Energy Analysis of a Mesh Network using NS3 Network Simulator

Ajin Jiji Tom - 13EC202, Kevin D'Souza - 13EC256

I. AIM

To conduct Energy Analysis of a mesh network and get insight about the duration of effective operation of the nodes.The simulator used for this purpose is an open source network simulator called NS3 and ireshark is used as a packet sniffer.

II. MOTIVATION

Over the past several years the topics of energy consumption and energy harvesting have gained significant importance as a means for improved operation of wireless sensor and mesh networks. Energy-awareness of operation is especially relevant for application scenarios from the domain of environmental monitoring in hard to access areas. With our work we aim to shed some light on energy-aware network operation and to help both users and developers in the planning and deployment of a new wireless (mesh) network for environmental research and other applications.

III. TOPOLOGY

The Topology of the Mesh network used for simulation is shown below

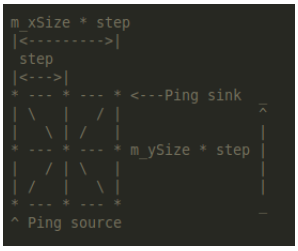

Fig. 1: 3*3 Mesh Topology

IV. ASSUMPTIONS

1) The initial Energy is taken to be 3J for all the nodes based on [1].
2) Reception of each byte is assumed to draw a constant energy and is not time dependent.
3) It is assumed that the drain of energy doesnt effect the voltage of operation.
4) The efficient working duration of the node is assumed to be till the time the node energy stays above 1/3rd of the initial energy according to [2].

V. FORMULAE USED

*Residual energy = Energy in the Previous Iteration - 3.3*I*No.of.bytes received*

Where *I* is the amount of current drawn per Byte transmission is assumed to be 0.12mA from [1].

VI. NS3 CODE SNIPPETS

A. Wifi channel and Mesh creation

```
void
   MeshTest::CreateNodes ()
   {
     /*
      * Create m_ySize*m_xSize stations to
      form a grid topology
      */
     nodes.Create (m_ySize*m_xSize);
     // Configure YansWifiChannel
     YansWifiPhyHelper wifiPhy =
     YansWifiPhyHelper::Default ();
     YansWifiChannelHelper wifiChannel =
     YansWifiChannelHelper::Default ();
     wifiPhy.SetChannel (wifiChannel.
     Create ());
     /*
      * Create mesh helper and set stack
      installer to it
      * Stack installer creates all
      needed protocols
      and install them to
      * mesh point device
      */
     mesh = MeshHelper::Default ();
     if (!Mac48Address (m_root.c_str ()).
     IsBroadcast ())
       {
         mesh.SetStackInstaller (m_stack,
         "Root",
         Mac48AddressValue (Mac48Address
          (m_root.c_str ())));
       }
     else
       {
         //If root is not set, we do not use "Root
         attribute, because it
         //is specified only for 11s
         mesh.SetStackInstaller (m_stack);
       }
```

```
    if (m_chan)
      {
        mesh.SetSpreadInterfaceChannels
        (MeshHelper::SPREAD_CHANNELS);
      }
    else
      {
        mesh.SetSpreadInterfaceChannels
        (MeshHelper::ZERO_CHANNEL);
      }
    mesh.SetMacType ("RandomStart",
    TimeValue (Seconds (m_randomStart)));
    // Set number of interfaces -
    default is single-
    interface mesh point
    mesh.SetNumberOfInterfaces (m_nIfaces);
    // Install protocols and return
    container if
    MeshPointDevices
    meshDevices = mesh.Install (wifiPhy,
    nodes);
```

B. Installing internet stack

```
void
  MeshTest::InstallInternetStack ()
  {
    InternetStackHelper internetStack;
    internetStack.Install (nodes);
    Ipv4AddressHelper address;
    address.SetBase ("10.1.1.0",
    "255.255.255.0");
    interfaces = address.Assign
    (meshDevices);
  }
```

C. Installing Applications

```
void
  MeshTest::InstallApplication ()
  {
    UdpEchoServerHelper echoServer (9);
    ApplicationContainer serverApps =
    echoServer.Install (nodes.Get (0));
    serverApps.Start (Seconds (0.0));
    serverApps.Stop (Seconds (m_totalTime));
    UdpEchoClientHelper echoClient
    (interfaces.GetAddress (0), 9);
    echoClient.SetAttribute ("MaxPackets",
    UintegerValue ((uint32_t)
    (m_totalTime*(1/m_packetInterval))));
    echoClient.SetAttribute ("Interval",
    TimeValue (Seconds (m_packetInterval)));
    echoClient.SetAttribute ("PacketSize",
    UintegerValue (m_packetSize));
    ApplicationContainer clientApps =
    echoClient.Install (nodes.Get
    (m_xSize*m_ySize-1));
    clientApps.Start (Seconds (0.0));
    clientApps.Stop (Seconds (m_totalTime));
  }
```

VII. Working screenshots

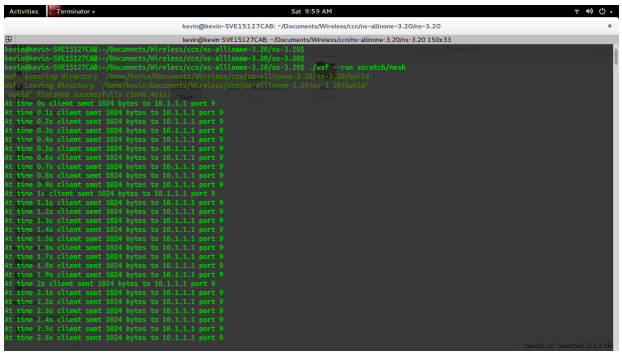

Fig. 2: Terminal screenshot of the running code

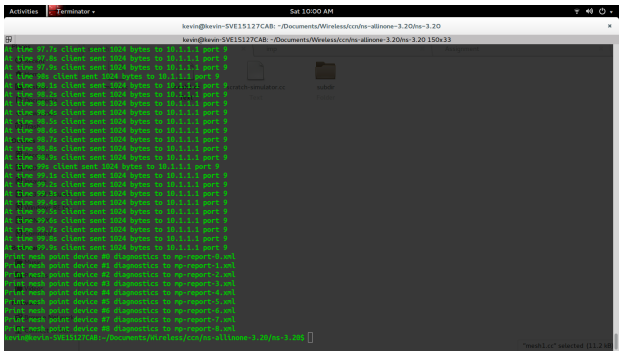

Fig. 3: Logging into pcap files

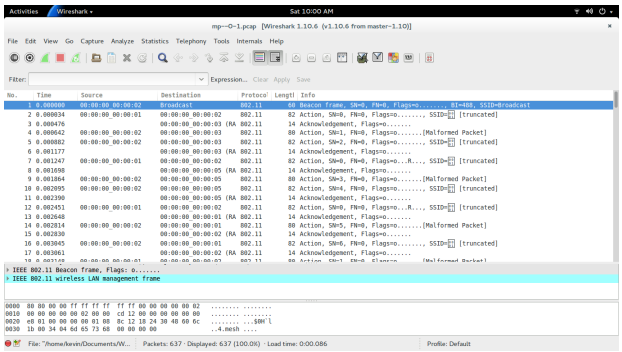

Fig. 4: pcap log of Node 0

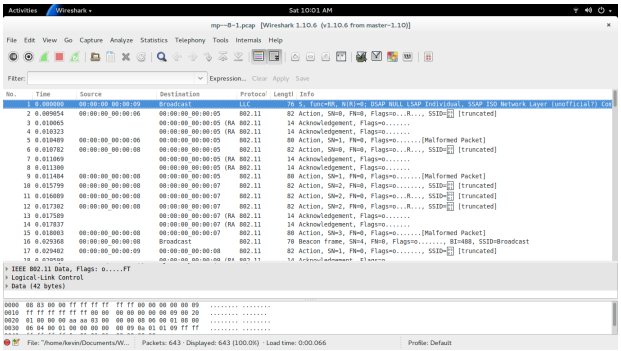

Fig. 5: pcap log of Node 8

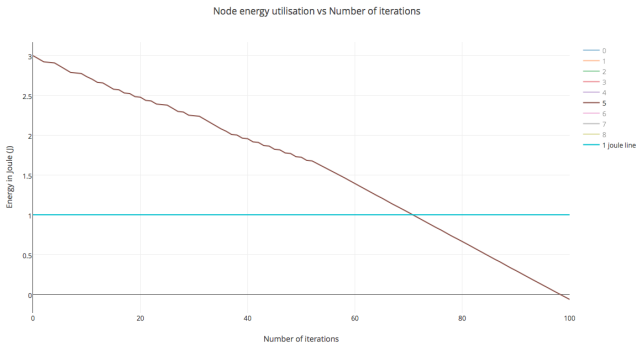

Fig. 8: Plot of Node 5 Energy : The Worst node

VIII. Plots

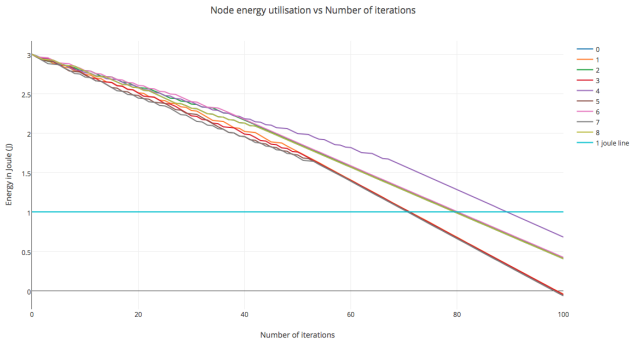

Fig. 6: A graph of Residual Enrgy v/s No.of.Iterations

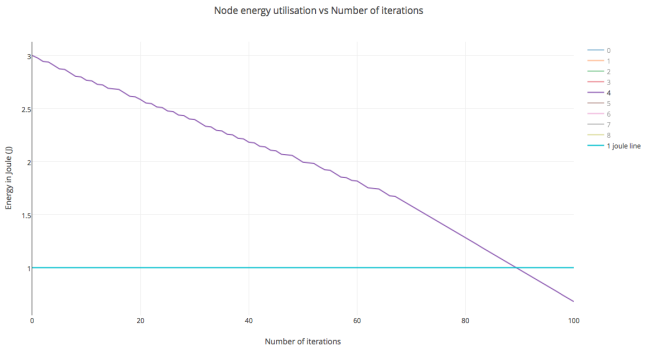

Fig. 7: Plot of Node 4 Energy : The Best node

IX. Observations

TABLE I: Time(No.of Iterations) taken for Nodes to reduce to 1/3rd of initial energy.

Node 0	80
Node 1	71
Node 2	80
Node 3	72
Node 4	90
Node 5	71
Node 6	80
Node 7	72
Node 8	79

X. Result

Energy Analysis of a 3*3 Mesh Topology was conducted. An Energy model was built based on the findings from other research work. The Energy v/s time graph for each node was plotted and the time taken by each of the nodes to reduce to 1/3rd of the initaial Energy was tabulated. It is observed that:

- The nodes closer to the source and sink get depleted faster than other nodes. Thus its important to have good backup at these nodes.
- Node 4 has the highest residual energy after a certain given time. Thus while routing more packets have to be routed through this node.
- Node 5 and 1 have the least energies and it is important to divert traffic from these nodes as much as possible and have energy backup at these nodes.

XI. References

1) *A Coverage-Aware Clustering Protocol for Wireless Sensor Networks by Bang Wang, Hock Beng Lim, Daiqin Yang and Di Ma*
2) *On the planning of wireless sensor networks: Energy-efficient clustering under the joint routing and coverage constraint by A Chamam and S.Pierre*