MOBILE NETWORK PERVASIVE COMPUTING (WIRELESS NETWORK)

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

PASCA SARJANA PENS

TOPIK PEMBAHASAN

- Wireless Network
- Mobility Network

Application (Traffic Generator)

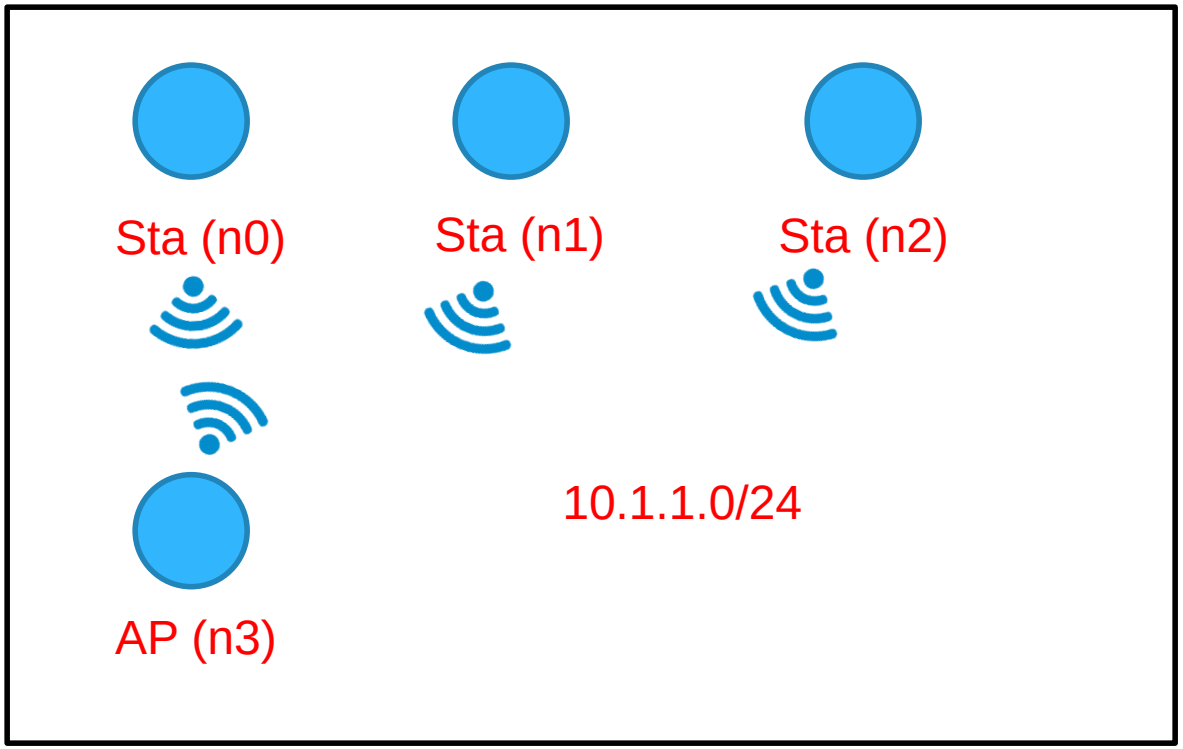
- Bulk-Send – Send data as fast as possible
  - BulkSendApplication
- On-Off – On off pattern
  - OnOffApplication
- Udp-Server – Receive UDP packets
  - UdpServer, UdpServerHelper
- UDP-Client – UDP packet with seq no and time stamp
  - UdpClient, UdpClientHelper
- V4ping – Sends one ICMP ECHO request, report the RTT ping6
  - V4ping, V4pingHelper

Main Program Structure

- Include HEADER files
- Include NAMESPACE
- Enable/disable LOGGING
- Create NODE
- Configure TOPOLOGY for Nodes
- Set up INTERNET STACK
- Set up APPLICATION
- Run SIMULATION

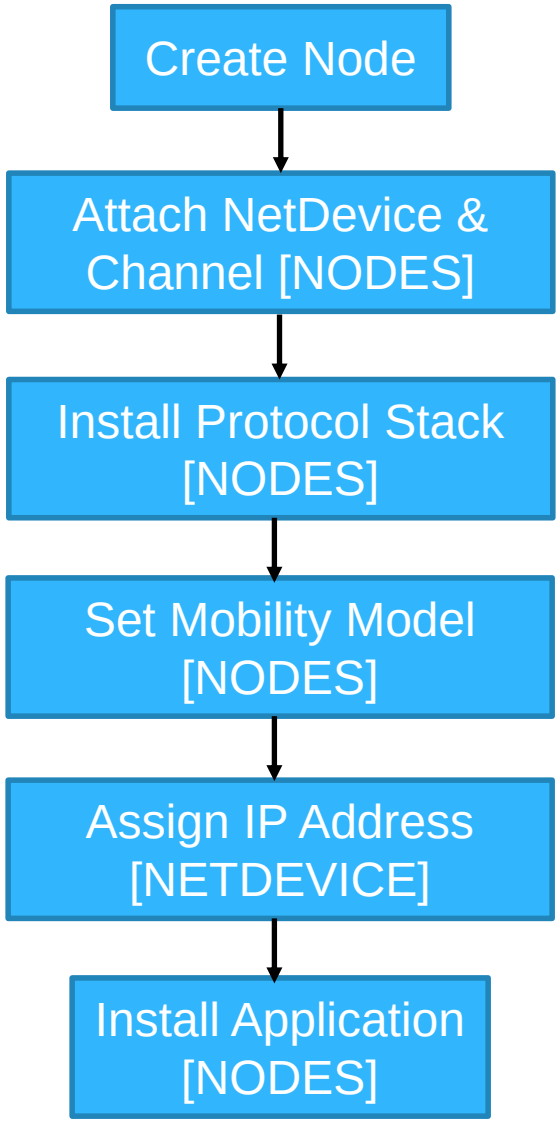
Wireless Network

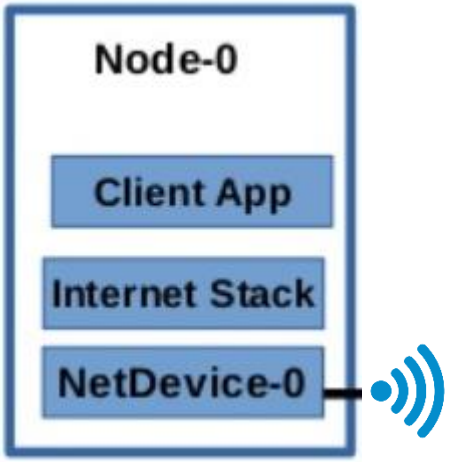
Topologi Jaringan Wireless



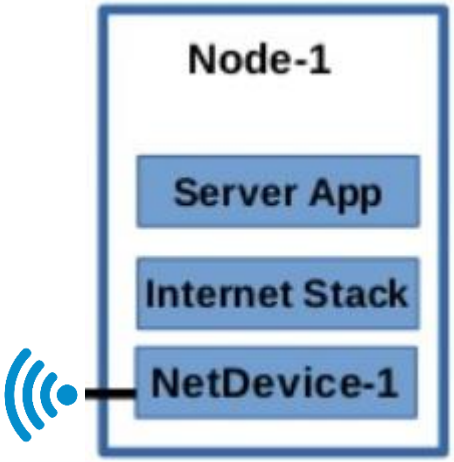
Sta : mobile
AP : fixed position

Flowchart





Mobile



Fixed

Classes

- NodeContainer
- YansWifiChannelHelper
- YansWifiPhyHelper
- WifiHelper
- NetDeviceContainer
- MobilityHelper
- InternetStackHelper
- Ipv4AddressHelper
- Ipv4InterfaceContainer
- UdpEchoServerHelper
- UdpEchoClientHelper
- ApplicationContainer

1. Program: Loading Module

```
/* Ilustrasi topologi
 *
 * Wifi 10.1.3.0
 *              AP
 *     *   *   *   *
 *     |   |   |   |
 *    n3   n2  n1  n0
 */

 #include "ns3/core-module.h"
 #include "ns3/point-to-point-module.h"
 #include "ns3/network-module.h"
 #include "ns3/applications-module.h"
 #include "ns3/wifi-module.h"
 #include "ns3/mobility-module.h"
 #include "ns3/csma-module.h"
 #include "ns3/internet-module.h"
 #include "ns3/netanim-module.h" // entered for animation configuration and
output file
```

2. Program: Parameters

```
bool verbose = true;
uint32_t nWifi = 3;
bool tracing = false;

CommandLine cmd;

cmd.AddValue ("nWifi", "Number of wifi STA devices", nWifi);
cmd.AddValue ("verbose", "Tell echo applications to log if true", verbose);
cmd.AddValue ("tracing", "Enable pcap tracing", tracing);

cmd.Parse (argc,argv);

if (verbose)
{
  LogComponentEnable ("UdpEchoClientApplication", LOG_LEVEL_INFO);
  LogComponentEnable ("UdpEchoServerApplication", LOG_LEVEL_INFO);
}
```

set no of nodes: 3

- Cmd with command:

  ./waf --run “scratch/mythird –nWifi=10”

3. Program: Create Node & WiFi

```
NodeContainer wifiStaNodes;
wifiStaNodes.Create (nWifi);
```

Create station nodes (Wifi-client)

```
NodeContainer wifiApNode;
wifiApNode.Create(1);
```

Create Access Point (AP) node

```
YansWifiChannelHelper channel = YansWifiChannelHelper::Default ();
YansWifiPhyHelper phy = YansWifiPhyHelper::Default ();
phy.SetChannel (channel.Create ());
```

Channel

PHY

```
WifiHelper wifi;
wifi.SetRemoteStationManager ("ns3::AarfWifiManager");
```

NetDevice

MAC protocol on clients

```
   WifiMacHelper mac;
   Ssid ssid = Ssid ("ns-3-ssid");
   mac.SetType ("ns3::StaWifiMac", "Ssid", SsidValue (ssid), "ActiveProbing",
BooleanValue (false));

   NetDeviceContainer staDevices;
   staDevices = wifi.Install (phy, mac, wifiStaNodes);
```

```
mac.SetType ("ns3::ApWifiMac", "Ssid", SsidValue (ssid));

NetDeviceContainer apDevices;
apDevices = wifi.Install (phy, mac, wifiApNode);
```

MAC protocol on AP

Yans WiFi

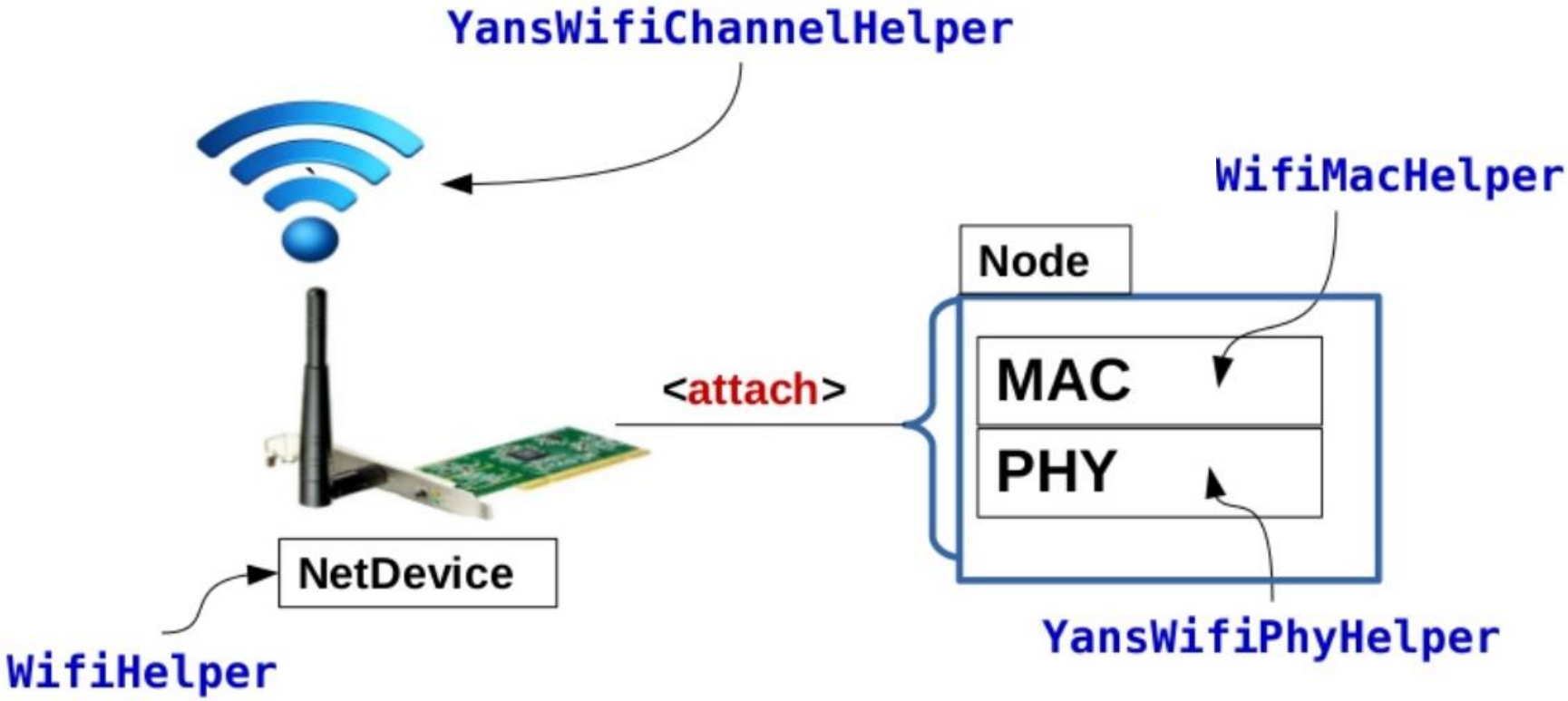
- Yans: Yet Another Network Simulator
- Based on IEEE 802.11p
  - Standard dedicated to vehicular communications (VANET)
  - WLAN operating in the 2.4 GHz and 5 GHz bands and OFDM wireless MAC/PHY transmission
  - It computes the channel transmission properties, such as signal-to-noise ratio (SNR) and packet error rate (PER).
  - It supports propagation models and model phenomena like path loss (Friis-deterministic), shadowing (Rayleigh-statistical), and small-scale fading.
  - It transmits a packet to the “channel level” with a specific txPower and has enough reception power typically -104 dBm.
  - It determines characteristic elements of the transmission: coding rate, modulation, frequency, etc.

Yans WiFi

- Propagation channel model simulation parameters

Path loss and shadowing	Abbas *et al*. Shadow-fading model
Small-scale fading	Acosta-Marum V2V Urban Canyon Oncoming or ITU Vehicular A
Packet size	256 bytes
Interpacket interval	0.1 s
Data rate	6 Mbps (QPSK, R = 1/2), 18 Mbps (16QAM, R = 3/4)
Transmission power	+20 dBm
Max. distance between N1 and N2	500 m
Simulation time	90 s

Configure WiFi NetDevice



NetDevice and Channel

- YansWifiChannelHelper – Channel
- YansWifiPhyHelper – PHY
  - It will share the same wireless medium and can communicate and interfere

```
phy.SetChannel (channel.Create ());
```

- WifiHelper – NetDevice

- It uses Install() method to attach NetDevice with Node
- It will return an object to NetDeviceContainer

Configure WiFi NetDevice

- **WifiChannel & WifiPhy** abstract class
  - **YansWifiChannel**
    - We use Helper [**YansWifiChannelHelper**]
    - Set Channel related attributes
      - [*channel switch delay, energy of received signal,etc]*
  - **YansWifiPhy**
    - We use Helper [**YansWifiPhyHelper**]
    - Set PHY Layer related attributes
      - [*propogation delay*]

Set CHANNEL to PHY

Class YansWifiPhy

```
void SetChannel(Ptr<YansWifiChannel>)-
```

Configure WiFi NetDevice

- WifiHelper

  WifiHelper wifi;

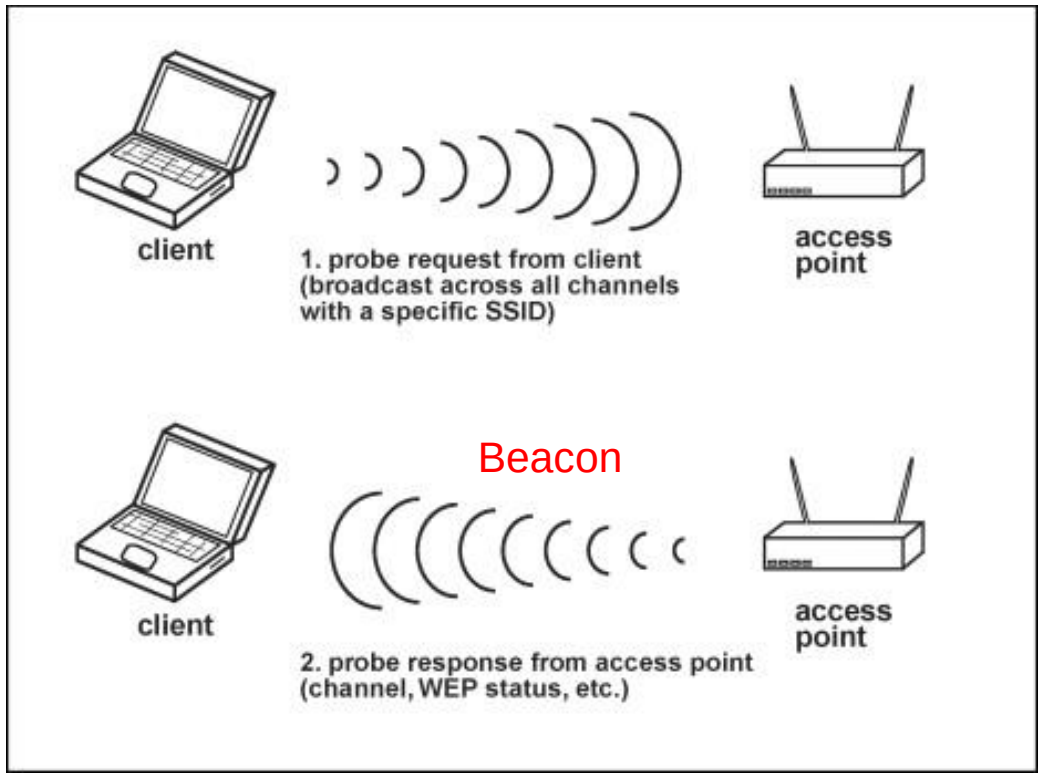
  wifi.SetRemoteStationManager (“ns3::AarfWifiManager”);

- SetRemoteStationManager
  - This method tells the helper the type of rate control algorithm i.e. AARF (adaptive auto rate fallback)
  - It will cover all provide multi-rate capabilities on the different physical (PHY) layers for the IEEE 802.11 (802.11a/b/g) => DSSS (11Mbps), OFDM (54Mbps)
  - The characteristics of wireless medium: fading, attenuation, interference from other radiation sources.

Configure MAC

- **WifiMac** abstract class
  - List of MAC Types
    - **AdhocWifiMac** – Infrastructure less network
    - **ApWifiMac** – Access point Node MAC
    - **StaWifiMac** – Station Node MAC
    - Etc.
  - We use Helper Class
    - **NqosWifiMacHelper** or **WifiMacHelper**
  - Set the appropiate MAC from the list and Set Attributes
    - **void SetType(T,A,V....);**
    - **T**- Type of MAC
    - **A**- Name of Attribute
    - **V** –Value of Attribute

Configure MAC




```
   WifiMacHelper mac;
   Ssid ssid = Ssid ("ns-3-ssid");
   mac.SetType ("ns3::StaWifiMac", "Ssid", SsidValue
(ssid), "ActiveProbing", BooleanValue (false));

   NetDeviceContainer staDevices;
   staDevices = wifi.Install (phy, mac, wifiStaNodes);
```

```
   mac.SetType ("ns3::ApWifiMac", "Ssid", SsidValue
(ssid));

   NetDeviceContainer apDevices;
   apDevices = wifi.Install (phy, mac, wifiApNode);
```

4. Program: Set Mobility

```
MobilityHelper mobility;
```

Set two-dimensional grid to place nodes

```
mobility.SetPositionAllocator ("ns3::GridPositionAllocator", "MinX",
DoubleValue (0.0), "MinY", DoubleValue (0.0), "DeltaX", DoubleValue
(5.0), "DeltaY", DoubleValue (10.0), "GridWidth", UintegerValue (3),
"LayoutType", StringValue ("RowFirst"));
```

```
mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel", "Bounds",
RectangleValue (Rectangle (-50, 50, -50, 50)));
mobility.Install (wifiStaNodes);
```

Set STA nodes

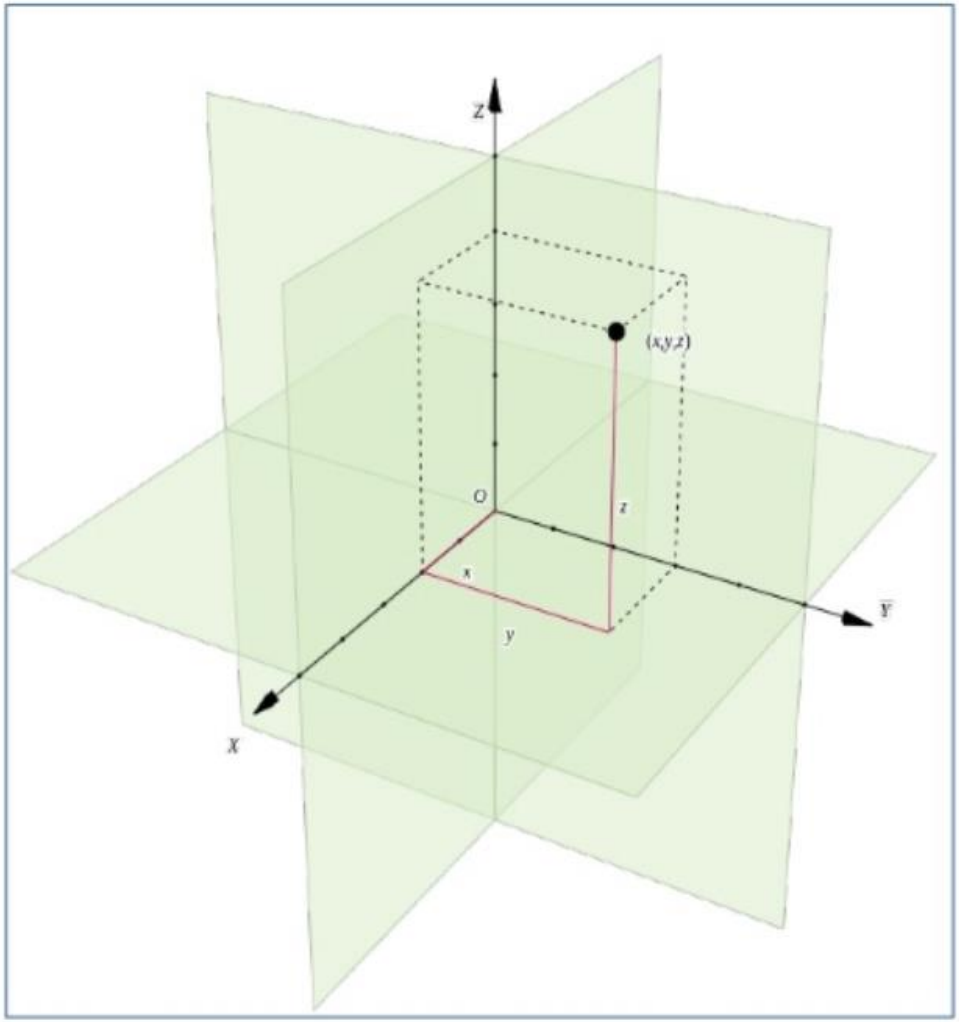
```
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (wifiApNode);
```

Set AP node for fixed position

Mobility

- It used to track and maintain the
  - Current Cartesian position
  - Speed of an object
  - Placement of Node
  - Setup Mobility Model

Mobility [Cartesian Position]



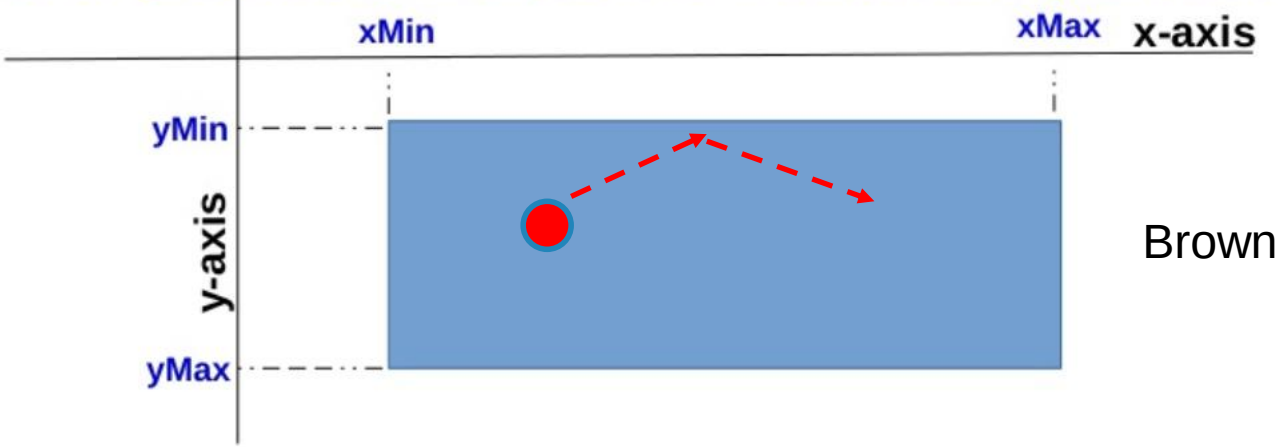
Source:http://en.wikipedia.org/wiki/Cartesian_coordinate_system

Mobility

- Assign Mobility to WIFI Nodes
  - List of **Mobility Model**
    - **ConstantAccelerationMobilityModel**
    - **ConstantPositionMobilityModel**
    - **ConstantVelocityMobilityModel**
    - **RandomDirection2dMobilityModel**
    - **RandomWalk2dMobilityModel**
    - **Etc**
  - List of **Allocator Model (placement of Node)**
    - **RandomDiscPositionAllocator**
    - **RandomRectanglePositionAllocator**
    - **GridPositionAllocator**
    - **Etc.**

Mobility Model

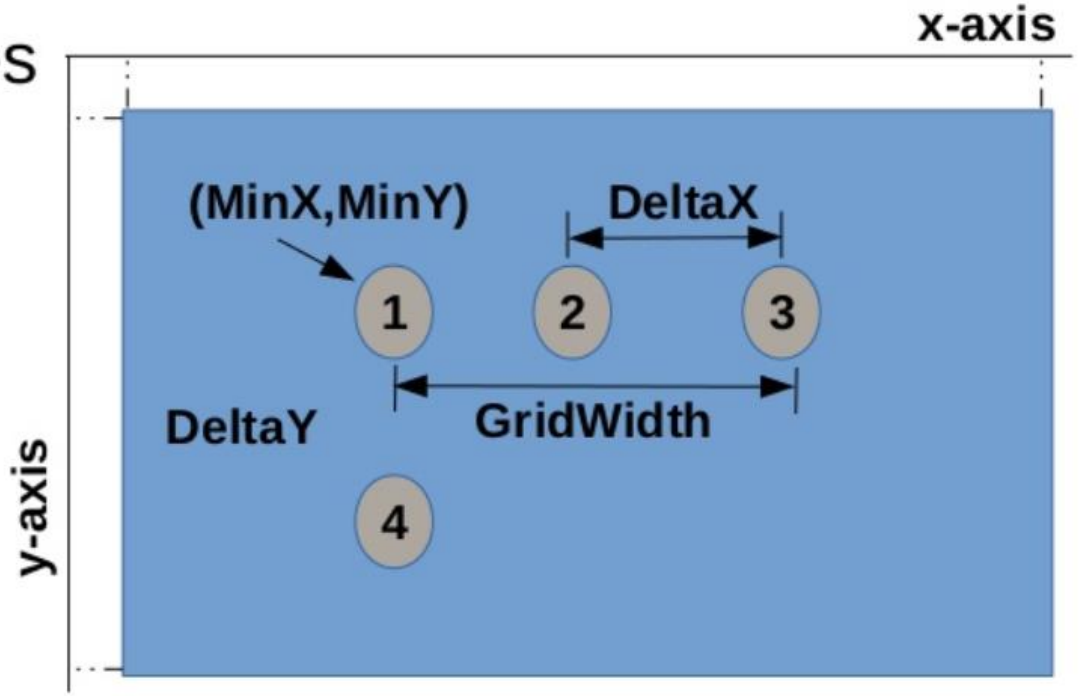
- RandomWalk2dMobilityModel
  - 2D random walk mobility model
  - Each instance moves with a speed and direction choosen at random
  - Nodes moves in Boundaries specified by Rectangle
  - `Rectangle(double xMin,double xMax,double yMin,double yMax)`



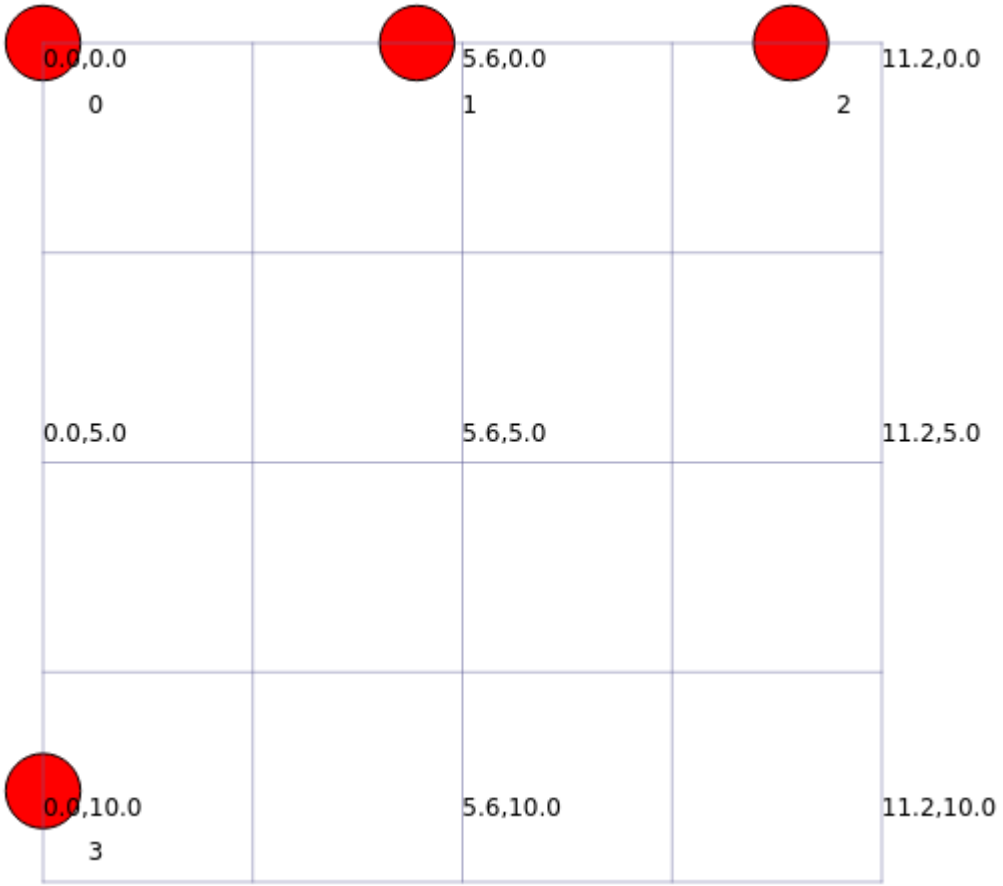
Brownian Motion

Allocator Model

- **GridPositionAllocator**
  - Allocate position on a rectangular 2D grid
  - List of Attributes
    - **MinX**
    - **MinY**
    - **DeltaX**
    - **DeltaY**
    - **GridWidth**
    - **LayoutType**
      - **ROW_FIRST**
      - **COLUMN_FIRST**



Nodes Position




```
mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
  "MinX", DoubleValue (0.0),
  "MinY", DoubleValue (0.0),
  "DeltaX", DoubleValue (5.0),
  "DeltaY", DoubleValue (10.0),
  "GridWidth", UintegerValue (3),
  "LayoutType", StringValue ("RowFirst"));
```

Mobility

- It uses Helper Class
  - MobilityHelper
    - ✓ SetMobilityModel () – Set Mobility Model
    - ✓ SetPositionAllocator () – Set Position Allocator
  - Install the mobility on Nodes
  - Mobility Model [Access Point Node]
    - ✓ ConstantPositionMobilityModel

5. Internet Stack & Ipv4Address

- Install Protocol Stack
  - ✓ InternetStackHelper
    - Install ()
- Assign IP address to the NetDevice
  - ✓ Ipv4InterfaceContainer
    - Assign ()

```
InternetStackHelper stack;
stack.Install (wifiApNode);
stack.Install (wifiStaNodes);

Ipv4AddressHelper address;

address.SetBase ("10.1.1.0", "255.255.255.0");
Ipv4InterfaceContainer wifiStaInterfaces;
wifiStaInterfaces = address.Assign (staDevices);
address.Assign (apDevices);
```

6. Application UDP

UDP Server (node(0))

```
   UdpEchoServerHelper echoServer (9);

   ApplicationContainer serverApps = echoServer.Install
(wifiStaNodes.Get(0));
   serverApps.Start (Seconds (1.0));
   serverApps.Stop (Seconds (10.0));
```

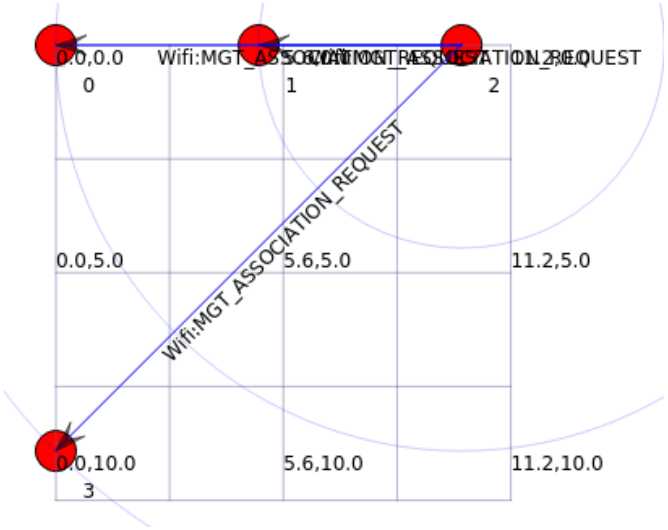
```
   UdpEchoClientHelper echoClient (wifiStaInterfaces.GetAddress (2), 9);
   echoClient.SetAttribute ("MaxPackets", UintegerValue (10));
   echoClient.SetAttribute ("Interval", TimeValue (Seconds (0.5)));
   echoClient.SetAttribute ("PacketSize", UintegerValue (1024));

   ApplicationContainer clientApps = echoClient.Install (wifiStaNodes.Get
(2));
   clientApps.Start (Seconds (2.0));
   clientApps.Stop (Seconds (10.0));
```

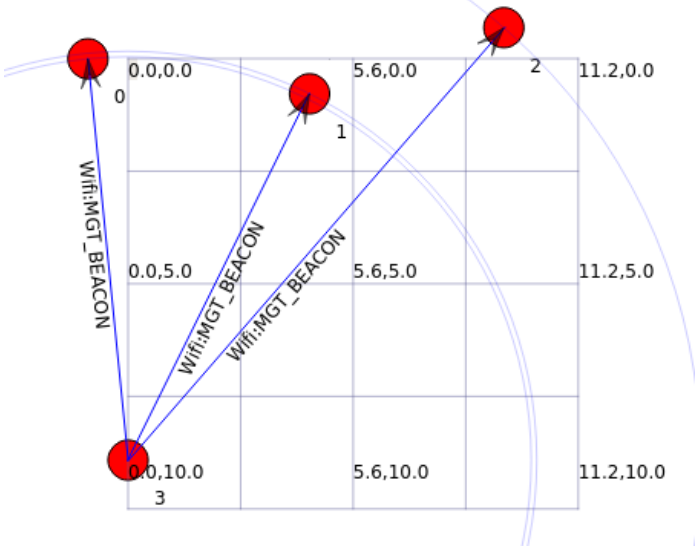
UDP Client (node(2))

7. Simulation Run

Client send data



AP send beacon




```
At time 2s client sent 1024 bytes to 10.1.1.3 port 9
At time 2.5s client sent 1024 bytes to 10.1.1.3 port 9
At time 3s client sent 1024 bytes to 10.1.1.3 port 9
At time 3.5s client sent 1024 bytes to 10.1.1.3 port 9
At time 4s client sent 1024 bytes to 10.1.1.3 port 9
At time 4.5s client sent 1024 bytes to 10.1.1.3 port 9
At time 5s client sent 1024 bytes to 10.1.1.3 port 9
At time 5.5s client sent 1024 bytes to 10.1.1.3 port 9
At time 6s client sent 1024 bytes to 10.1.1.3 port 9
At time 6.5s client sent 1024 bytes to 10.1.1.3 port 9
```