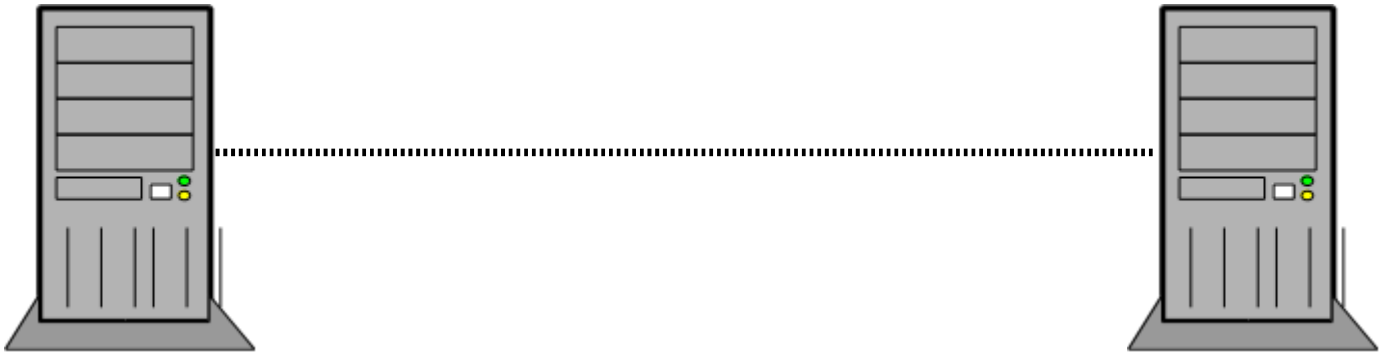
Point-To-Point Topology
Default Application

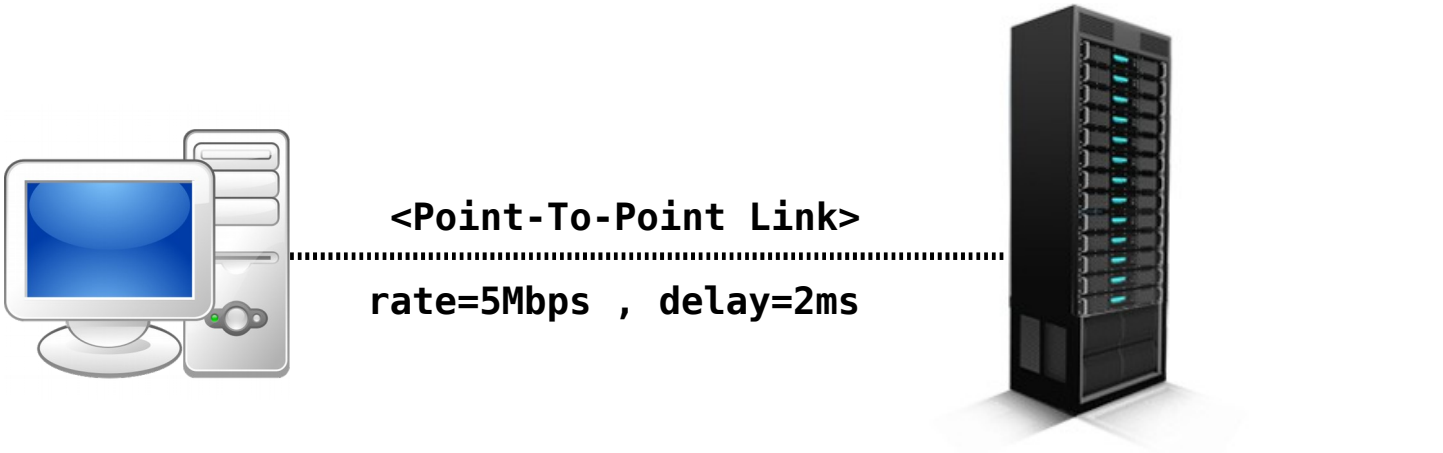


STVN-2016
Jointly organized by
Poornima University, Jaipur & IIIT-Kota

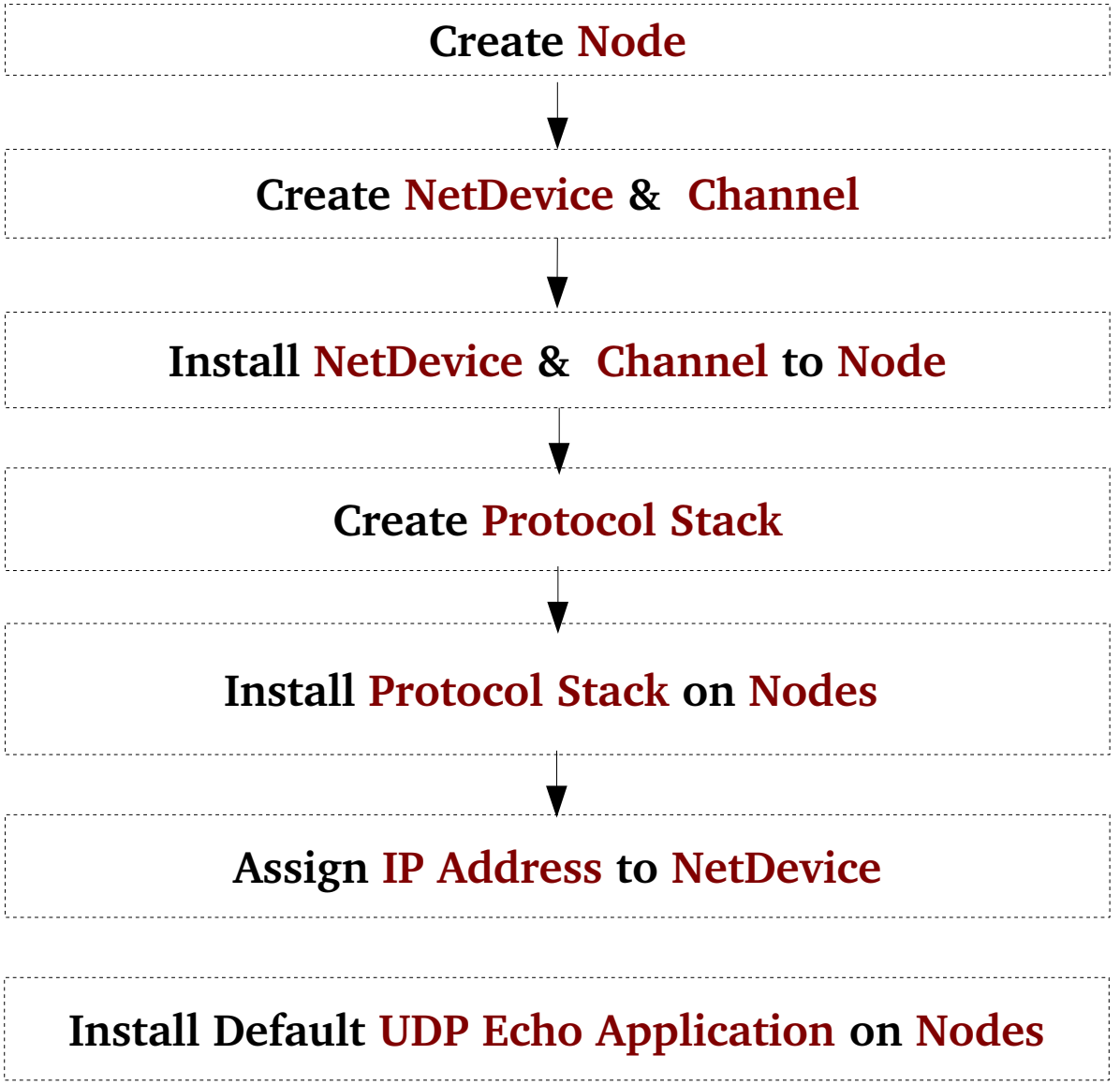
Rahul Hada
hada.rahul@gmail.com

Scenario 1

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



Flow Chart


Create Node
Create NetDevice & Channel
Install NetDevice & Channel to Node
Create Protocol Stack
Install Protocol Stack on Nodes
Assign IP Address to NetDevice
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
    Ptr<T>CreateObject(void)

Maintain Reference Count

Wrapper handle new operator

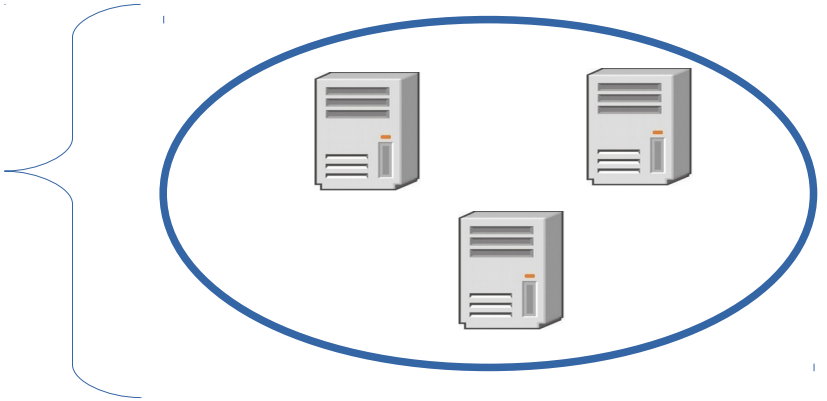
Helper Node Class

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

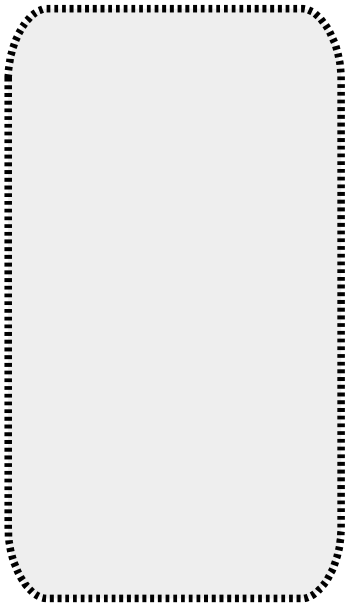
NodeContainer

Create(num_of_nodes)

```
NodeContainer nc;
nc.Create(3);
```



CheckPoint

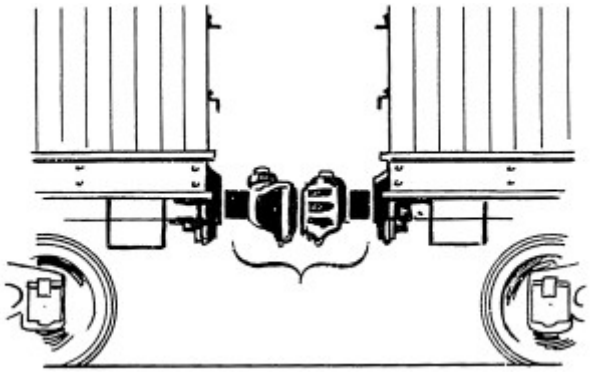


Channel & NetDevice

- 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…**

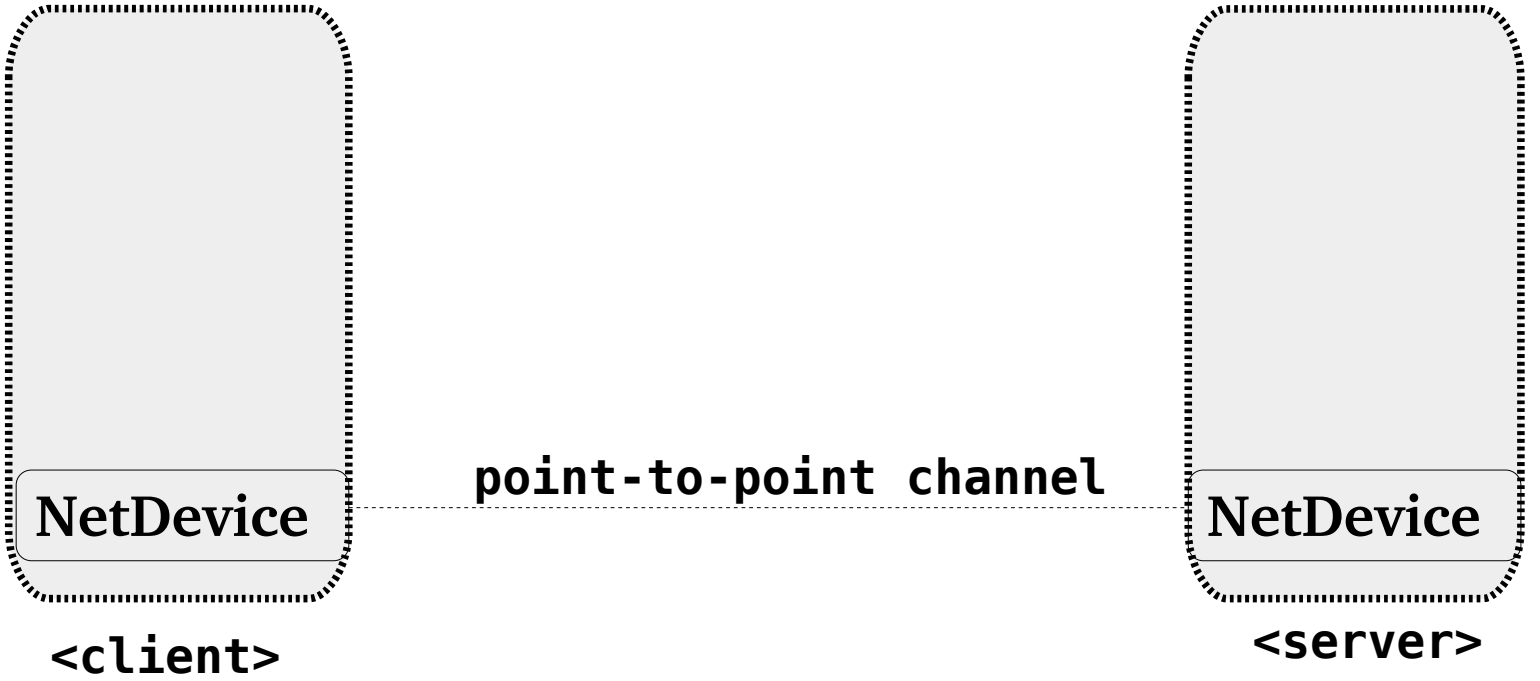
Channel & NetDevice

- 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);**

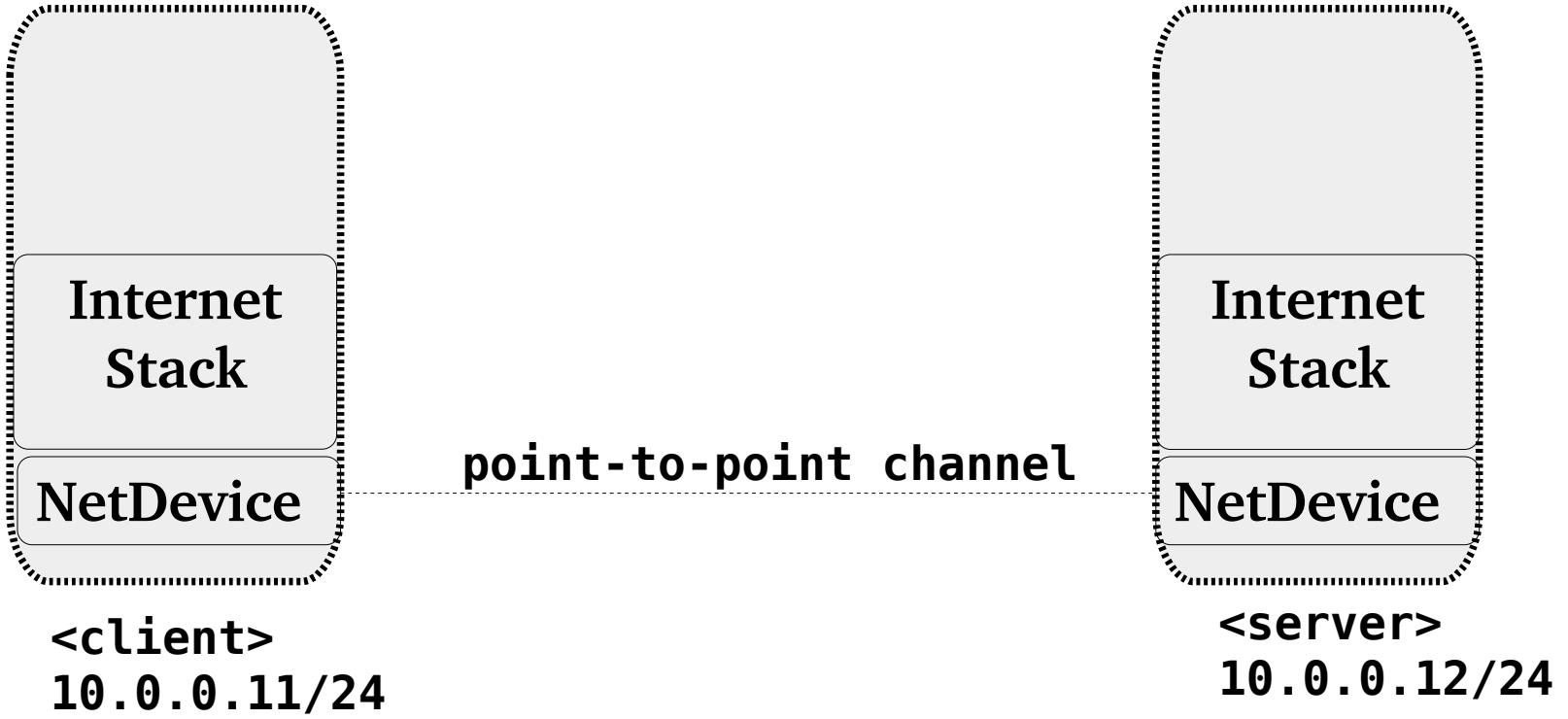
CheckPoint



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


Internet Stack
NetDevice
point-to-point channel
Internet Stack
NetDevice
<client>
10.0.0.11/24
<server>
10.0.0.12/24

Application

- **Application** abstract class
  - **UdpEchoServerApplication** – server application
  - **UdpEchoClientApplication** -client application
- We use Helper classes
  - **UdpEchoServerHelper**
  - **UdpEchoClientHelper**

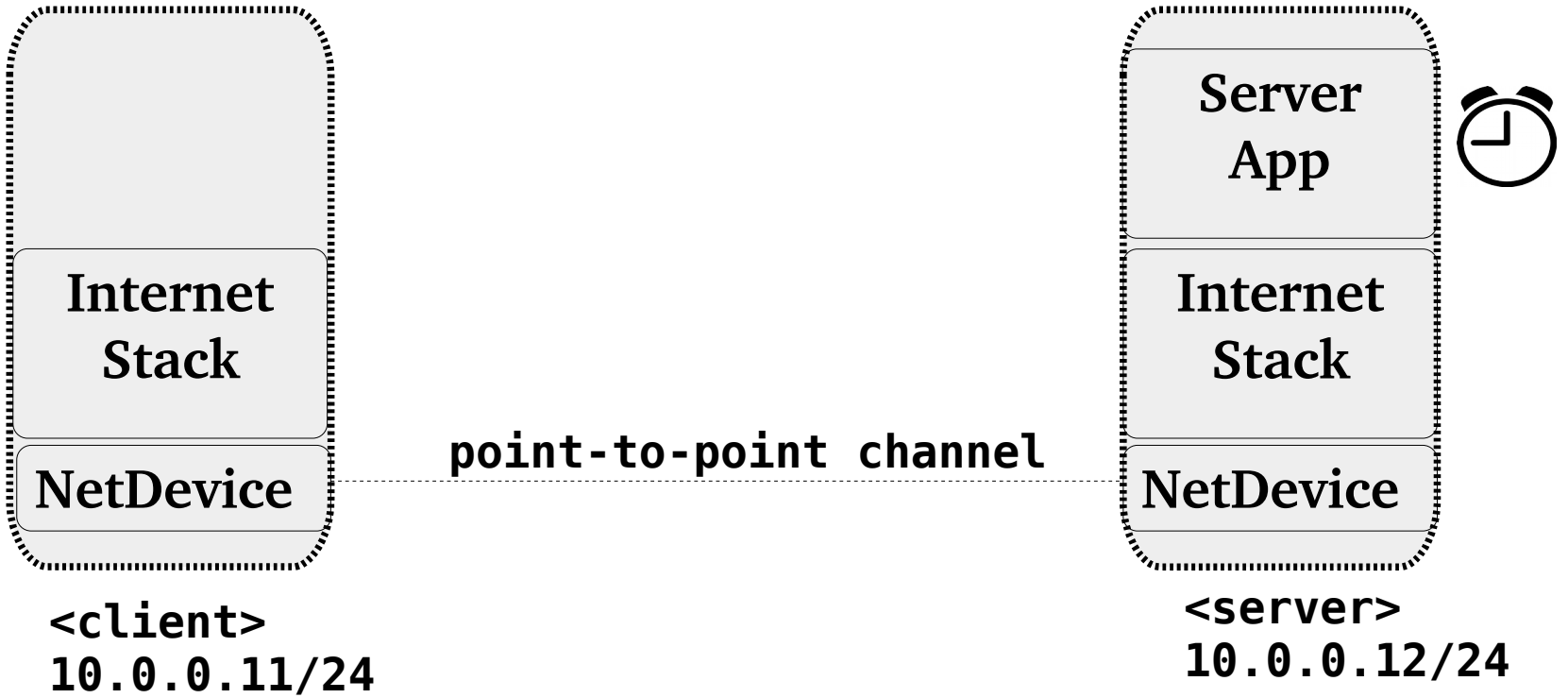
Server Application

```
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

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

CheckPoint


Server App
Internet Stack
NetDevice
point-to-point channel
Internet Stack
NetDevice
<client>
10.0.0.11/24
<server>
10.0.0.12/24

Client Application

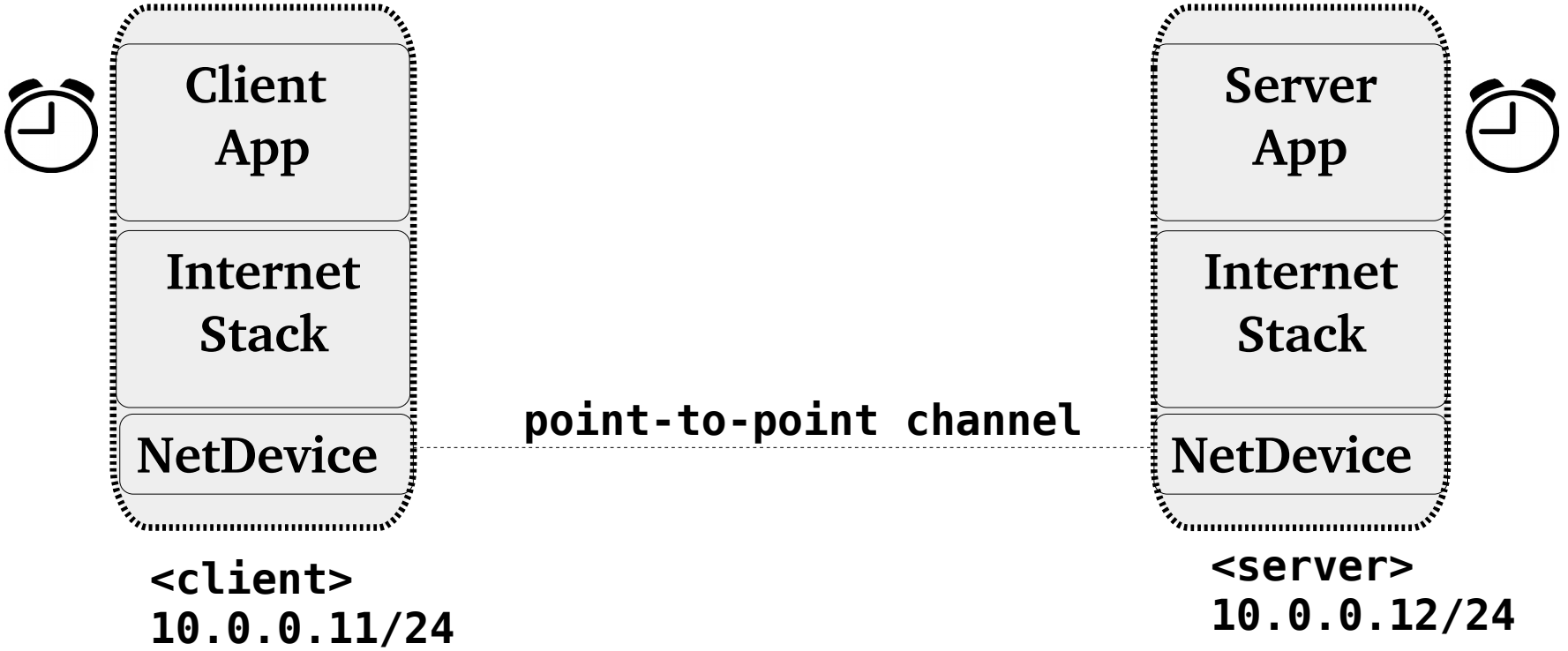
- Pass parameter (to helper) to ***set the Remote Address*** and ***Remote port for for client to connect server.***

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

- **Install client application on client node**

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

CheckPoint


Client App
Internet Stack
NetDevice
point-to-point channel
Server App
Internet Stack
NetDevice
<client>
10.0.0.11/24
<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 Compoment

- 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`**

Lets Play
with
Command Line Arguments

Command Line Argument

- We use command line parser

```
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`**>

```
TypeId
PointToPointNetDevice::GetTypeId (void)
{
  static TypeId tid = TypeId ("ns3::PointToPointNetDevice")
    .SetParent<NetDevice> ()
    .SetGroupName ("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 ())
```

Brief Description

Name of attribute

Type of value

Accessing variable

Let's Play with CMD

- Run the script in the following way
  - `./waf --run “scratch/senario1 --PrintHelp”`
- Now use **`--PrintAttributes`** option to do know the attributes

```
./waf --run “scratch/scenario1
--PrintAttributes=ns3::PointToPointNetDevice”
```

```
./waf –run “scratch/senario1 –
ns3::PointToPointNetDevice::DataRate=5Mbps”
```

Lets Hook User Defined Arguments

- **AddValue** method of command line parser

```
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));
   . . .
   . . .

}
```

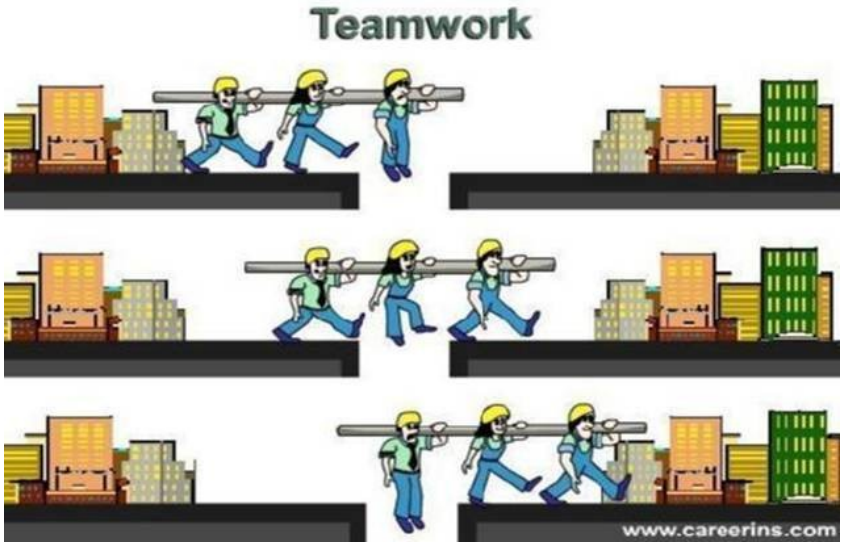
```
./waf –run "scratch/myfirst –PrintAttributes"
./waf –run "scratch/myfirst --nPacket"
```

Simple Modification in Topology CODE

- Change following values:-
- **NetDevice**
  - DataRate
  - Delay
- **UDP packet**
  - MaxPacket
  - Interval
  - Packet Size

RECALL ALERT

NETWORK SIMULATOR



- 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...