Koushik Sahu
118CS0597
Network Simulation Lab

Aim: To implement AODV routing protocol for nodes in a grid topology and simulate for 100sec with 10 nodes deployed at a distance of 50 meters. To understand the working of AODV as a protocol, with result analysis by pcap and .tr files and visualize the same with netanim.

Description:
AODV (Ad-hoc On-demand Distance Vector) is a loop-free routing protocol for ad-hoc networks. It is designed to be self-starting in an environment of mobile nodes, withstanding a variety of network behaviours such as node mobility, link failures and packet losses. This section describes the AODV protocol in brief; the reader is referred to for complete details of the protocol.

At each node, AODV maintains a routing table. The routing table entry for a destination contains three essential fields: a next hop node, a sequence number and a hop count. All packets destined to the destination are sent to the next hop node. The sequence number acts as a form of time-stamping, and is a measure of the freshness of a route. The hop count represents the current distance to the destination node.

Suppose we have two nodes a and b such that b is the next hop of a to some destination d. Also, suppose the sequence number and hop count of the routes to d at a and b are (seq_a, $hcnt_a$) and (seq_b, $hcnt_b$) respectively. Then the AODV protocol maintains the following property at all times:

$$(seq_a < seq_b) \vee (seq_a = seq_b \wedge hcnt_a > hcnt_b)$$

In other words, b either has a newer route to d than a, or b has a shorter route that is equally recent. Under this partial order constraint, the protocol is guaranteed to be free of routing loops.

In AODV, nodes discover routes in request-response cycles. A node requests a route to a destination by broadcasting an *RREQ* message to all its neighbours. When a node receives an RREQ message but does not have a route to the requested destination, it in turn broadcasts the RREQ message. Also, it remembers a *reverse-route* to the requesting node which can be used to forward subsequent responses to this RREQ. This process repeats until the RREQ reaches a node that has a valid route to the destination. This node (which can be the destination itself) responds with an *RREP* message. This RREP is unicast along the reverse-routes of the intermediate nodes until it reaches the original requesting node. Thus, at the end of this request-response cycle a *bidirectional* route is established between the requesting node and the destination. When a node loses connectivity to its next hop, the node invalidates its route by sending an *RERR* to all nodes that potentially received its *RREP*.

On receipt of the three AODV messages: RREQ, RREP and RERR, the nodes update the next hop, sequence number and the hop counts of their routes in such a way as to satisfy the partial order constraint mentioned above.

Code:

```
#include <iostream>
#include <cmath>
#include "ns3/aodv-module.h"
#include "ns3/core-module.h"
#include "ns3/network-module.h"
```

```
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/point-to-point-module.h"
#include "ns3/v4ping-helper.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/netanim-module.h"

using namespace ns3;

class AodvExample
{
public:
  AodvExample ();
  bool Configure (int argc, char **argv);
  void Run ();
  void Report (std::ostream & os);

private:
  uint32_t size;
  double step;
  double totalTime;
  bool pcap;
  bool printRoutes;

  NodeContainer nodes;
  NetDeviceContainer devices;
  Ipv4InterfaceContainer interfaces;

private:
  void CreateNodes ();
  void CreateDevices ();
  void InstallInternetStack ();
  void InstallApplications ();
};

int main (int argc, char **argv)
{
  AodvExample test;
  if (!test.Configure (argc, argv))
    NS_FATAL_ERROR ("Configuration failed. Aborted.");

  test.Run ();
  test.Report (std::cout);
  return 0;
}

AodvExample::AodvExample () :
  size (10),
  step (50),
  totalTime (100),
  pcap (true),
```

```
  printRoutes (true)
{
}

bool
AodvExample::Configure (int argc, char **argv)
{
  SeedManager::SetSeed (12345);
  CommandLine cmd (__FILE__);

  cmd.AddValue ("pcap", "Write PCAP traces.", pcap);
  cmd.AddValue ("printRoutes", "Print routing table dumps.", printRoutes);
  cmd.AddValue ("size", "Number of nodes.", size);
  cmd.AddValue ("time", "Simulation time, s.", totalTime);
  cmd.AddValue ("step", "Grid step, m", step);

  cmd.Parse (argc, argv);
  return true;
}

void
AodvExample::Run ()
{
  CreateNodes ();
  CreateDevices ();
  InstallInternetStack ();
  InstallApplications ();

  std::cout << "Starting simulation for " << totalTime << " s ...\n";

  Simulator::Stop (Seconds (totalTime));
  AnimationInterface anim("aodv_lab5.xml");
  Simulator::Run ();
  Simulator::Destroy ();
}

void
AodvExample::Report (std::ostream &)
{
}

void
AodvExample::CreateNodes ()
{
  std::cout << "Creating " << (unsigned)size << " nodes " << step << " m apart.\n";
  nodes.Create (size);
  for (uint32_t i = 0; i < size; ++i)
    {
      std::ostringstream os;
      os << "node-" << i;
      Names::Add (os.str (), nodes.Get (i));
```

```
    }
  MobilityHelper mobility;
  mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                                 "MinX", DoubleValue (0.0),
                                 "MinY", DoubleValue (0.0),
                                 "DeltaX", DoubleValue (step),
                                 "DeltaY", DoubleValue (0),
                                 "GridWidth", UintegerValue (size),
                                 "LayoutType", StringValue ("RowFirst"));
  mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
  mobility.Install (nodes);
}

void
AodvExample::CreateDevices ()
{
  WifiMacHelper wifiMac;
  wifiMac.SetType ("ns3::AdhocWifiMac");
  YansWifiPhyHelper wifiPhy;
  YansWifiChannelHelper wifiChannel = YansWifiChannelHelper::Default ();
  wifiPhy.SetChannel (wifiChannel.Create ());
  WifiHelper wifi;
  wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager", "DataMode", StringValue
("OfdmRate6Mbps"), "RtsCtsThreshold", UintegerValue (0));
  devices = wifi.Install (wifiPhy, wifiMac, nodes);

  if (pcap)
    {
      wifiPhy.EnablePcapAll (std::string ("assign5"));
    }

  AsciiTraceHelper ascii;
  wifiPhy.EnableAsciiAll(ascii.CreateFileStream("assign5.tr"));
}

void
AodvExample::InstallInternetStack ()
{
  AodvHelper aodv;
  InternetStackHelper stack;
  stack.SetRoutingHelper (aodv);
  stack.Install (nodes);
  Ipv4AddressHelper address;
  address.SetBase ("10.0.0.0", "255.0.0.0");
  interfaces = address.Assign (devices);

  if (printRoutes)
    {
      Ptr<OutputStreamWrapper> routingStream = Create<OutputStreamWrapper>
("assign5.routes", std::ios::out);
      aodv.PrintRoutingTableAllAt (Seconds (8), routingStream);
```

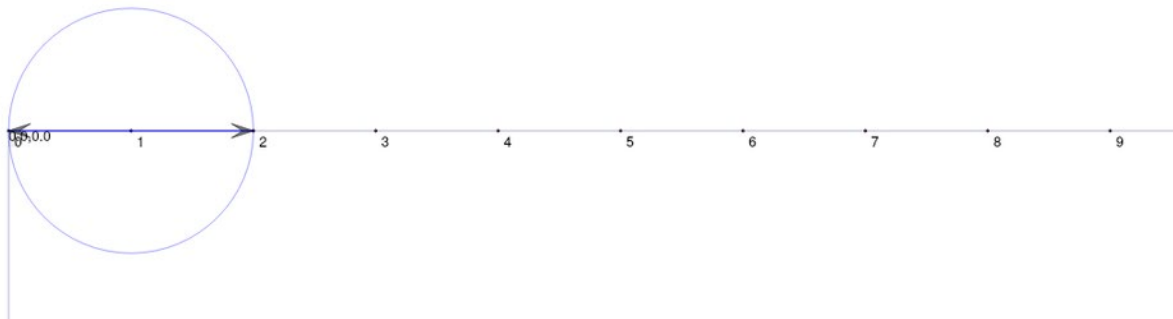
```
  }
}

void
AodvExample::InstallApplications ()
{
  V4PingHelper ping (interfaces.GetAddress (size - 1));
  ping.SetAttribute ("Verbose", BooleanValue (true));

  ApplicationContainer p = ping.Install (nodes.Get (0));
  p.Start (Seconds (0));
  p.Stop (Seconds (totalTime) - Seconds (0.001));

  Ptr<Node> node = nodes.Get (size/2);
  Ptr<MobilityModel> mob = node->GetObject<MobilityModel> ();
  Simulator::Schedule (Seconds (totalTime/3), &MobilityModel::SetPosition, mob, Vector (1e5,
1e5, 1e5));
}
```

Results:

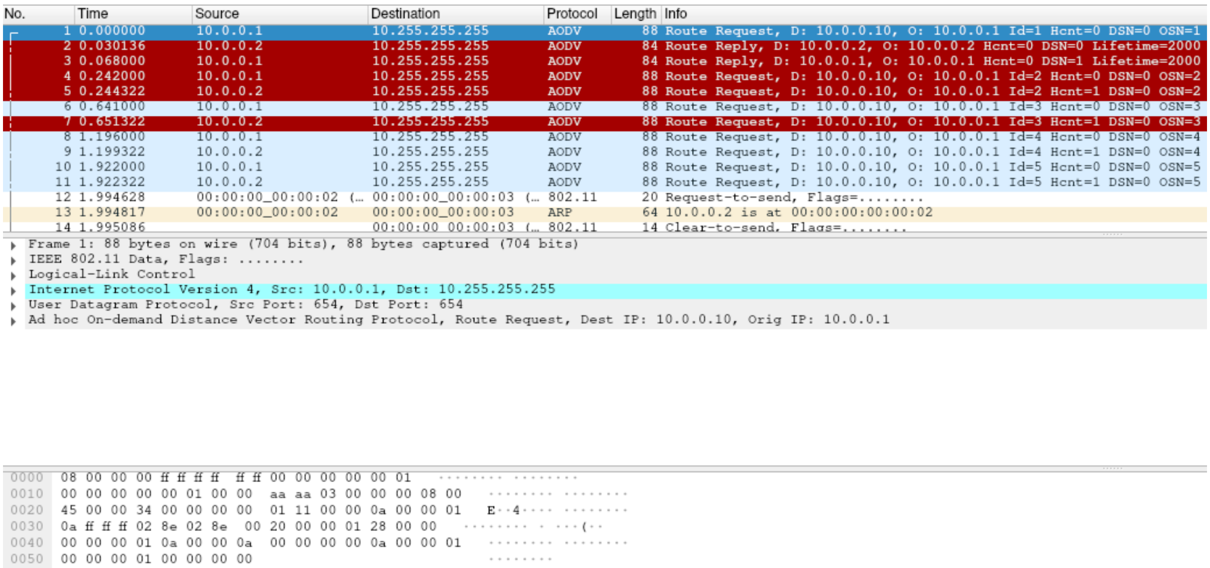
Terminal output:

```
Creating 10 nodes 50 m apart.
Starting simulation for 100 s ...
PING 10.0.0.10 - 56 bytes of data - 84 bytes including ICMP and IPv4 headers.
64 bytes from 10.0.0.10: icmp_seq=0 ttl=56 time=+2056.49ms
64 bytes from 10.0.0.10: icmp_seq=1 ttl=56 time=+1058.19ms
64 bytes from 10.0.0.10: icmp_seq=2 ttl=56 time=+59.8094ms
64 bytes from 10.0.0.10: icmp_seq=3 ttl=56 time=+7.39202ms
64 bytes from 10.0.0.10: icmp_seq=4 ttl=56 time=+7.33802ms
64 bytes from 10.0.0.10: icmp_seq=5 ttl=56 time=+7.31102ms
64 bytes from 10.0.0.10: icmp_seq=6 ttl=56 time=+7.29302ms
64 bytes from 10.0.0.10: icmp_seq=7 ttl=56 time=+7.36502ms
64 bytes from 10.0.0.10: icmp_seq=8 ttl=56 time=+7.37402ms
64 bytes from 10.0.0.10: icmp_seq=9 ttl=56 time=+7.39202ms
64 bytes from 10.0.0.10: icmp_seq=10 ttl=56 time=+7.34702ms
64 bytes from 10.0.0.10: icmp_seq=11 ttl=56 time=+7.32002ms
64 bytes from 10.0.0.10: icmp_seq=12 ttl=56 time=+7.26602ms
64 bytes from 10.0.0.10: icmp_seq=13 ttl=56 time=+7.32002ms
64 bytes from 10.0.0.10: icmp_seq=14 ttl=56 time=+7.32002ms
64 bytes from 10.0.0.10: icmp_seq=15 ttl=56 time=+7.35602ms
64 bytes from 10.0.0.10: icmp_seq=16 ttl=56 time=+7.32002ms
64 bytes from 10.0.0.10: icmp_seq=17 ttl=56 time=+7.28402ms
64 bytes from 10.0.0.10: icmp_seq=18 ttl=56 time=+7.28402ms
64 bytes from 10.0.0.10: icmp_seq=19 ttl=56 time=+7.32902ms
64 bytes from 10.0.0.10: icmp_seq=20 ttl=56 time=+7.33802ms
64 bytes from 10.0.0.10: icmp_seq=21 ttl=56 time=+7.31102ms
64 bytes from 10.0.0.10: icmp_seq=22 ttl=56 time=+7.31102ms
64 bytes from 10.0.0.10: icmp_seq=23 ttl=56 time=+7.32002ms
64 bytes from 10.0.0.10: icmp_seq=24 ttl=56 time=+7.34702ms
64 bytes from 10.0.0.10: icmp_seq=25 ttl=56 time=+7.34702ms
64 bytes from 10.0.0.10: icmp_seq=26 ttl=56 time=+7.28402ms
64 bytes from 10.0.0.10: icmp_seq=27 ttl=56 time=+7.29302ms
64 bytes from 10.0.0.10: icmp_seq=28 ttl=56 time=+7.30202ms
64 bytes from 10.0.0.10: icmp_seq=29 ttl=56 time=+7.29302ms
64 bytes from 10.0.0.10: icmp_seq=30 ttl=56 time=+7.28402ms
64 bytes from 10.0.0.10: icmp_seq=31 ttl=56 time=+7.37402ms
64 bytes from 10.0.0.10: icmp_seq=32 ttl=56 time=+7.31102ms
64 bytes from 10.0.0.10: icmp_seq=33 ttl=56 time=+7.36502ms
--- 10.0.0.10 ping statistics ---
100 packets transmitted, 34 received, 66% packet loss, time +1e+05ms
rtt min/avg/max/mdev = 7/99.71/2056/389.8 ms
```

Netanim output:



Wireshark output for first node:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	10.0.0.1	10.255.255.255	AODV	88	Route Request, D: 10.0.0.10, O: 10.0.0.1 Id=1 Hcnt=0 DSN=0 OSN=1
2	0.030136	10.0.0.2	10.255.255.255	AODV	84	Route Reply, D: 10.0.0.2, O: 10.0.0.2 Hcnt=0 DSN=0 Lifetime=2000
3	0.068000	10.0.0.1	10.255.255.255	AODV	84	Route Reply, D: 10.0.0.1, O: 10.0.0.1 Hcnt=0 DSN=1 Lifetime=2000
4	0.242000	10.0.0.1	10.255.255.255	AODV	88	Route Request, D: 10.0.0.10, O: 10.0.0.1 Id=2 Hcnt=0 DSN=0 OSN=2
5	0.244322	10.0.0.2	10.255.255.255	AODV	88	Route Request, D: 10.0.0.10, O: 10.0.0.1 Id=2 Hcnt=1 DSN=0 OSN=2
6	0.641000	10.0.0.1	10.255.255.255	AODV	88	Route Request, D: 10.0.0.10, O: 10.0.0.1 Id=3 Hcnt=0 DSN=0 OSN=3
7	0.651322	10.0.0.2	10.255.255.255	AODV	88	Route Request, D: 10.0.0.10, O: 10.0.0.1 Id=3 Hcnt=1 DSN=0 OSN=3
8	1.196000	10.0.0.1	10.255.255.255	AODV	88	Route Request, D: 10.0.0.10, O: 10.0.0.1 Id=4 Hcnt=0 DSN=0 OSN=4
9	1.199322	10.0.0.2	10.255.255.255	AODV	88	Route Request, D: 10.0.0.10, O: 10.0.0.1 Id=4 Hcnt=1 DSN=0 OSN=4
10	1.922000	10.0.0.1	10.255.255.255	AODV	88	Route Request, D: 10.0.0.10, O: 10.0.0.1 Id=5 Hcnt=0 DSN=0 OSN=5
11	1.922322	10.0.0.2	10.255.255.255	AODV	88	Route Request, D: 10.0.0.10, O: 10.0.0.1 Id=5 Hcnt=1 DSN=0 OSN=5
12	1.994628	00:00:00_00:00:02 (…	00:00:00_00:00:03 (…	802.11	20	Request-to-send, Flags=........
13	1.994817	00:00:00_00:00:02	00:00:00_00:00:03	ARP	64	10.0.0.2 is at 00:00:00:00:00:02
14	1.995086		00:00:00_00:00:03 (…	802.11	14	Clear-to-send, Flags=........

```
Frame 1: 88 bytes on wire (704 bits), 88 bytes captured (704 bits)
IEEE 802.11 Data, Flags: ........
Logical-Link Control
Internet Protocol Version 4, Src: 10.0.0.1, Dst: 10.255.255.255
User Datagram Protocol, Src Port: 654, Dst Port: 654
Ad hoc On-demand Distance Vector Routing Protocol, Route Request, Dest IP: 10.0.0.10, Orig IP: 10.0.0.1
```

```
0000  08 00 00 00 ff ff ff ff  ff ff 00 00 00 00 00 01   ........ ........
0010  00 00 00 00 00 01 00 00  aa aa 03 00 00 00 08 00   ........ ........
0020  45 00 00 34 00 00 00 00  01 11 00 00 0a 00 00 01   E..4.... ........
0030  0a ff ff ff 02 8e 02 8e  00 20 00 00 01 28 00 00   ........ . ...(..
0040  00 00 00 01 0a 00 00 0a  00 00 00 00 0a 00 00 01   ........ ........
0050  00 00 00 01 00 00 00 00                            ........
```

Conclusion:

With the transmission of 100 packets we observe that only 34 are receive which means there is a 66% packet loss. The protocol is inefficient but is easy to setup requiring not infrastructure.