MOBILE NETWORK PERVASIVE COMPUTING (MANET)

Mochammad Zen Samsono Hadi, ST. MSc. Ph.D

Discussion

- MANET (Mobile Adhoc Network)
- Performance Evaluation

MANET

- MANETs are formed dynamically by an autonomous system of mobile nodes that are connected via wireless links.
- Mobile nodes are free to move randomly, network topology changes frequently
- May operate as standalone fashion or also can be connected to the larger internet

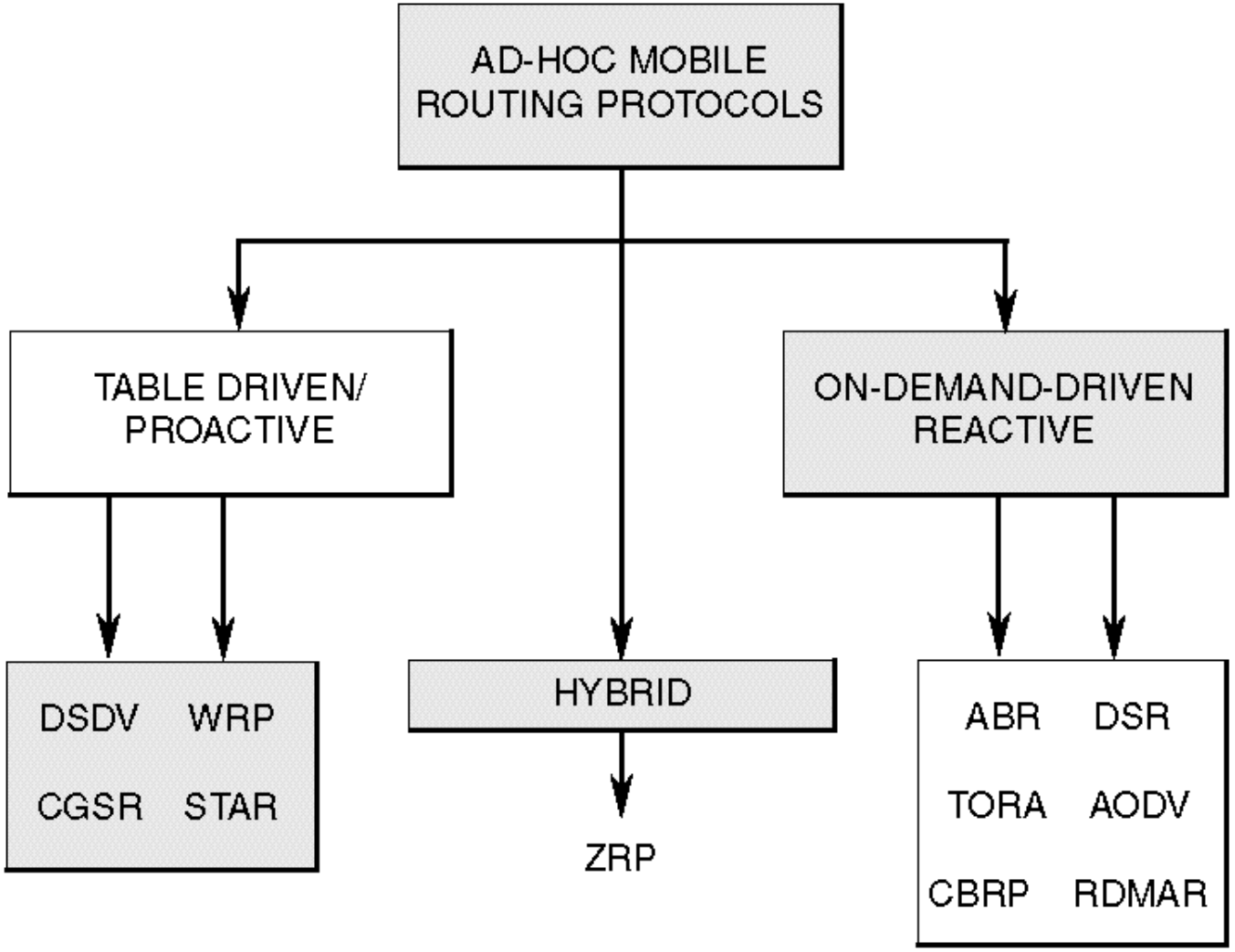
Applications

- Tactical networks
  - Military communication, automated battlefields
- Emergency Services
  - Search and rescue operations
  - Disaster recovery: Earthquakes, hurricanes
- Educational
  - Virtual classrooms or conference rooms
  - Set up ad hoc communication during conferences, meeting, or lectures
- Home and Entertainment
  - Home/office wireless networking
  - Personal Area Network
  - Multiuser games
  - Outdoor internet access

Challenges

- Infrastructure less
  - Brings new network designing challenges
- Dynamically changing topologies
  - Cause route changes, frequent network partitions and packet loss
- Physical layer limitations
  - Limited wireless range
  - Packet loss during transmission
  - Broadcast nature of the communication
- Limitations of mobile nodes
  - Short battery life
  - Limited capacities
- Network security

Categorization of Ad Hoc Routing Protocols

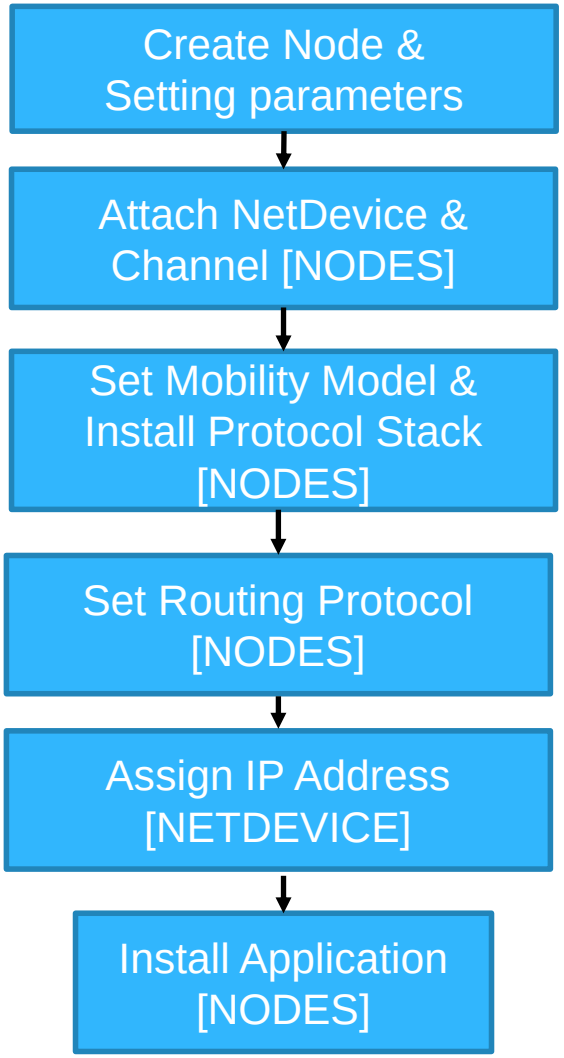


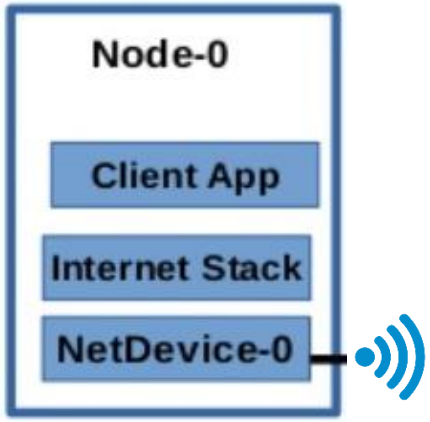
MANET in ns-3

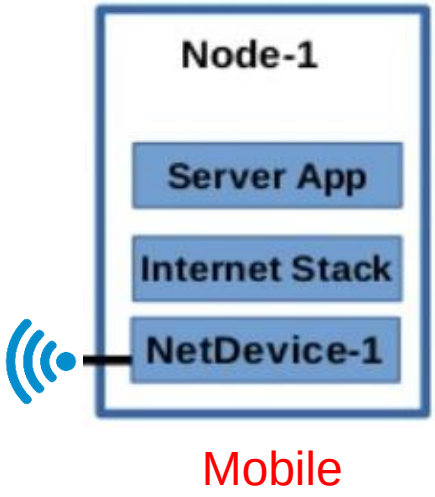
- Example of manet:
  - ~/ns-allinone-3.29/ns-3.29/examples/routing/manet-routing-compare.cc
- Parameters

No	Parameters	Remarks
1	Simulation time	200 seconds
2	Number of nodes	50
3	Mobility model	RandomWaypointMobilityModel
4	Speed	20 m/s and no pause time
5	Region	300 x 1500 m
6	Wifi model	Ad hoc mode with a 2 Mb/s rate (IEEE 802.11b)
7	Propagation model	Friis loss model
8	Transmit power	7.5 dBm
9	Routing protocol	DSDV, OLSR, DSR, **AODV (default)**
10	Number of source / sink	10 source / sink data pairs sending UDP data at an application rate of 2.048 Kb/s each. This is typically done at a rate of 4 64-byte packets per second.
11	Application data	It is started at a random time between 50 and 51 seconds and continues to the end of the simulation

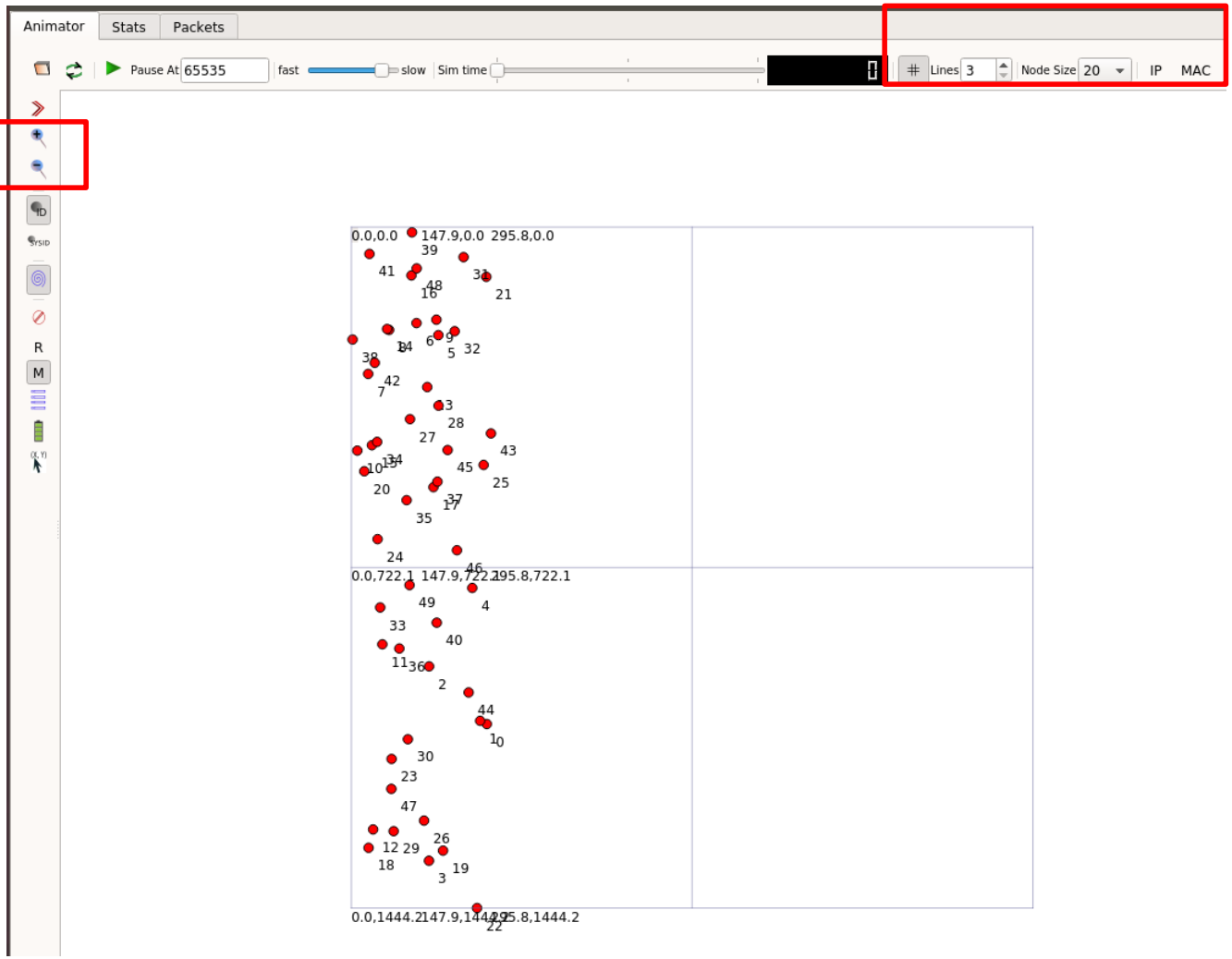
Flowchart


Create Node &
Setting parameters
Attach NetDevice &
Channel [NODES]
Set Mobility Model &
Install Protocol Stack
[NODES]
Set Routing Protocol
[NODES]
Assign IP Address
[NETDEVICE]
Install Application
[NODES]


Node-0
Client App
Internet Stack
NetDevice-0
Mobile


Node-1
Server App
Internet Stack
NetDevice-1
Mobile

Topologi Jaringan Wireless



1. Program: Command setup

```
std::string
RoutingExperiment::CommandSetup (int argc, char **argv)
{
  CommandLine cmd;
  cmd.AddValue ("CSVfileName", "The name of the CSV output file name", m_CSVfileName);
  cmd.AddValue ("traceMobility", "Enable mobility tracing", m_traceMobility);
  cmd.AddValue ("protocol", "1=OLSR;2=AODV;3=DSDV;4=DSR", m_protocol);
  cmd.Parse (argc, argv);
  return m_CSVfileName;
}
```

- Cmd with command:

  ./waf --run “scratch/manet-routing-compare --protocol=1”

2. Program: Parameters

```
int nSinks = 10;
double txp = 7.5;

int nWifis = 50;

double TotalTime = 200.0;
std::string rate ("2048bps");
std::string phyMode ("DsssRate11Mbps");
std::string tr_name ("manet-routing-compare");
int nodeSpeed = 20; //in m/s
int nodePause = 0; //in s
m_protocolName = "protocol";

Config::SetDefault  ("ns3::OnOffApplication::PacketSize",StringValue ("64"));
Config::SetDefault ("ns3::OnOffApplication::DataRate",  StringValue (rate));
```

- Some parameters are determined in function:
  - ***void RoutingExperiment::Run***

3. Program: Create Node & WiFi

```
NodeContainer adhocNodes;
adhocNodes.Create (nWifis);
```

Create nodes

```
// setting up wifi phy and channel using helpers
WifiHelper wifi;
wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
```

NetDevice

Channel & PHY

```
YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel;
wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
wifiPhy.SetChannel (wifiChannel.Create ());
```

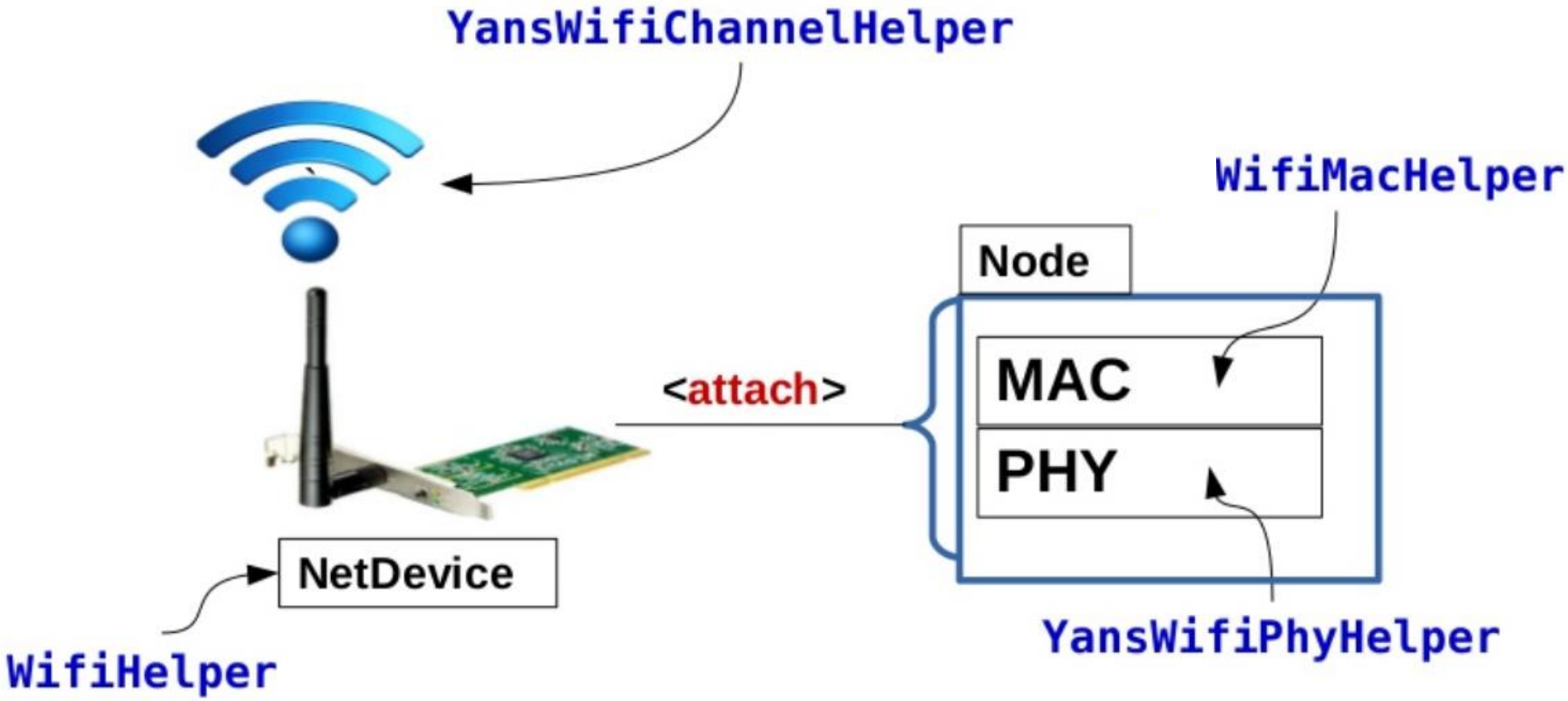
MAC protocol

```
// Add a mac and disable rate control
WifiMacHelper wifiMac;
wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
                              "DataMode",StringValue (phyMode),
                              "ControlMode",StringValue (phyMode));

wifiPhy.Set ("TxPowerStart",DoubleValue (txp));
wifiPhy.Set ("TxPowerEnd", DoubleValue (txp));
```

MAC protocol

```
wifiMac.SetType ("ns3::AdhocWifiMac");
NetDeviceContainer adhocDevices = wifi.Install (wifiPhy, wifiMac, adhocNodes);
```

Configure WiFi NetDevice



Characteristics of Wifi

- WIFI_PHY_STANDARD_80211b

WIFI_PHY_STANDARD_80211a	OFDM PHY for the 5 GHz band (Clause 17)
WIFI_PHY_STANDARD_80211b	DSSS PHY (Clause 15) and HR/DSSS PHY (Clause 18)
WIFI_PHY_STANDARD_80211g	ERP-OFDM PHY (Clause 19, Section 19.5)

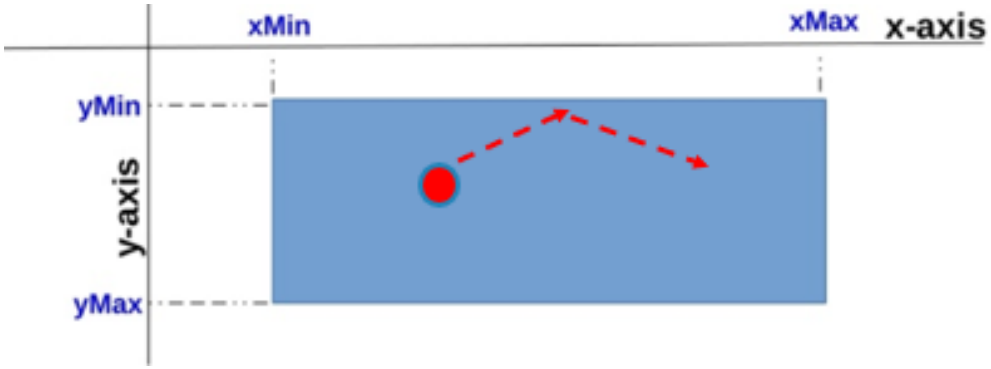
- ConstantSpeedPropagationDelayModel
  - Calculate the propagation delay between source and destination
- ConstantRateWifiManager
  - It uses constant rates for data and control transmissions and uses always the same transmission rate for every packet sent.
- WifiMac
  - AdhocWifiMac : Infrastructure less network
  - ApWifiMac : Access point Node MAC
  - StaWifiMac : Station Node MAC

4. Program: Set Mobility

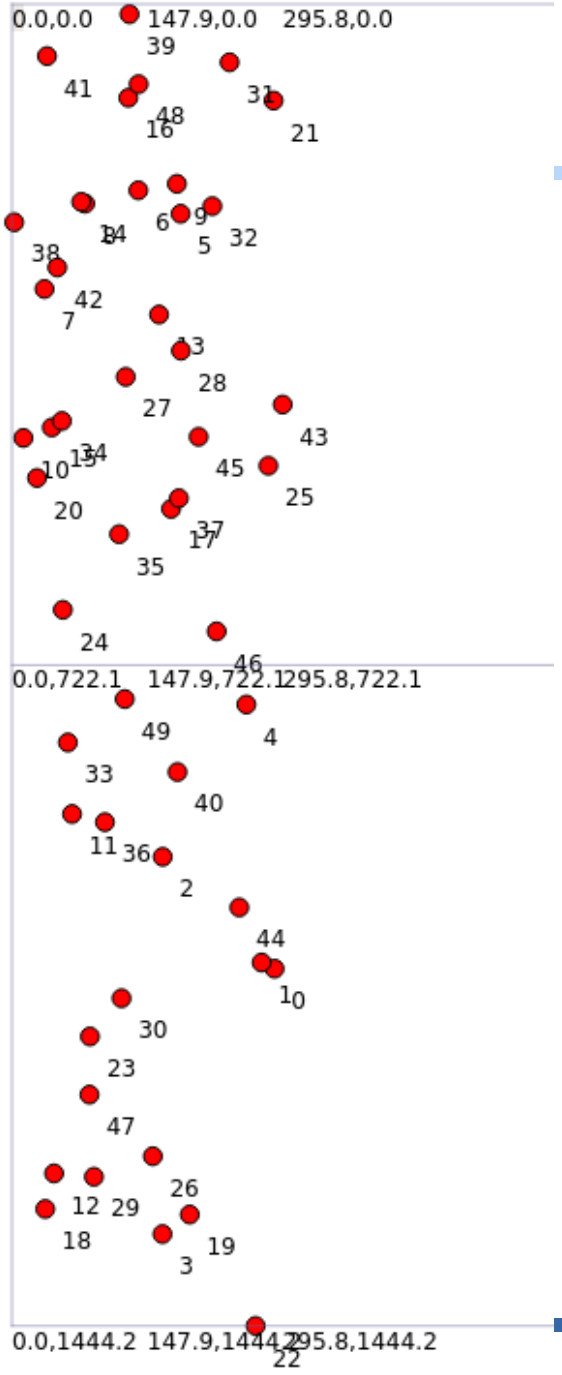
```
MobilityHelper mobilityAdhoc;
int64_t streamIndex = 0; // used to get consistent mobility across scenarios

ObjectFactory pos;
pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
pos.Set ("X", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=300.0]"));
pos.Set ("Y", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=1500.0]"));

Ptr<PositionAllocator> taPositionAlloc = pos.Create ()->GetObject<PositionAllocator> ();
streamIndex += taPositionAlloc->AssignStreams (streamIndex);

std::stringstream ssSpeed;
ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max=" << nodeSpeed << "]";
std::stringstream ssPause;
ssPause << "ns3::ConstantRandomVariable[Constant=" << nodePause << "]";
mobilityAdhoc.SetMobilityModel ("ns3::RandomWaypointMobilityModel",
                                "Speed", StringValue (ssSpeed.str ()),
                                "Pause", StringValue (ssPause.str ()),
                                "PositionAllocator", PointerValue (taPositionAlloc));
mobilityAdhoc.SetPositionAllocator (taPositionAlloc);
mobilityAdhoc.Install (adhocNodes);
```

Mobility Model

- RandomWaypointMobilityModel
- Set random position within a rectangle according to a pair of random variables (x, y) to place nodes
- Each object starts by pausing at time zero for the duration governed by the random variable "Pause". After pausing, the object will pick a new waypoint (via the **PositionAllocator**) and a new random speed via the random variable "Speed", and will begin moving towards the waypoint at a constant speed. When it reaches the destination, the process starts over (by pausing).



Nodes Position



5. Program: Set Routing Protocol

```
AodvHelper aodv;
OlsrHelper olsr;
DsdvHelper dsdv;
DsrHelper dsr;
DsrMainHelper dsrMain;
Ipv4ListRoutingHelper list;
InternetStackHelper internet;
```

```
switch (m_protocol)
  {
  case 1:
    list.Add (olsr, 100);
    m_protocolName = "OLSR";
    break;
  case 2:
    list.Add (aodv, 100);
    m_protocolName = "AODV";
    break;
  case 3:
    list.Add (dsdv, 100);
    m_protocolName = "DSDV";
    break;
  case 4:
    m_protocolName = "DSR";
    break;
  default:
    NS_FATAL_ERROR ("No such protocol:" << m_protocol);
  }
```

Routing Protocol OLSR, AODV and DSDV are default

5. Program: Routing Protocol & IP address

```
if (m_protocol < 4)
  {
    internet.SetRoutingHelper (list);
    internet.Install (adhocNodes);
  }
else if (m_protocol == 4)
  {
    internet.Install (adhocNodes);
    dsrMain.Install (dsr, adhocNodes);
  }

NS_LOG_INFO ("assigning ip address");
```

Set to [NODES]

Set to [NODES]

IPv4 Address

```
Ipv4AddressHelper addressAdhoc;
addressAdhoc.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer adhocInterfaces;
adhocInterfaces = addressAdhoc.Assign (adhocDevices);
```

6. Application OnOff

Set OnOff Application

```
OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
onoff1.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Constant=1.0]"));
onoff1.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Constant=0.0]"));
```

```
for (int i = 0; i < nSinks; i++)
  {
    Ptr<Socket> sink = SetupPacketReceive (adhocInterfaces.GetAddress (i), adhocNodes.Get (i));

    AddressValue remoteAddress (InetSocketAddress (adhocInterfaces.GetAddress (i), port));
    onoff1.SetAttribute ("Remote", remoteAddress);

    Ptr<UniformRandomVariable> var = CreateObject<UniformRandomVariable> ();
    ApplicationContainer temp = onoff1.Install (adhocNodes.Get (i + nSinks));
    temp.Start (Seconds (var->GetValue (100.0,101.0)));
    temp.Stop (Seconds (TotalTime));
  }
```

Process send – receive between nodes

OnfOffApplication

- Generate traffic to a single destination according to an OnOff pattern.
- This traffic generator follows an On/Off pattern:
  after **StartApplication** is called, "On" and "Off" states alternate.
- The duration of each of these states is determined with the onTime and the offTime random variables.
- During the "Off" state, no traffic is generated.
- During the "On" state, cbr traffic is generated.
- This cbr traffic is characterized by the specified "data rate" (bps) and "packet size" (byte).

```
std::string rate ("2048bps");

Config::SetDefault  ("ns3::OnOffApplication::PacketSize",StringValue ("64"));
Config::SetDefault ("ns3::OnOffApplication::DataRate",  StringValue (rate));
```

7. Simulation Run

```
temp.Start (Seconds (var->GetValue (100.0,101.0)));
temp.Stop (Seconds (TotalTime));
```

```
root@zenhadi-VirtualBox:~/ns-allinone-3.29/ns-3.29# ./waf --run scratch/manet-ro
uting-compare
Waf: Entering directory `/home/zenhadi/ns-allinone-3.29/ns-3.29/build'
[2576/2633] Compiling scratch/manet-routing-compare.cc
[2593/2633] Linking build/scratch/manet-routing-compare
Waf: Leaving directory `/home/zenhadi/ns-allinone-3.29/ns-3.29/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (32.421s)
100.348 4 received one packet from 10.1.1.15
100.596 4 received one packet from 10.1.1.15
100.846 4 received one packet from 10.1.1.15
100.872 3 received one packet from 10.1.1.14
100.873 3 received one packet from 10.1.1.14
101.06 3 received one packet from 10.1.1.14
101.079 9 received one packet from 10.1.1.20
101.082 9 received one packet from 10.1.1.20
101.09 5 received one packet from 10.1.1.16
```

8. Performance Analysis

- FlowMonitor: bitrates (bps), packet loss, delay

```
Ptr<FlowMonitor> flowmon;
FlowMonitorHelper flowmonHelper;
flowmon = flowmonHelper.InstallAll ();
```

- CSV file: Received rate and packets received

```
//blank out the last output file and write the column headers
std::ofstream out (CSVfileName.c_str ());
out << "SimulationSecond," <<
"ReceiveRate," <<
"PacketsReceived," <<
"NumberOfSinks," <<
"RoutingProtocol," <<
"TransmissionPower" <<
std::endl;
out.close ();
```

TASK

- Change to OLSR protocol
- What's the different with AODV protocol in the performance?