University at Buffalo
*The State University of New York*

# EE450/550: Principles of Networking
# Laboratory Project - Assignment 3

**Given:** April 8, 2017
**Due:** May 7, 2017 (11:59 PM)

**Submission Instructions:**

Submit your assignment in electronic format (such as a *doc* or *pdf*) **directly on UB Learns**

**Elena Bernal Mor, Ph.D.**
Teaching Assistant Professor
Department of Electrical Engineering
University at Buffalo, The State University of New York
*Office:* 209 Davis Hall
*E-mail:* elbermo@buffalo.edu

**Objective:**
In this final lab assignment, you will learn how to implement your own protocol in NS-3. In particular, you will implement your medium access control protocol, physical layer and channel. Before diving into the implementation you are recommended to read the following two sections of the NS-3 manual:
https://www.nsnam.org/docs/manual/html/new-models.html
https://www.nsnam.org/docs/manual/html/new-modules.html

A very high level block diagram of the NS-3 protocol stack implementation has been portrayed in the following Figure 1.

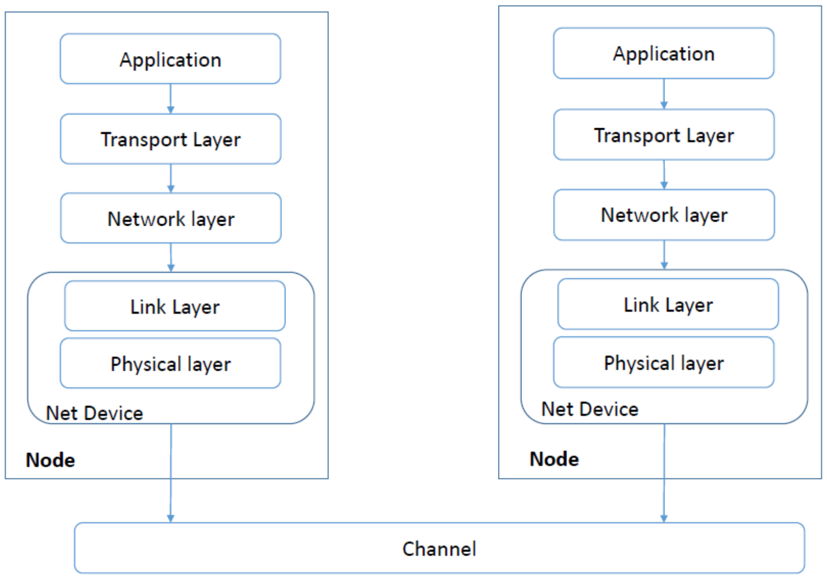

**Figure 1:** High level block diagram of the NS-3 protocol stack implementation

As you can see from the figure, first two protocol layers are implemented in the NetDevice and Channel class. So to implement your MAC protocol, you need to subclass NetDevice and add your protocol specific functionalities. You can implement both MAC and PHY layer in the PnetNetDevice class or you can separate them by creating new class for each.

We have provided you with a skeleton module named "pnet" with the basic implementation of net-device and channel, namely PnetNetDevice and PnetChannel classes, which were subclassed from the NetDevice and Channel classes, respectively. We have also implemented MAC layer and PHY layer, namely MyPnetMac and MyPnetPhy respectively, which are held together by PnetNetDevice.

The helper class basically connects all these components together to provide the total functionality shown in the figure. For your convenience we have already provided the implementation of the helper class for "pnet" module.

**Current implementation:**
PnetNetDevice: Upon receiving the packet from upper layer, the device passes the packet to the MAC layer by calling Enqueue() method of MAC layer. The interaction between the classes in handling the packet flow is shown in figure 2.

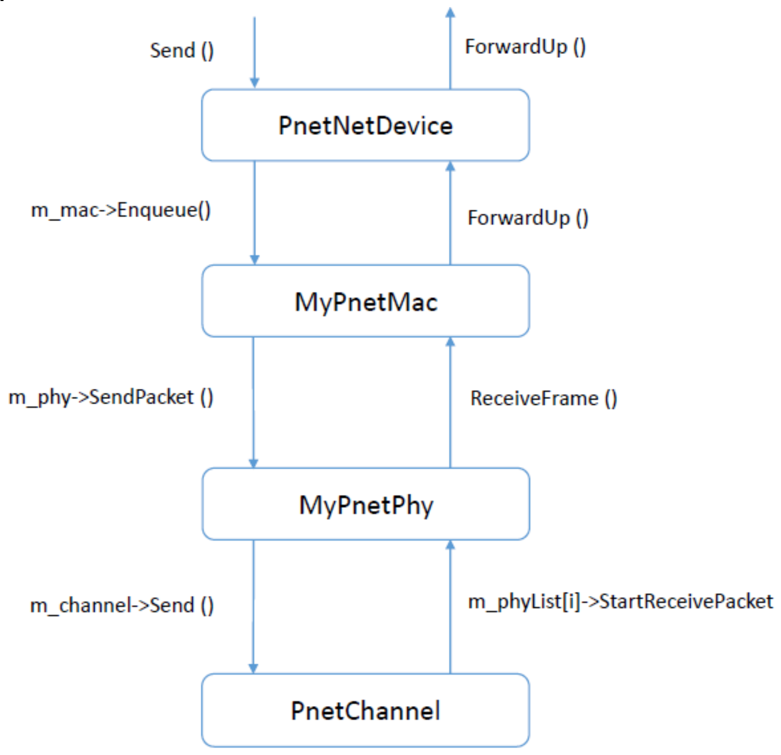

Figure 2: Class interaction diagram that shows the packet flow

- **MAC Layer:** MAC layer puts the packet in the Queue and calls the SendFrame() function. MAC layer then passes the packet to the PHY layer by calling the SendPacket() method on the pointer to PHY layer.

```
#ifndef MY_PNET_MAC_H
#define MY_PNET_MAC_H


namespace ns3 {

class PnetPhy;


class MyPnetMac : public PnetMac
{
public:

  static TypeId GetTypeId (void);
  MyPnetMac ();
  virtual~MyPnetMac ();

  void Enqueue (Ptr<Packet> packet, Mac48Address to);
  void SetForwardUpCallback (PnetMac::UpCallback upCallback);
  void SendFrame (void);
  void ReceiveFrame (Ptr<Packet> packet);

  void SetAddress (Mac48Address ad);
  Mac48Address GetAddress (void) const;
  void SetPhy (Ptr<PnetPhy> phy);
  Ptr<PnetPhy> GetPhy (void) const;

private:

  Ptr<PnetPhy> m_phy;
  Mac48Address m_address;//source address

  uint16_t m_dataRetryLimit;
  std::queue<Ptr<Packet> > m_queue;

  PnetMac::UpCallback m_forwardUpCb;
protected:
};

} // namespace ns3

#endif /* MY_PNET_MAC_H */
```

- **PHY Layer:** PHY layer passes the packet to the channel by calling the Send() method on the pointer to Channel. In addition, it schedules a transmission complete event after the transmission time.

```
#ifndef MY_PNET_PHY_H
#define MY_PNET_PHY_H

namespace ns3 {

class PnetNetDevice;

class MyPnetPhy : public PnetPhy
{
public:

  static TypeId GetTypeId (void);

  MyPnetPhy ();
  virtual ~MyPnetPhy ();

  void SetReceiveCallback (PnetPhy::RxCallback callback);
  void SendPacket (Ptr<Packet> packet);
  void TransmitComplete ();
  void StartReceivePacket (Ptr<Packet> packet, Time rxDuration);
  void ReceiveComplete (Ptr<Packet> packet);

  void SetChannel (Ptr<PnetChannel> channel);
  void SetDevice (Ptr<PnetNetDevice> device);
  Ptr<PnetNetDevice> GetDevice (void) const;
  Ptr<PnetChannel> GetChannel (void) const;

  Time CalculateTxDuration (Ptr<const Packet> packet);

private:
  Ptr<PnetNetDevice> m_device;
  Ptr<PnetChannel> m_channel;
  Ptr<MobilityModel>   m_mobility;       //!< Pointer to the mobility model
  DataRate m_bps; //!< The physical layer nominal Data rate. Zero means infinite

  PnetPhy::RxCallback m_rxCallback;

};

} //namespace ns3
```

```
#endif /* MY_PNET_PHY_H */
```

- **Channel:** Upon receiving the packet from the PnetNetDevice, PnetChannel schedules the packet transfer to all the PnetNetDevices connected to the PnetChannel except the sender PnetNetDevice. The transfers are scheduled to happen after the propagation delay.

```
class PnetChannel : public Channel
{
public:
  static TypeId GetTypeId (void);
  PnetChannel ();

  void Send (Ptr<PnetNetDevice> sender ,Ptr<Packet> p, Time duration);

  void Add (Ptr<PnetPhy> phy);

  // inherited from ns3::Channel
  virtual uint32_t GetNDevices (void) const;
  virtual Ptr<NetDevice> GetDevice (uint32_t i) const;

private:
  std::vector<Ptr<PnetPhy> > PhyList;
  PhyList m_phyList;
  Time m_delay;
};
```

**Assignment:**
**Task1:**

- Modify the PnetChannel class to get the propagation delay between the sender and every other node unlike the current implementation where one propagation delay is applied to all. You can calculate the propagation delay by finding out the distance using the mobility model and divide that by the speed of light. Modify only the *Send()* method to achieve the task.

**Task2:**

- Modify the MyPnetMac class to implement positive acknowledgement for each packet. You may want to create a new function like SendAck() in the class. For implementing timeout for the acknowledgement, NS-3 provides very convenient function Simulator::Schedule(). You may want to read the following link to explore the function details.
**https://www.nsnam.org/docs/manual/html/events.html**

- Add an attribute to the MyPnetMac class that will allow you to set the maximum number of retransmission. Refer to following link for detailed description about how to add attributes.
**https://www.nsnam.org/docs/manual/html/attributes.html**

**Task3:**

- Add states to the PHY layer class MyPnetPhy that will indicate if the PHY layer is transmitting, receiving, idle state. Implement how to detect collision using the states.