Point-To-Point Topology
Default Application
Scenario 1

• Create a simple client server node and send a UDP echo packet

<client>
10.0.0.11/24

<Point-To-Point Link>
rate=5Mbps, delay=2ms

<server>
IP 10.0.0.12/24
Port 9
Flow Chart

1. Create Node
2. Create NetDevice & Channel
3. Install NetDevice & Channel to Node
4. Create Protocol Stack
5. Install Protocol Stack on Nodes
6. Assign IP Address to NetDevice
7. Install Default UDP Echo Application on Nodes
Header Files

• Include header files
  – point-to-point.h
  – internet-module.h
  – applications-module.h
  – core-module.h
Node

- Use to create **NODE**
  - Use Node class
  - Create object using template function
    
    ```cpp
    Ptr<T> CreateObject(void)
    ```

- Maintain Reference Count
- Wrapper handle new operator
Helper Node Class

• Use of **Helper Class**
  
  – To create Group of **NODE** we use **NodeContainer**

  ```
  NodeContainer
  Create(num_of_nodes)
  ```

  NodeContainer nc;
  nc.Create(3);
CheckPoint

<client>

<server>
Channel & NetDevice

• It is a logical path over which information flows
  
  – To create **Channel** we use following classes:-
    
    • WifiChannel
    • CsmaChannel
    • PointToPointChannel
    • Etc...

  – To create **NetDevice** we use following classes:-
    
    • WifiNetDevice
    • CsmaNetDevice
    • PointToPointNetDevice
    • Etc...
Channel & NetDevice

• Helper Classes Couple NetDevice with Channel
  – We can use Helper Classes:
    • WifiHelper
    • CsmaHelper
    • PointToPointHelper

• Now **Install** NetDevice & Channel
  – NetDeviceContainer device
  – device=pointTopoint.Install(node_container);
CheckPoint

NetDevice <client> point-to-point channel NetDevice <server>
Internet Stack & Ipv4Address

- Now its time to install **Protocol Stack**
  - Helper Class
    - InternetStackHelper ish
- Now **INSTALL** protocol stack on Nodes
  - *ish.install(node_container)*;
- Ipv4Address
  - Now associate the **devices** on our node with **IP addresses**
    - Ipv4AddressHelper iaddr;
    - iaddr.SetBase("10.1.1.0","255.255.255.0");
  - Now **ASSIGN this IP address to NetDevice** using Ipv4Interface object
    - Ipv4InterfaceContainer iinter = iaddr.Assign(device)
CheckPoint

NetDevice

<client>
10.0.0.11/24

Internet Stack

point-to-point channel

NetDevice

<server>
10.0.0.12/24

Internet Stack
Application

- **Application** abstract class
  - `UdpEchoServerApplication` – server application
  - `UdpEchoClientApplication` - client application

- We use Helper classes
  - `UdpEchoServerHelper`
  - `UdpEchoClientHelper`
Server Application

UdpEchoServerHelper server(9);

- Set up a **UDP echo server** application on one of the node
- Require the *port number as a parameter* to the constructor
- Install server application on server node

```cpp
ApplicationContainer serApp;
serApp = server.Install(nodes.Get(1));
servApp.Start(Seconds(1.0));
servApp.Stop(Seconds(10.0));
```
CheckPoint

Internet Stack
NetDevice

<client>
10.0.0.11/24

point-to-point channel

Server
App
Internet Stack
NetDevice

<server>
10.0.0.12/24
Client Application

- Pass parameter (to helper) to *set the Remote Address* and *Remote port for for client to connect server*.

  ```cpp
  UdpEchoClientHelper client(i.GetAddress(1),9);
  ```

- Install client application on client node

  ```cpp
  ApplicationContainer clientApp;
  clientApp=client.Install(nodes.Get(0));
  clientApp.Start(Seconds(2.0));
  clientApp.Stop(Seconds(9.0));
  ```
CheckPoint

Client App
Internet Stack
NetDevice

<client>
10.0.0.11/24

point-to-point channel

Server App
Internet Stack
NetDevice

<server>
10.0.0.12/24
Start Simulation

Simulation::Run();
Simulation::Destroy();
return 0;
Running Example

- Copy program from example to scratch folder
- And run using following commands
  - `./waf --run /scratch/scenario1`
No Output :(
Log Component

• Logs are generally used to get useful information.
• NS-3 provides different levels of logs which are as follows:-
  - LOG_INFO
  - LOG_FUNCTION
  - LOG_LOGIC
  - LOG_ALL
  - etc.
Enable Log

- To enable log:
  - `LogComponentEnable(module_name, log_level)`

- Example
  - `module_name`
    - `ns3::UdpEchoClientApplication`
    - `ns3::UdpEchoServerApplication`
    - and, more
  - `log_level`
    - `LOG_LEVEL_INFO`
    - `LOG_LEVEL_FUNCTION`
    - `LOG_LEVEL_ALL`
    - and, more
Let's Play
with
Command Line Arguments
Command Line Argument

- We use command line parser

```c
int main(int argc, char * argv[])
{
    
    CommandLine cmd
    cmd.Parse(argc,argv)
    
}
```

- It opens the door to the ns-3 global variable and `Attribute` subsystem
Attributes

- Goal of Attribute system is to *organize the access of internal members* objects of a simulation.
- Most often user is *interested in studying or tracing* particular internal variables.
- Example:
  - What is network performance if we change the packet size?
  - How transmission time varies with data rate?
Attribute

- How to add attribute?
  - `TypeId AddAttribute(...)
  - Sample Code `<PointToPointNetDevice>`

```cpp
TypeId PointToPointNetDevice::GetTypeId (void)
{
    static TypeId tid = TypeId ("ns3::PointToPointNetDevice")
        .SetParent<NetDevice>()
        .AddGroupName ("PointToPoint")
        .AddConstructor<PointToPointNetDevice> ()
        .AddAttribute<"Mtu", "The MAC-level Maximum Transmission Unit", 
                        UintegerValue (DEFAULT MTU),
                        MakeUintegerAccessor (&PointToPointNetDevice::SetMtu, 
                                             &PointToPointNetDevice::GetMtu),
                        MakeUintegerChecker<uint16_t> () )

        .AddAttribute("Address",
                    "The MAC address of this device.",
                    Mac48AddressValue (Mac48Address ("ff:ff:ff:ff:ff:ff")),
                    MakeMac48AddressAccessor (&PointToPointNetDevice::m_address),
                    MakeMac48AddressChecker () )

        .AddAttribute("DataRate",
                    "The default data rate for point to point links",
                    DataRateValue (DataRate ("32768b/s")),
                    MakeDataRateAccessor (&PointToPointNetDevice::m_bps),
                    MakeDataRateChecker () )

        .AddAttribute("ReceiveErrorModel",
                    "The receiver error model used to simulate packet loss",
                    PointerValue (),
                    MakePointerAccessor (&PointToPointNetDevice::m_receiveErrorModel),
                    MakePointerChecker<ErrorModel> () )

        .AddAttribute("InterframeGap",
                    "The time to wait between packet (frame) transmissions",
                    TimeValue (Seconds (0.0)),
                    MakeTimeAccessor (&PointToPointNetDevice::m_tInterframeGap),
                    MakeTimeChecker ()
                       )
```
Let's Play with CMD

- Run the script in the following way
  
  ```bash
  ./waf --run "scratch/senario1 --PrintHelp"
  ```

- Now use **--PrintAttributes** option to do know the attributes

  ```bash
  ./waf --run "scratch/scenario1 --PrintAttributes=ns3::PointToPointNetDevice"
  ```

  ```bash
  ./waf --run "scratch/senario1 --ns3::PointToPointNetDevice::DataRate=5Mbps"
  ```
Lets Hook User Defined Arguments

- **AddValue** method of command line parser

```c
int main(int argc,char * argv[ ])
{
    uint32_t nPacket=1;
    ...
    CommandLine cmd;
    cmd.AddValue("nPackets","Number of packets to echo",nPacket);
    cmd.Parse(argc,argv);
    ...
    ...
    echoClient.setAttribute("MaxPackets",UintegerValue(nPackets));
    ...
    ...
}
./waf -run "scratch/myfirst --PrintAttributes"
./waf -run "scratch/myfirst --nPacket"
```
Simple Modification in Topology CODE

- Change following values:-
  - **NetDevice**
    - DataRate
    - Delay
  - **UDP packet**
    - MaxPacket
    - Interval
    - Packet Size
• What is the use of **Helper Classes**?

• What is the **effect of packet size** on the **Round Trip Time (RTT)**?
  - **Packet Size=256,512,786,1024,1586,1782**

• What is effect of **Maximum Transmission Unit (MTU)** on the **Round Trip Time (RTT)**?
  - **MTU= 256 , 512 , 1024 , 1500**
To be Continued...