ns-3 经验分享

杨占武

大连理工大学软件学院

2024 年 5 月 20 日

简介

- 离散事件网络模拟器
- 模块化设计，便于扩展
- 免费的开源软件，文档详尽
- 代码晦涩难懂，学习曲线陡峭
- 开发环境推荐：Linux+Vscode+Copilot

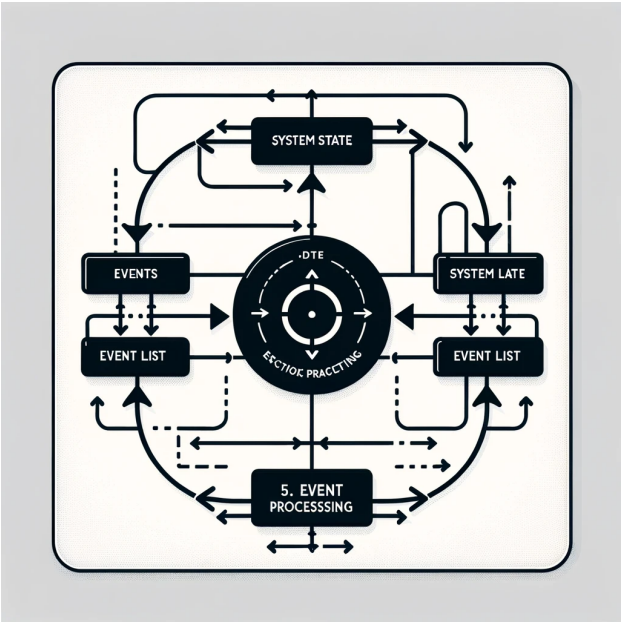

图 1: 事件示意图

- Node，基本计算设备的抽象
- Application，生成要模拟活动的用户程序的抽象
- Channel，数据在网络中传输的介质的抽象
- NetDevice，软件驱动程序和模拟硬件的抽象
- Topology Helpers，将 Nodes 设置 Channel 进行连接

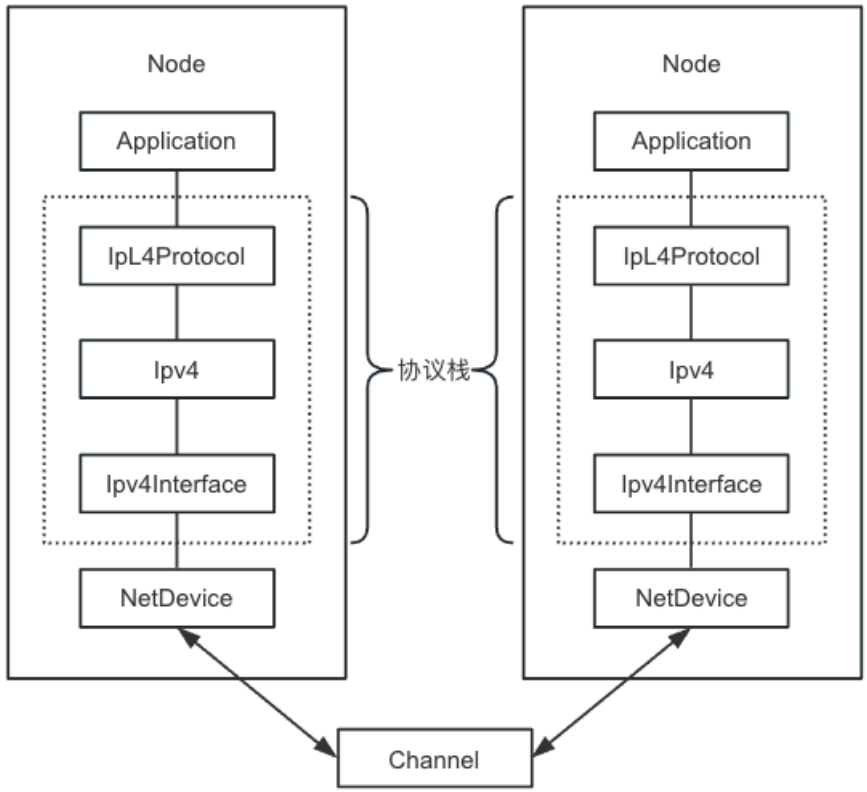

图 2: ns-3 模型

- 选择或开发相应的模块
- 编写网络仿真脚本
  1. 生成节点，ns-3 中节点相当于一个空的计算机外壳
  2. 安装网卡设备，不同的网络类型有不同的网络设备，从而提供不同的通信、物理层和 MAC 层
  3. 安装协议栈，ns-3 网络中一般是 TCP/IP 协议栈
  4. 安装应用层协议，依据选择的传输层协议选择相应的应用层协议
  5. 其他配置：如节点是否移动，是否需要能量管理等
  6. 启动仿真：整个网络场景配置完毕，启动仿真
  7. debug：查看仿真结果，分析问题，这步相当痛苦

性能评判标准

- 吞吐量：在单位时间内通过某个网络的实际的数据量
- 时延：数据从信源发送到信宿所需的时间
- 交付率：实际接收的数据包 / 总共发送的数据包
- 丢包率：1 - 交付率
- 能耗：一段时间内消耗的能量
- 路径长度：路由的跳数

关键实现

回调函数

```
1  template <typename R, typename... Args>
2  Callback<R, Args...>
3  MakeCallback(R (*fnPtr)(Args...))
4  {
5      return Callback<R, Args...>(fnPtr);
6  }
7
8  source->TraceConnectWithoutContext
9  ("Tx", MakeCallback(&TxPacket));
10 sink->TraceConnectWithoutContext("Rx",
       MakeCallback
11 (&MacCompareExample::ReceivePacket, this));
```

关键实现

Schedule 函数

```
1 template <typename... Us, typename... Ts>
2 EventId
3 Simulator::Schedule(const Time& delay, void (*f)
      (Us...), Ts&&... args)
4 {
5     return DoSchedule(delay, MakeEvent(f, std::
          forward<Ts>(args)...));
6 }
7
8 Simulator::Schedule(Seconds(TotalTime / 100), &
      PrintProgress, TotalTime);
```

OnOffApplication

```
1  void
2  OnOffApplication::ScheduleNextTx()
3  {
4      if (m_maxBytes == 0 || m_totBytes <
           m_maxBytes)
5      {
6          uint32_t bits = m_pktSize * 8 -
               m_residualBits;
7          Time nextTime(Seconds(bits / static_cast
               <double>(m_cbrRate.GetBitRate())));
8          m_sendEvent = Simulator::Schedule(
               nextTime, &OnOffApplication::
               SendPacket, this);
9      }
10 }
```

PacketSink

```
1 void HandleRead(Ptr<Socket> socket);
2
3 void HandleAccept(Ptr<Socket> socket, const
      Address& from);
4
5 void HandlePeerClose(Ptr<Socket> socket);
6
7 void HandlePeerError(Ptr<Socket> socket);
8
9 void PacketReceived(const Ptr<Packet>& p, const
      Address& from, const Address& localAddress);
```

模块实现

- PHY 层模型：模拟特定修正案和共有的 PHY 层操作和功能
- MAC 低层模型：
  1. FrameExchangeManager：处理帧交换序列，帧聚合、帧重传、保护和确认
  2. ChannelAccessManager：实现了 DCF 和 EDCAF 功能
  3. Txop 和 QosTxop：处理数据包队列
- MAC 高层模型：实现了 Wifi 中非时间关键的过程，如 MAC 层的信标生成、探测和关联状态机，以及一系列速率控制算法。

模块实现

```
1 int Socket::Send(const uint8_t* buf, uint32_t
    size, uint32_t flags);
2 int UdpSocketImpl::Send(Ptr<Packet> p, uint32_t
    flags);
3 int UdpSocketImpl::DoSendTo(Ptr<Packet> p,
    Ipv4Address dest, uint16_t port, uint8_t tos)
    ;
4 route = ipv4->GetRoutingProtocol()->RouteOutput(
    p, header, oif, errno_);
5 m_udp->Send(p->Copy(),header.GetSource(),header.
    GetDestination(),m_endPoint->GetLocalPort(),
    port,route);
6
7 void Ipv4Interface::Send(Ptr<Packet> p, const
    Ipv4Header& hdr, Ipv4Address dest);
```

模块实现

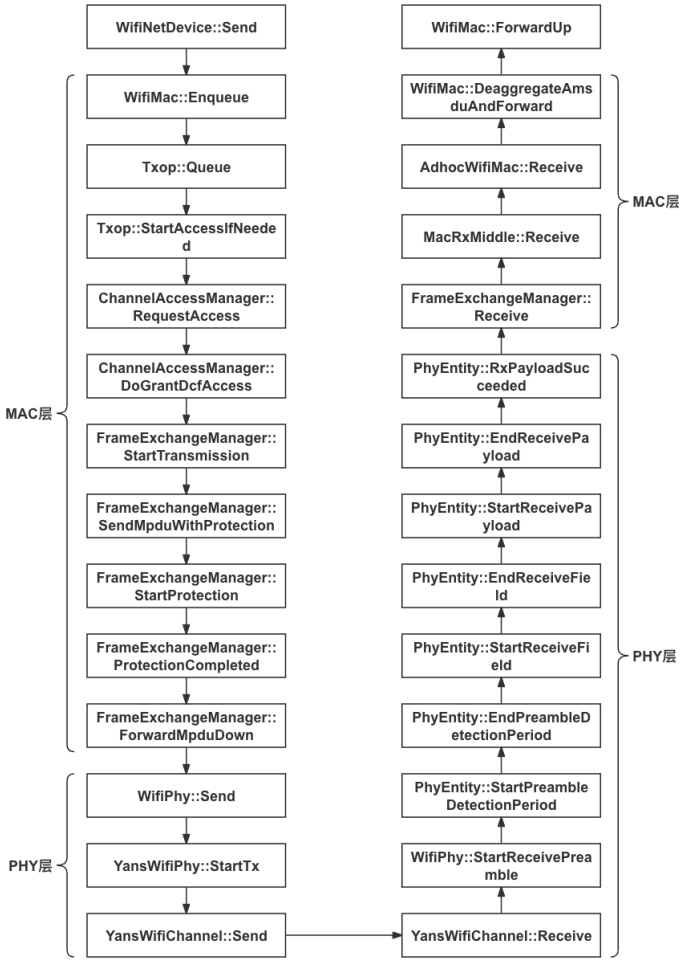

图 3: 数据传递流程图

模块实现

```
89  void YansWifiChannel::Send(Ptr<YansWifiPhy> sender, Ptr<const WifiPpdu> ppdu, double txPowerDbm) const
90  {
91      Ptr<MobilityModel> senderMobility = sender->GetMobility();
92      NS_ASSERT(senderMobility);
93      for (PhyList::const_iterator i = m_phyList.begin(); i != m_phyList.end(); i++)
94      {
95          if (sender != (*i))
96          {
97              // For now don't account for inter channel interference nor channel bonding
98              if ((*i)->GetChannelNumber() != sender->GetChannelNumber())
99              {
100                 continue;
101             }
102             Ptr<MobilityModel> receiverMobility = (*i)->GetMobility()->GetObject<MobilityModel>();
103             Time delay = m_delay->GetDelay(senderMobility, receiverMobility);
104             double rxPowerDbm = m_loss->CalcRxPower(txPowerDbm, senderMobility, receiverMobility);
105             NS_LOG_DEBUG("propagation: txPower="
106                          << txPowerDbm << "dbm, rxPower=" << rxPowerDbm << "dbm, "
107                          << "distance=" << senderMobility->GetDistanceFrom(receiverMobility)
108                          << "m, delay=" << delay);
109             Ptr<NetDevice> dstNetDevice = (*i)->GetDevice();
110             uint32_t dstNode;
111             if (!dstNetDevice)
112             {
113                 dstNode = 0xffffffff;
114             }
115             else
116             {
117                 dstNode = dstNetDevice->GetNode()->GetId();
118             }
119
120             Simulator::ScheduleWithContext(dstNode,
121                                            delay,
122                                            &YansWifiChannel::Receive,
123                                            (*i),
124                                            ppdu,    "ppdu": Unknown word.
125                                            rxPowerDbm);
126         }
127     }
128 }
```

图 4: Channel 模块关键函数

模块实现

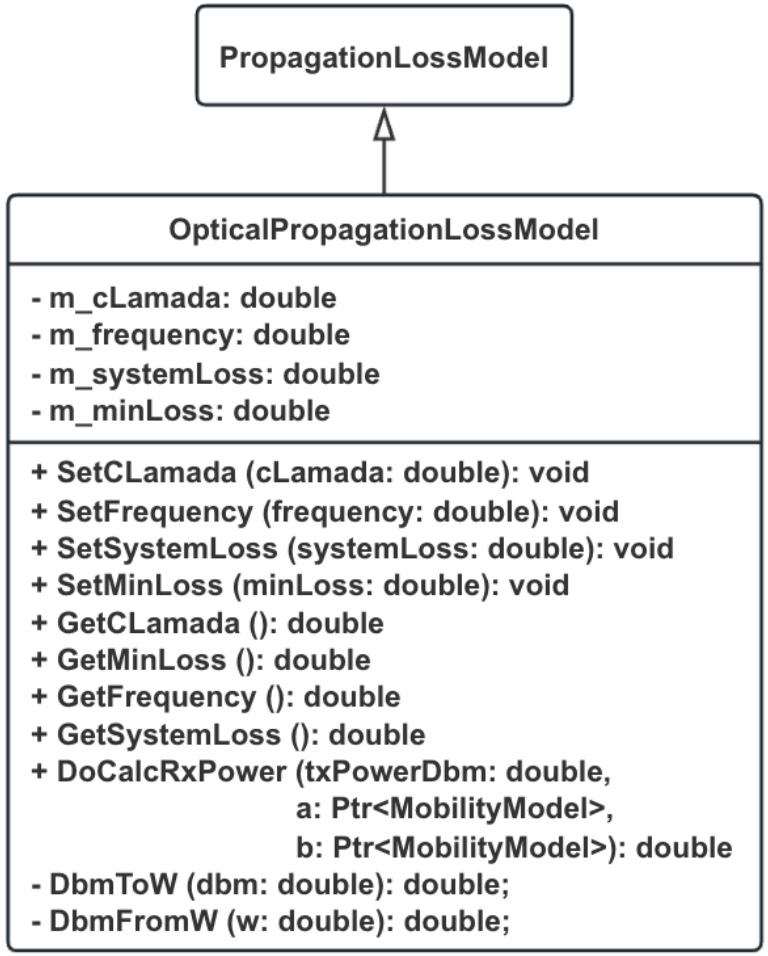

图 5: 传播衰减模型

模块实现

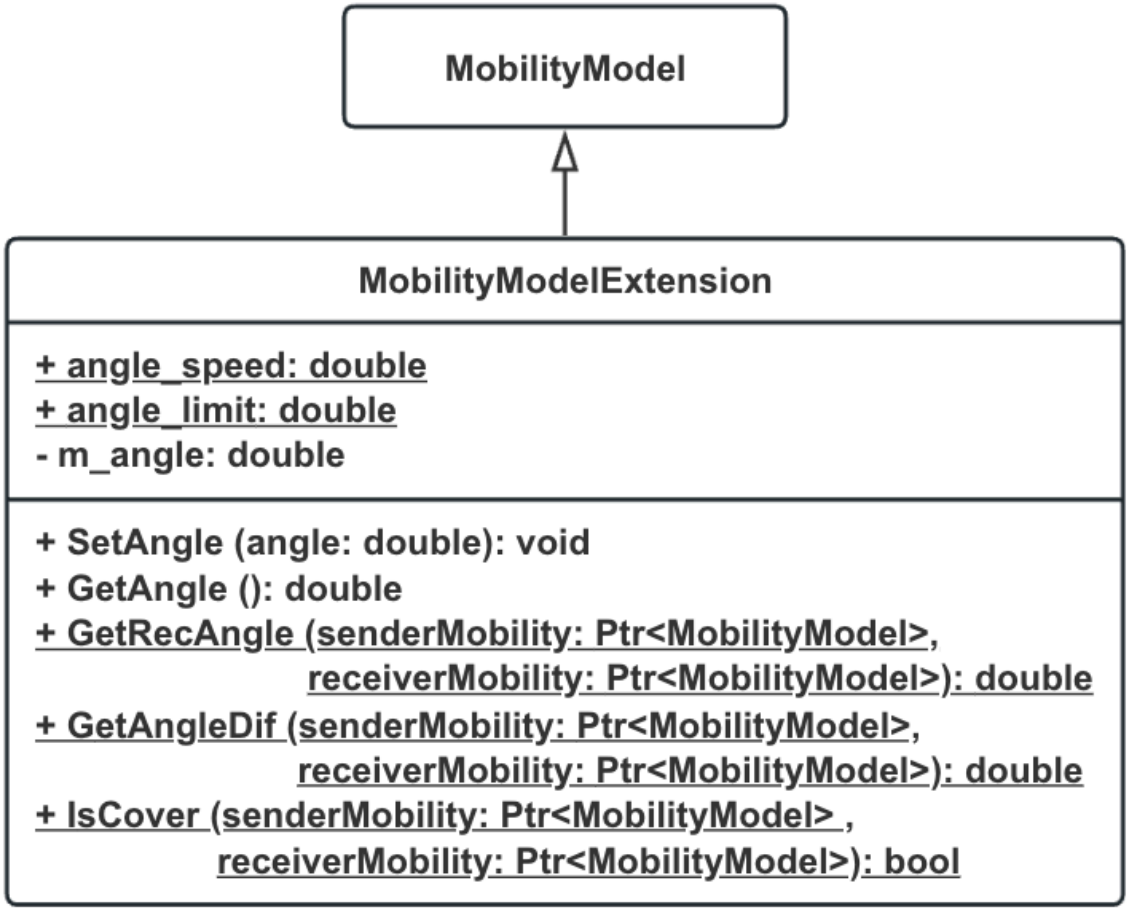

图 6: 定向发送模型

个人推荐

推荐至少预留 2 周时间实验

ns-3 官方文档

- https://www.nsnam.org/documentation/

ns-3 技术博客

- https://jluyeyu.com/categories/ns3/
- https://www.jianshu.com/p/8d419dae1f9f/

gdb 入门

- https://gitsang.github.io/docs/cpp/gdb_quick_start/

Thanks!