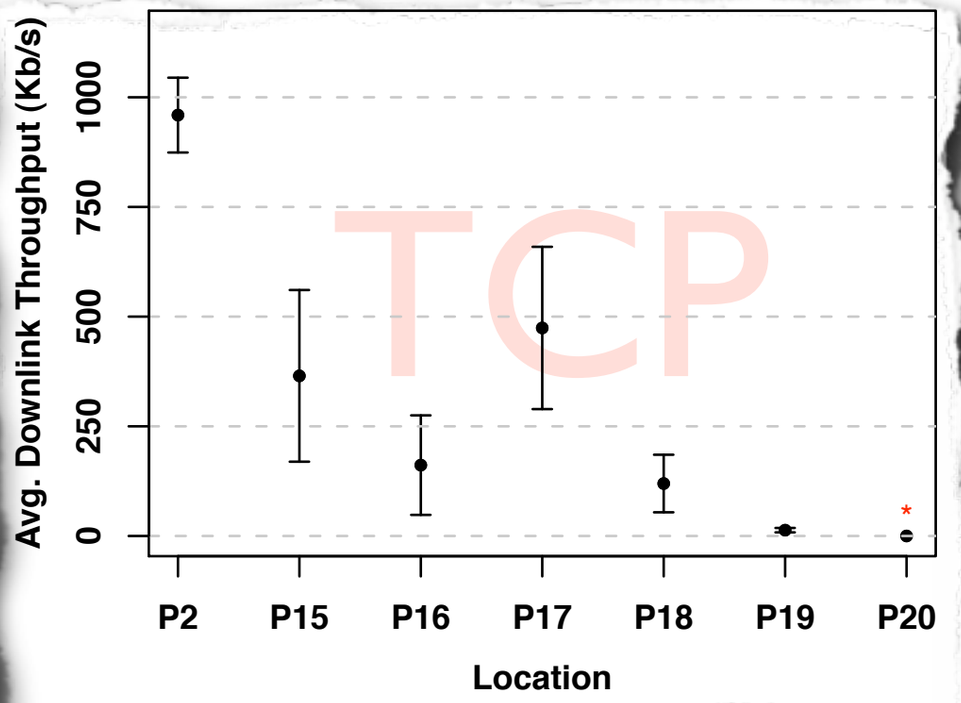

BaseStation
Point 1
Point 2
Point 3
Point 4
Point 5
Point 6
Point 7
Point 8

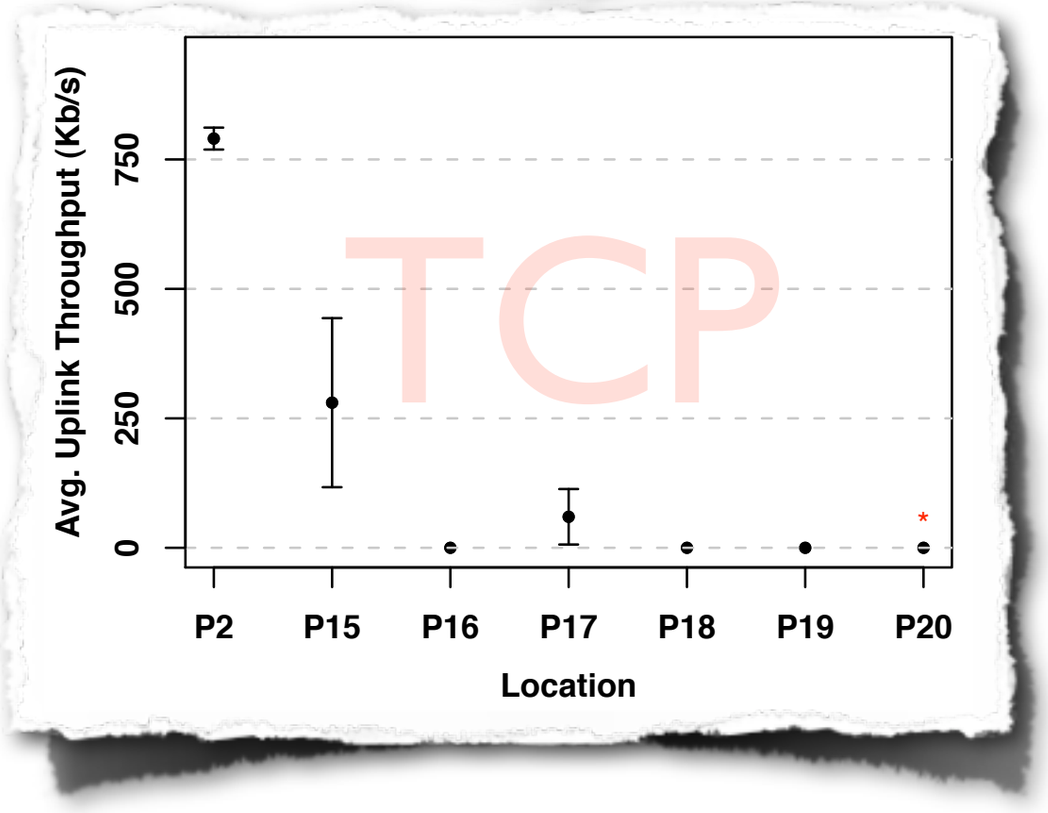

Point 14
Point 13
Point 12
Point 10
Point 11
Point 9

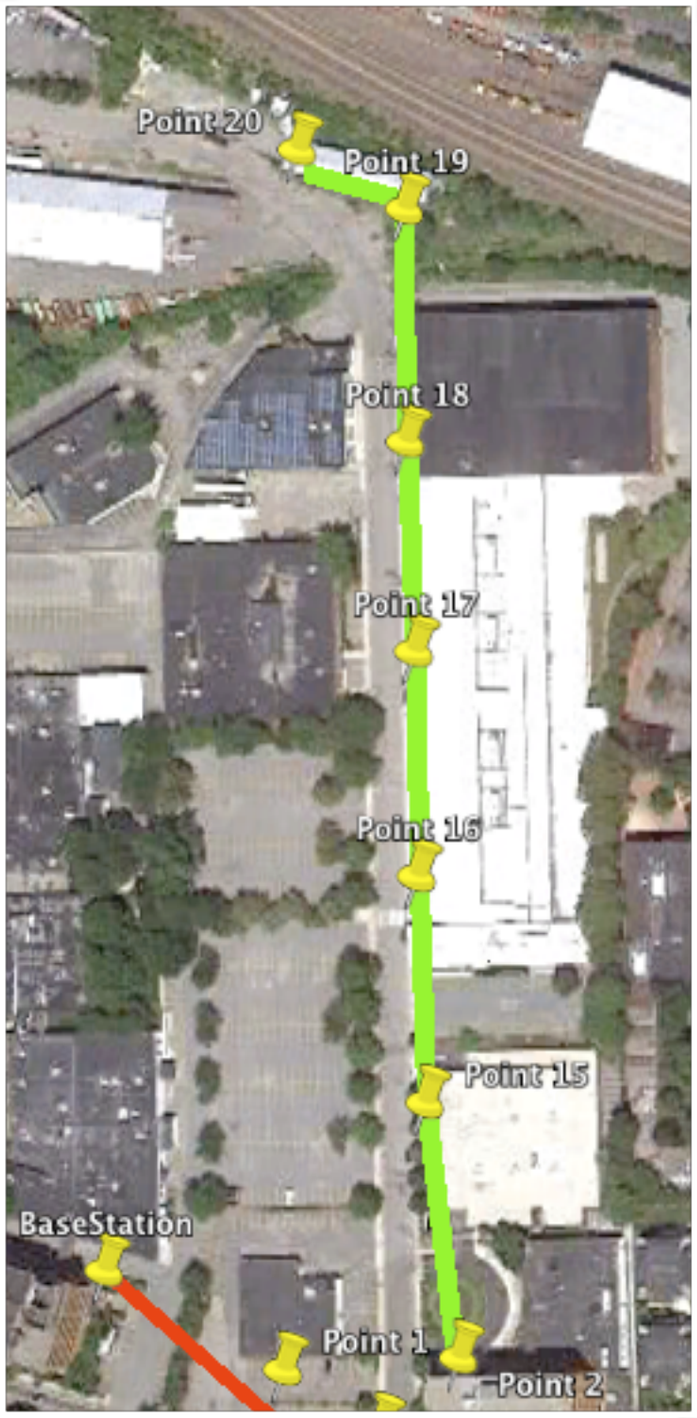

Point 20
Point 19
Point 18
Point 17
Point 16
Point 15
BaseStation
Point 1
Point 2

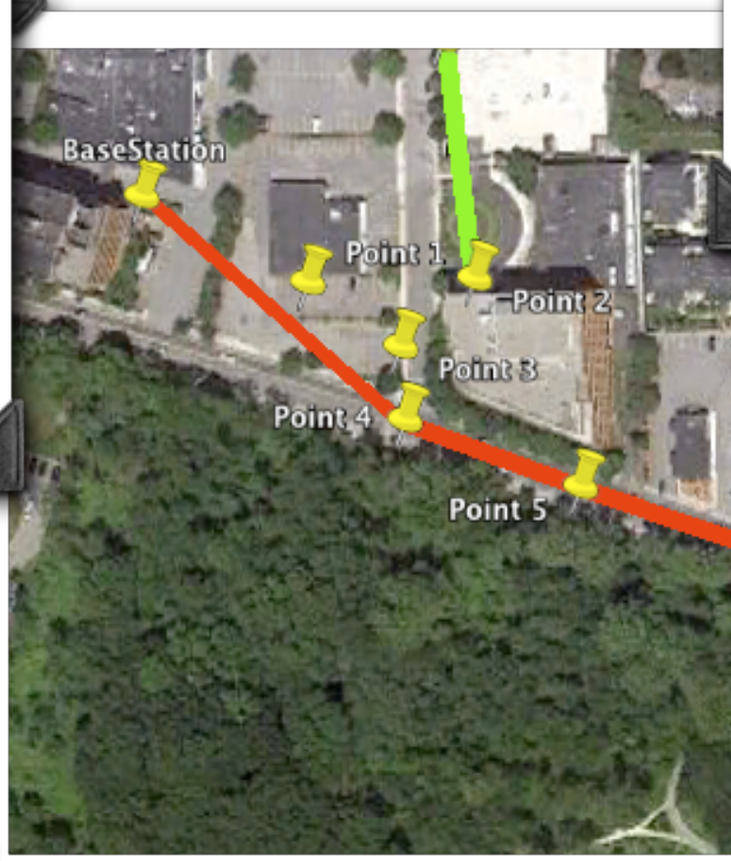

Point 20
Point 19
Point 18
Point 17
Point 16
Point 15
BaseStation
Point 1
Point 2

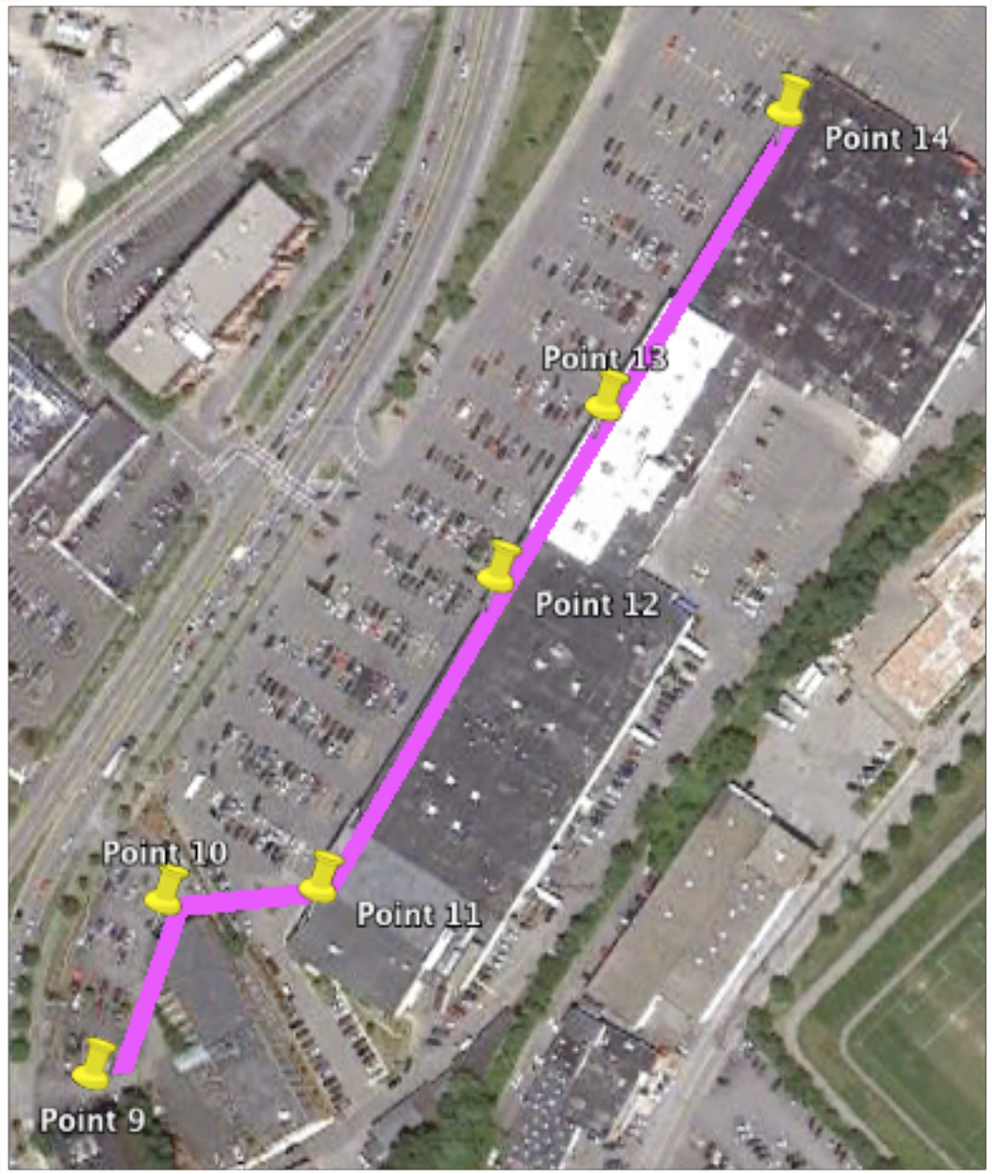

Avg. Downlink Throughput (Kb/s)
0
5000
15000
25000
UDP
P2
P15
P16
P17
P18
P19
P20
Location

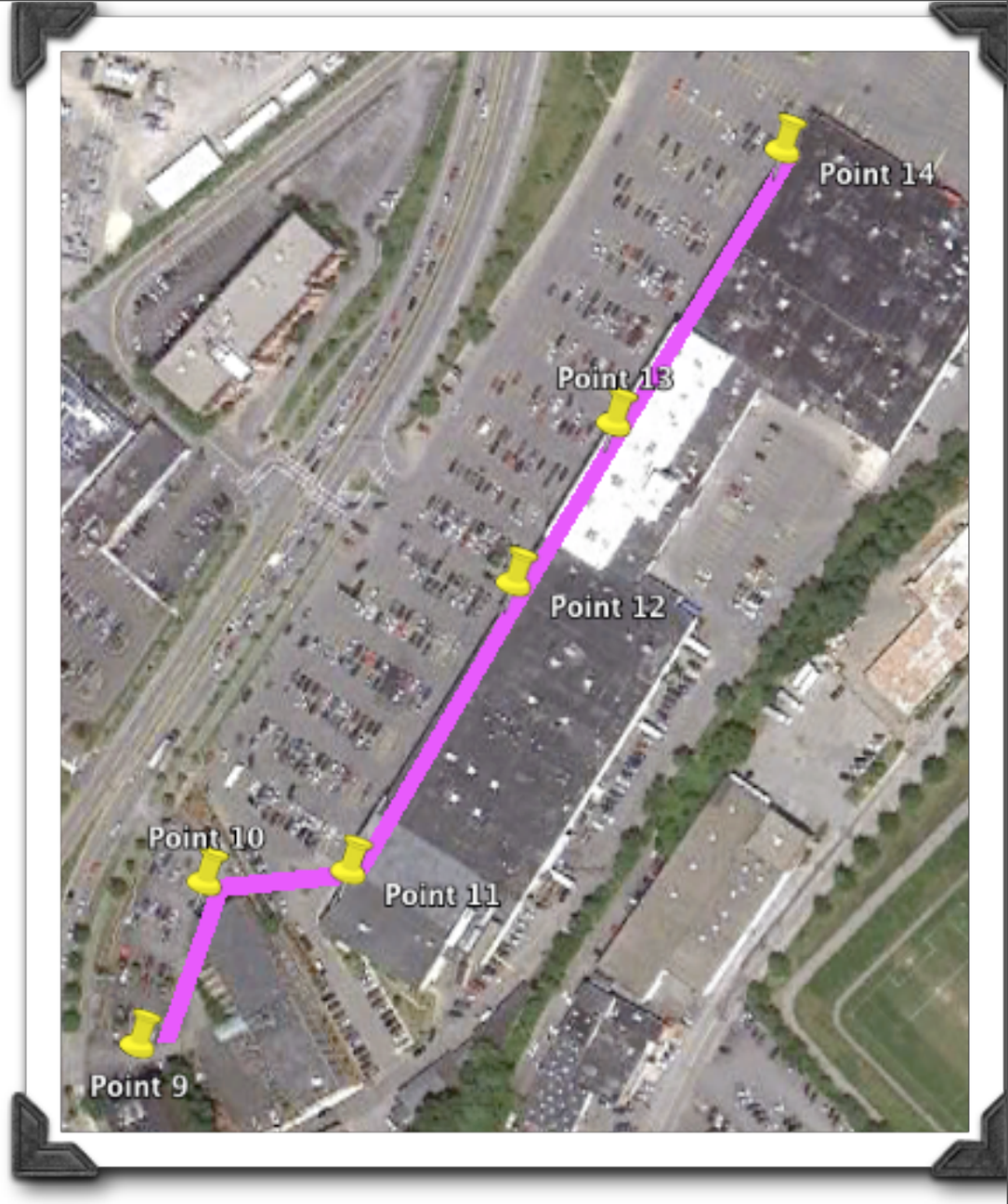

Avg. Downlink Throughput (Kb/s)
0
250
500
750
1000
TCP
P2
P15
P16
P17
P18
P19
P20
Location

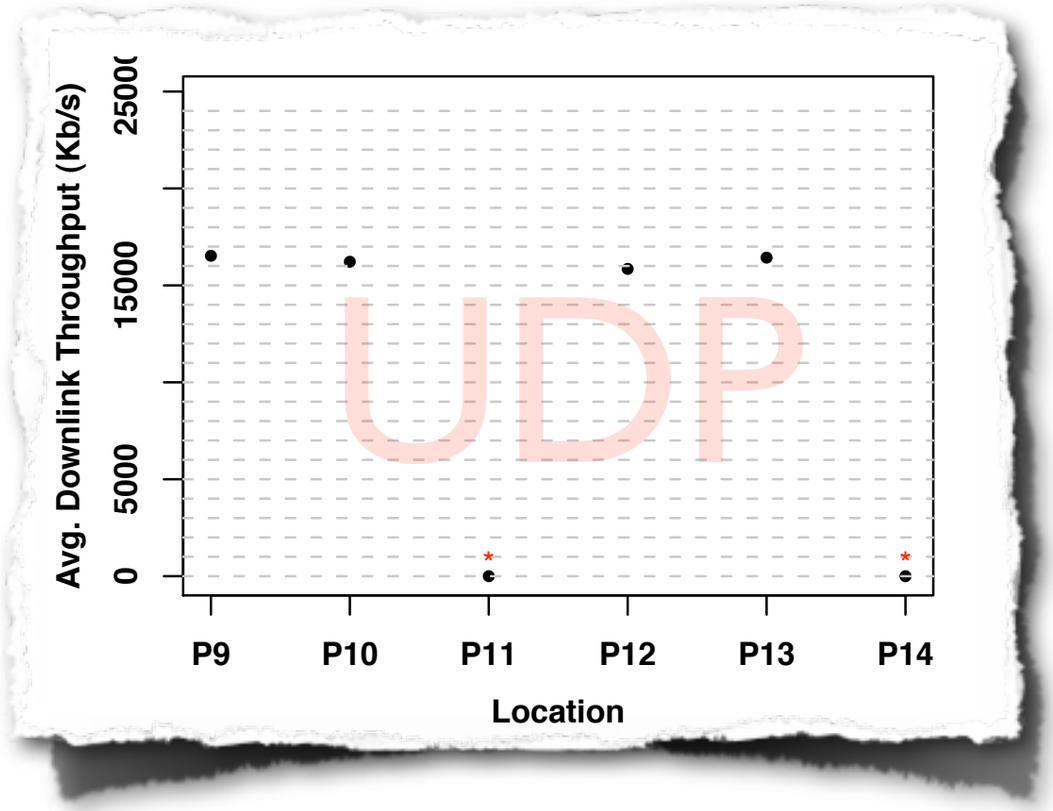

Avg. Uplink Throughput (Kb/s)
0
250
500
750
TCP
P2
P15
P16
P17
P18
P19
P20
Location

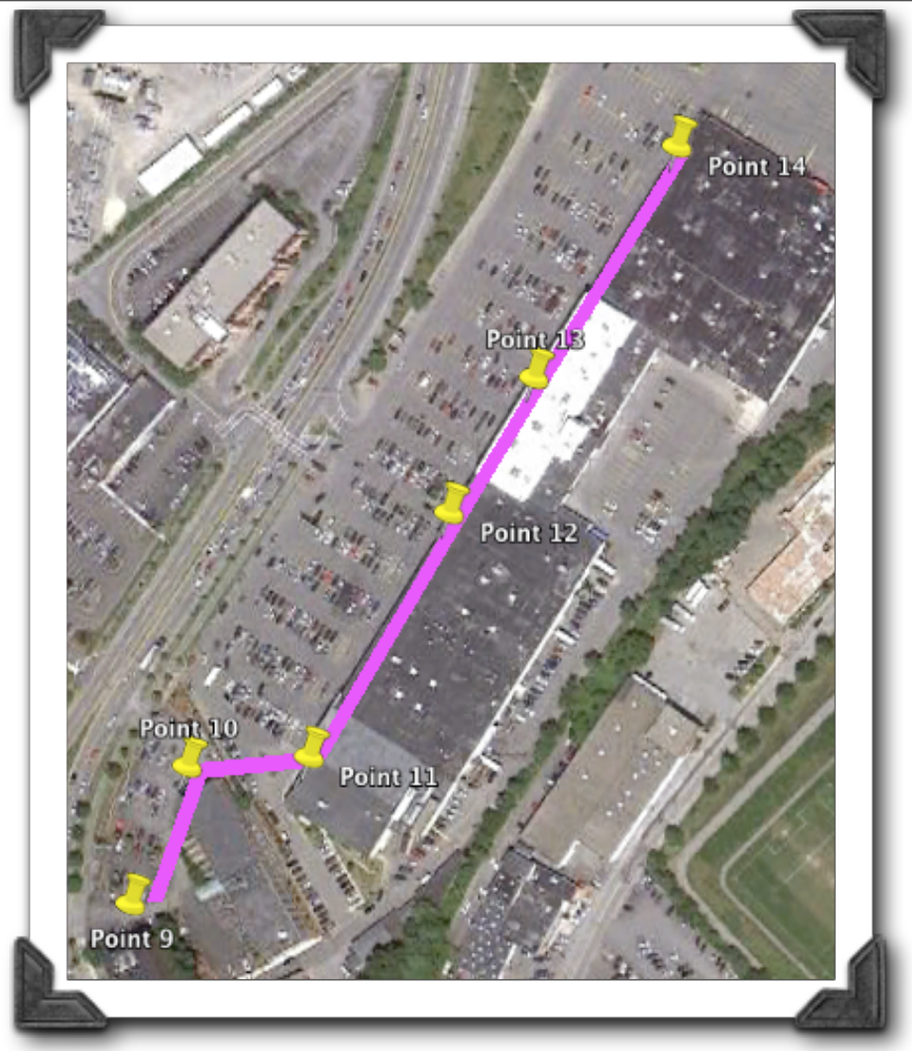

Point 20
Point 19
Point 18
Point 17
Point 16
Point 15
BaseStation
Point 1
Point 2

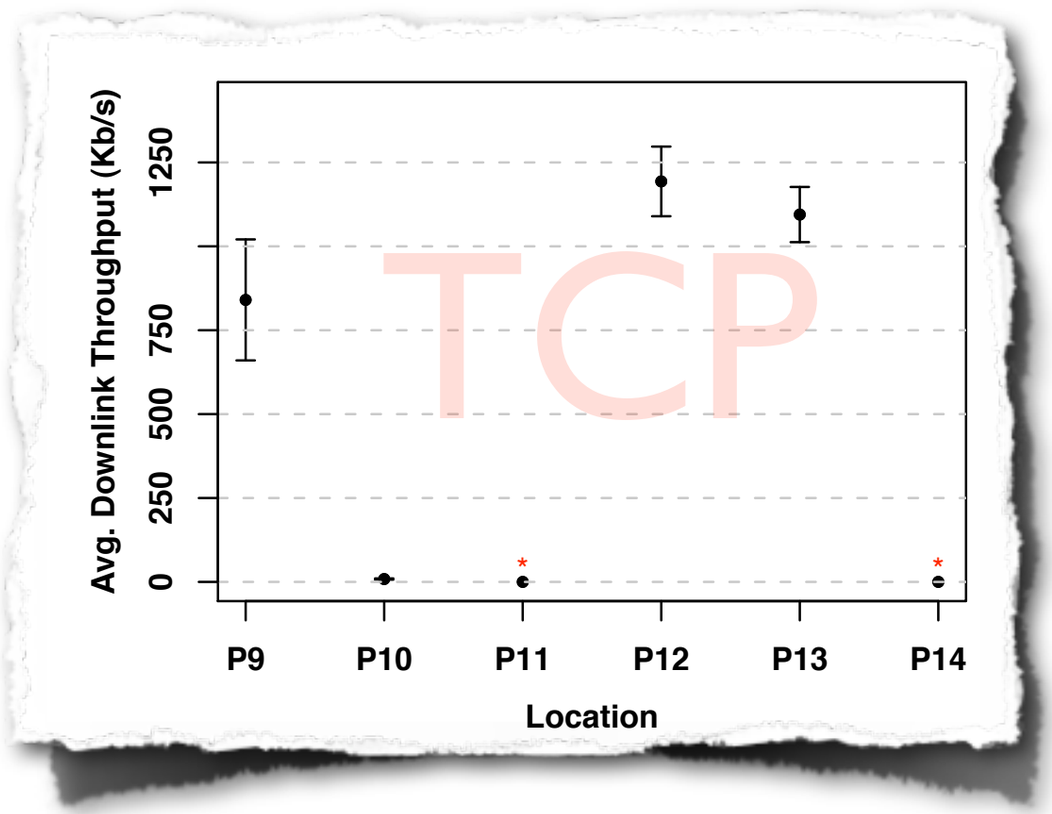

BaseStation
Point 1
Point 2
Point 3
Point 4
Point 5
Point 6
Point 7
Point 8

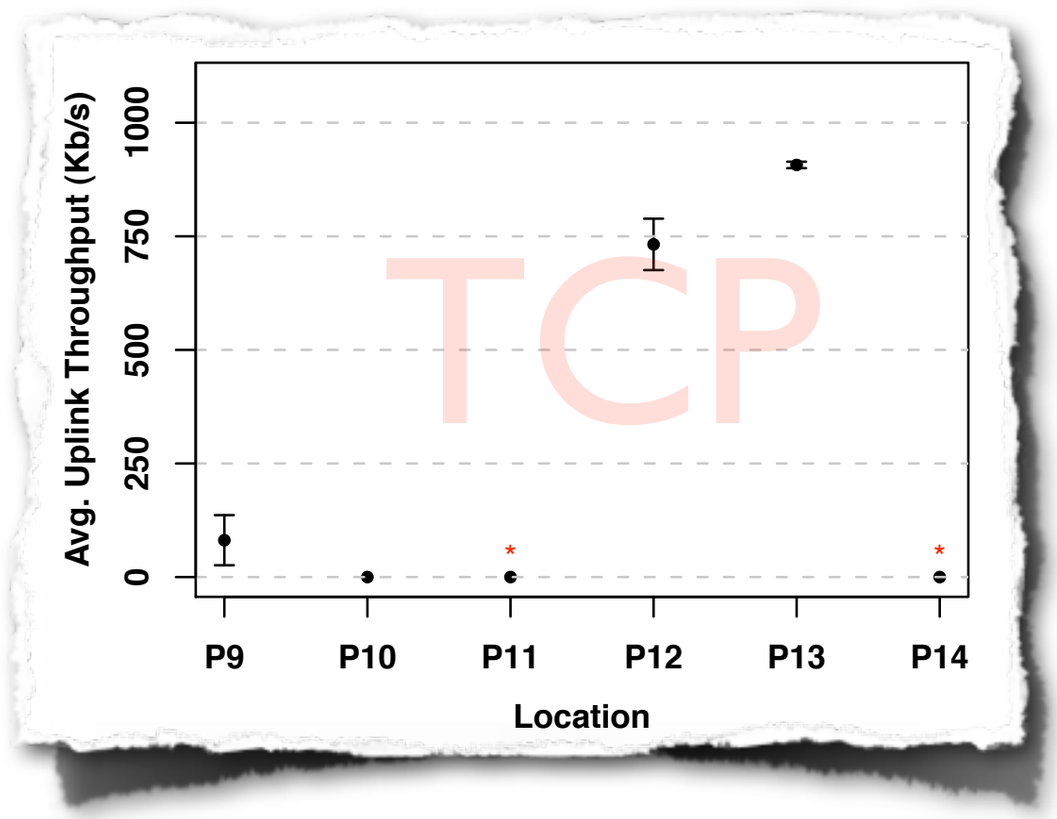

Point 14
Point 13
Point 12
Point 10
Point 11
Point 9


Point 14
Point 13
Point 12
Point 11
Point 10
Point 9


Avg. Downlink Throughput (Kb/s)
0
5000
15000
25000
P9
P10
P11
P12
P13
P14
Location
UDP


Point 14
Point 13
Point 12
Point 11
Point 10
Point 9


Avg. Downlink Throughput (Kb/s)
0
250
500
750
1250
P9
P10
P11
P12
P13
P14
Location
TCP


Avg. Uplink Throughput (Kb/s)
0
250
500
750
1000
P9
P10
P11
P12
P13
P14
Location
TCP