LAB 2 利用NS3部署不同拓扑无线网络

051720205 岳雨涵 1617101班

1 实验要求

利用`NS2`/`NS3`部署一个星型无线网络（一个AP，不少于5个接入点）、一个多跳无线网络（不少于6个网络节点）。并测量两种网络拓扑下的网络链路吞吐量，用图表表示。

2 实验环境

操作系统： Linux Ubuntu

网络模拟器： NS3

代码可视化：NetAnim(基于Qt 4)

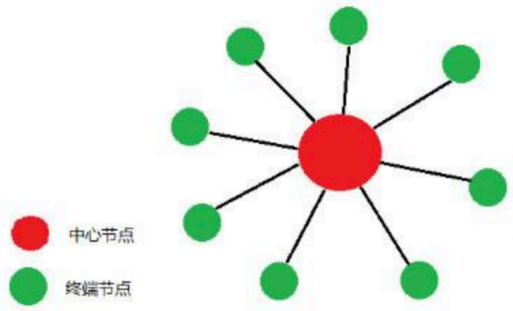
吞吐量计算： Wireshark/ thourought.awk脚本

吞吐量图标表示：Wireshark/ gnuplot

3 实验原理

3.1 星型网络

- 星型网络是有两种网络设备构成：中心节点 (此实验中即AP) 和终端节点。中心节点是整个星型网络的枢纽，所有终端节点通过无线或者有线的形式连接到中心节点，与中心节点进行信息交互。终端节点之间不能直接进行信息交互，只能通过中心节点进行信息转发，从而达到终端节点之间相互通信的作用。拓扑图如下：

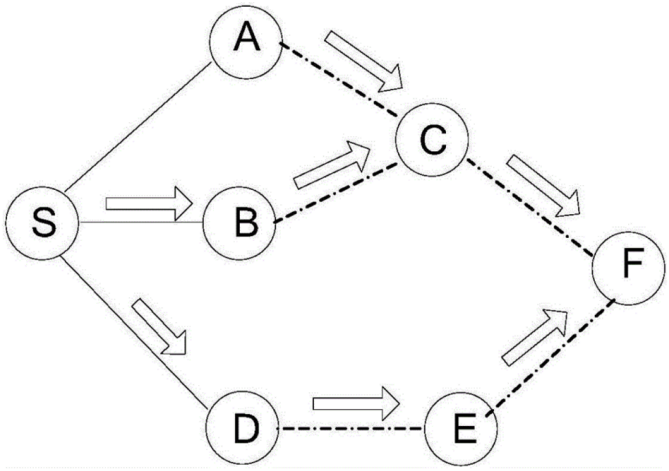


- ns3中实现星型网络: 使用`PointToPointStarHelper Class Reference`

  源码参见： https://www.nsnam.org/doxygen/classns3_1_1_point_to_point_star_helper.html

3.2 多跳网络

- 多跳网络的核心是让网络中的每个节点都发送和接收信号。网络中的大量终端设备连接成网状结构，每个节点都具有自动路由功能，每个节点只和临近节点通信。拓扑图如下：



- 需要初始化信道/wifi物理层.
- reference: http://blog.chinaunix.net/uid-23982854-id-4416998.html

4 实验过程

4.1 环境搭建

4.1.1 安装`ns3`

巨良心的教程：https://blog.51cto.com/joedlut/1825512

安装过程：

1. 先写脚本来安装所有ns3.25的依赖并执行该脚本；
2. 下载ns3 (wget或官网都可)
3. 编译安装ns3 ,执行 `./build.py --enable-tests --enable-examples`
4. 执行 `./waf --run hello-simulator`测试是否安装成功

之后所有的文件全部放入scratch文件夹中使用./waf运行，因为scratch文件夹是ns3的默认脚本存放地址.

4.1.2 安装`NetAnim`

1. 为了使代码可视化，需要安装`NetAnim`.由于它基于qt5，则先执行

```
sudo apt-get install mercurial
sudo apt-get install qt5-default
```

来安装Qt5和Mecurial.

2. 安装完成后使用`qmake`进行编译.

```
cd ns-allinone-3.25/netanim-3.108
qmake ./NetAnim.pro
```

4.1.3 `gnuplot`和`gwak`的安装(Optional)

为了求吞吐量并可视化，先安装上述两个包.

```
sudo apt-get install gnuplot//画图
sudo apt-get install gwak
```

gnuplot是根据`ns3`运行时产生的`trace`文件生成图标的软件，可以通过`trace`文件中的数据，使用gwak和throughout.awk（求吞吐量的awk脚本）来求出吞吐量，并使用gnuplot绘图.

```
gawk -f throughout.awk star.tr > star
cat star//查看是否有数据
gnuplot>>> plot "star" with lines
```

4.1.4 安装`wireshark`

使用`gwak`+`gnuplot`计算吞吐量并绘图的方式太麻烦，直接安装wireshark，在statistic选项卡中即可查看.

安装wireshark直接`sudo apt-get install wireshark`即可.

4.2 星型网络

4.2.1 代码实现

- 创建nSpoke个节点和ap的点对点连接

```
PointToPointHelper pointToPoint;
  pointToPoint.SetDeviceAttribute ("DataRate", StringValue ("5Mbps"));
  pointToPoint.SetChannelAttribute ("Delay", StringValue ("2ms"));
  PointToPointStarHelper star (nSpokes, pointToPoint);//封装好的class
```

- 创建网络协议栈

```
  InternetStackHelper internet;
  star.InstallStack (internet);
```

- 分配ip地址

```
star.AssignIpv4Addresses (Ipv4AddressHelper ("10.1.1.0", "255.255.255.0"));
```

- 创建应用

```
  uint16_t port = 50000;
  Address hubLocalAddress (InetSocketAddress (Ipv4Address::GetAny (), port));
  PacketSinkHelper packetSinkHelper ("ns3::TcpSocketFactory",
hubLocalAddress);
  ApplicationContainer hubApp = packetSinkHelper.Install (star.GetHub ());
  hubApp.Start (Seconds (1.0));
  hubApp.Stop (Seconds (10.0));

  OnOffHelper onOffHelper ("ns3::TcpSocketFactory", Address ());
  onOffHelper.SetAttribute ("OnTime", StringValue
("ns3::ConstantRandomVariable[Constant=1]"));
  onOffHelper.SetAttribute ("OffTime", StringValue
("ns3::ConstantRandomVariable[Constant=0]"));

  ApplicationContainer spokeApps;
```

- 给ap添加接口

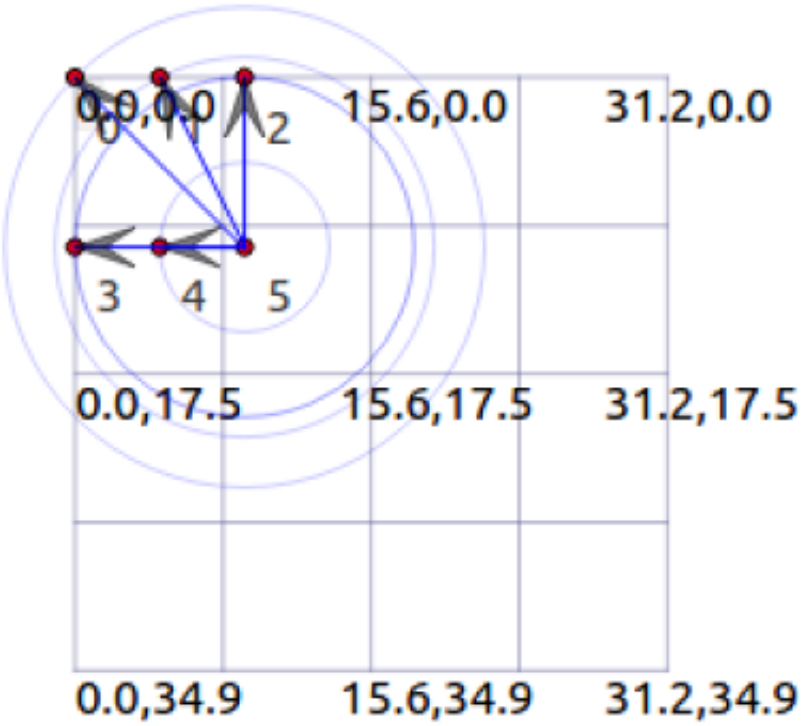
```
for (uint32_t i = 0; i < star.SpokeCount (); ++i)
    {
      AddressValue remoteAddress (InetSocketAddress
(star.GetHubIpv4Address (i), port));
      onOffHelper.SetAttribute ("Remote", remoteAddress);
      spokeApps.Add (onOffHelper.Install (star.GetSpokeNode (i)));
}
```

- 开启路由，因为使用了不同网段

```
Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
```

4.2.2 网络拓扑图

使用`netAnim`打开`star.xml`文件得到网络拓扑图.

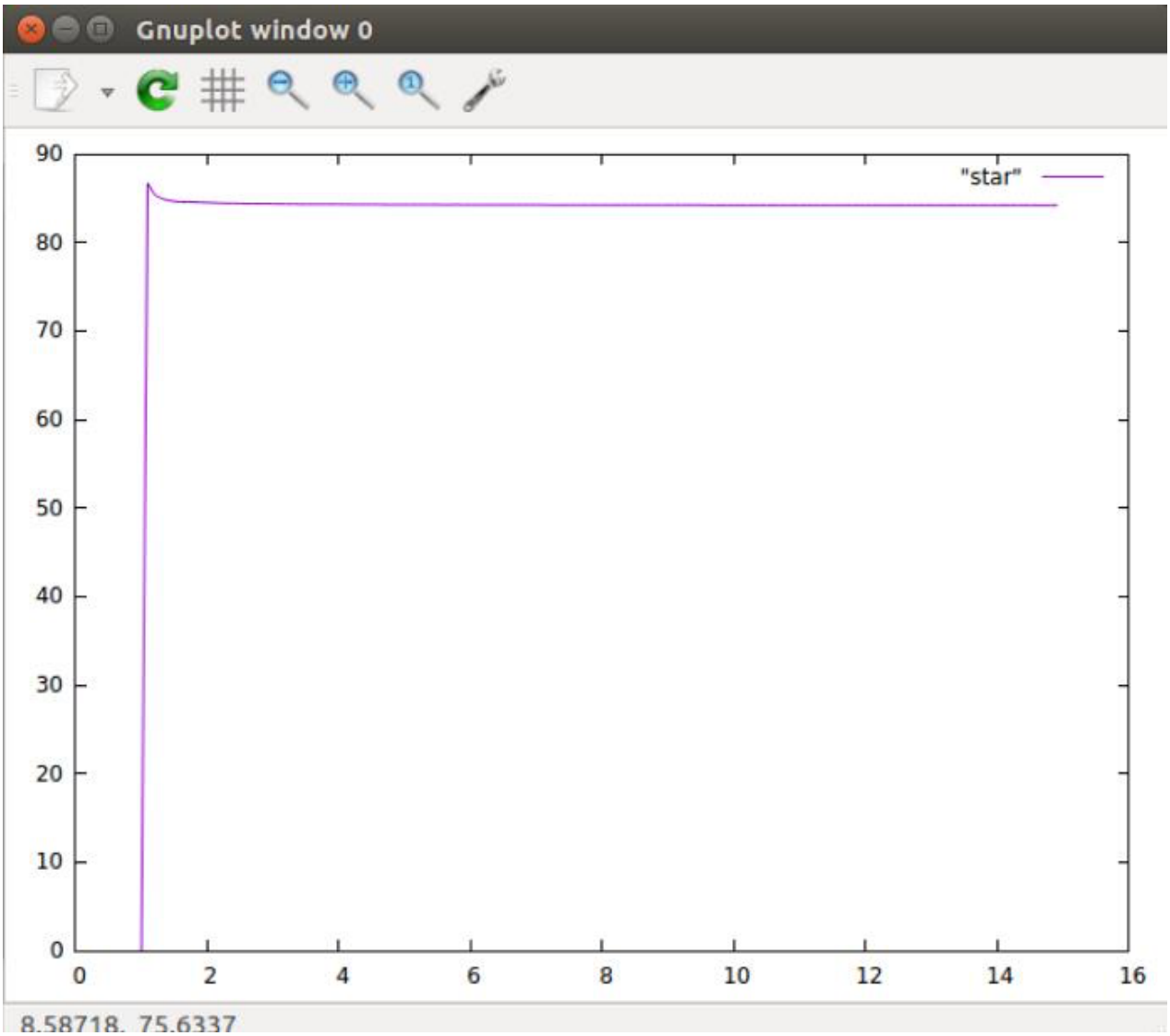


4.2.3吞吐量的计算与绘图 `gwak`+`gnuplot`

在./waf相同目录下运行：

```
gawk -f throughout.awk star.tr > star
gnuplot>>> plot "star" with lines
```

可得到吞吐量的图标表示：



4.3 多跳网络

多跳无线网络没有一个固定的ap，也没有固定的sta，每一个节点都可以成为ap和sta，所以需要建立一个hocwifi网络。整个网络没有固定的基础设施，每个节点都是移动的，每一个节点同时是router，它们能完成发现以及维持到其它节点路由的功能。

4.3.1 代码实现

- 创建信道和物理信息

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

- 创建Wi-Fi

```
WifiHelper wifi;
    wifi.SetStandard(WIFI_PHY_STANDARD_80211a);

 wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager","DataMode",String
Value("OfdmRate6Mbps"));
```

- 位置信息：设置节点在simulate时移动或静止

```
obilityHelper mobility;
    mobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                                   "MinX", DoubleValue (0.0),
                                   "MinY", DoubleValue (0.0),
                                   "DeltaX", DoubleValue (5.0),
                                   "DeltaY", DoubleValue (5.0),
                                   "GridWidth", UintegerValue (10),
                                   "LayoutType", StringValue ("RowFirst"));

    mobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                               "Bounds", RectangleValue (Rectangle (-500,
500, -500, 500)));
    mobility.Install (AdHocNode);
```

- 安装ip协议栈/分配ip地址/创建应用的服务端，客户端与星型网络相似，不再赘述。

4.3.2 网络拓扑图

4.3.3 吞吐量的计算与绘图 `wireshark`

Details

Hardware: Unknown
OS: Unknown
Application: Unknown

Interfaces

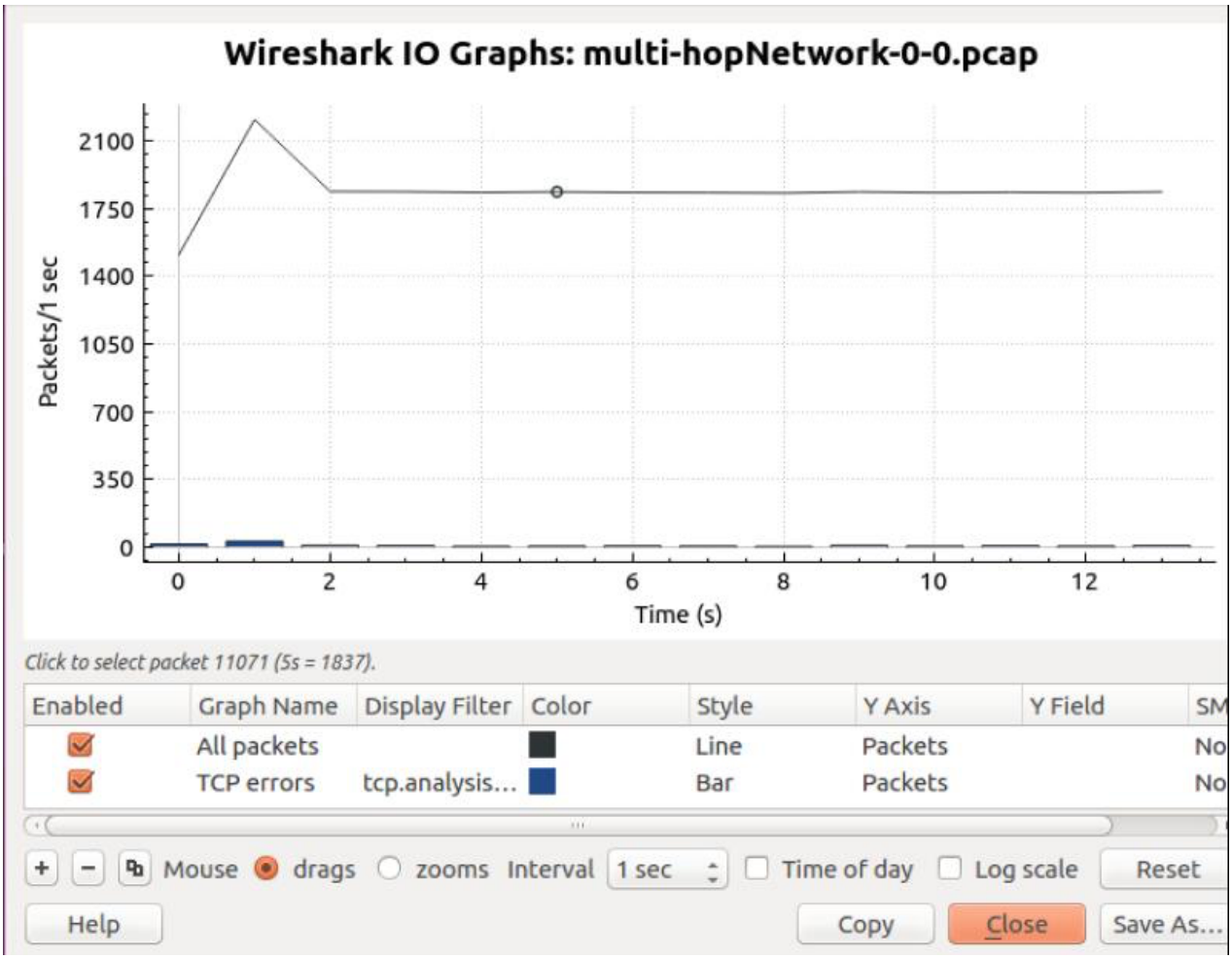
Interface	Dropped packets	Capture filter	Link type	Packet size limit
Unknown	Unknown	Unknown	IEEE 802.11 Wireless LAN	65535 bytes

Statistics

Measurement	Captured	Displayed	Marked
Packets	25744	25744 (100.0%)	—
Time span, s	13.998	13.998	—
Average pps	1839.1	1839.1	—
Average packet size, B	221	221	—
Bytes	5688758	5688758 (100.0%)	0
Average bytes/s	406 k	406 k	—
Average bits/s	3,251 k	3,251 k	—

Capture file comments

Help | Refresh | Copy To Clipboard | Close | Save Comments



Click to select packet 11071 (5s = 1837).

Enabled	Graph Name	Display Filter	Color	Style	Y Axis	Y Field	SM
☑	All packets			Line	Packets		No
☑	TCP errors	tcp.analysis…		Bar	Packets		No

+ − Mouse ◉ drags ○ zooms Interval 1 sec ☐ Time of day ☐ Log scale Reset

Help | Copy | Close | Save As…

附录1 吞吐量计算脚本`throughout.awk`源码

```
BEGIN {
  init=0;
  cnt = 0;
  FS="[() \t]";#field seperator is ')' or'('or ' '
  myScrIP = "10.1.3.5";#This is the link that we pay attention to
  myDstIP = "10.1.3.1";
}
{
  action = $1;
  time = $2
  namespace=$3;
  #printf("%d\n",NF);
  for (i=1;i<=NF;i++)#find packet ID
  {
    if ($i ~ /id/) #if $i field matches "id"
           myPacketID = $(i+1);#record the id of the packet for future use
    else if ($i ~ /length:/) #if $i field matches "length:"
           myLength =  $(i+1);#record the length of the packet for future use
    else if ($i ~ />/) #if $i field matches ">"
    {
            srcIP = $(i-1);
            dstIP = $(i+1);
            if(match(srcIP, myScrIP) && match(dstIP, myDstIP) )#link matches
            {
        packet_id = myPacketID;
                pktsize = myLength;
                #record send time of the packet
                if (start_time[packet_id]==0)
                {
          start_time[packet_id]=time;
                }
                if (action=="r")
                {
                     if(end_time_packet[packet_id] ==0 )#the first time we
receive this packet
                    {
                        end_time_packet[packet_id] = time;#record time
according to id
                        packetCNT[packet_id]=cnt;
                        pkt_byte_sum[cnt+1]=pkt_byte_sum[cnt]+ pktsize;
                        end_time[cnt]=time;
                        cnt++;
                    }#if(end_time_packet[packet_id] ==0)
```

```
          else#not the 1st time we receive this packet,so we update receive
time
                    {
                    #printf("*****duplicate packetID:
%s,cnt=%s,end_time_old=%s,end_time new:
%s\n",packet_id,cnt,end_time[packetCNT[packet_id]], time);
                      end_time[packetCNT[packet_id]]=time;
          }
                }#if (action=="r")
            }#if match(srcIP, myScrIP)

        }#else if ($i ~ />/) #if $i field matches ">"
  }#for (i=1;i<=NF;i++)#find packet ID
}

END {
        printf("%s\t%s\n", end_time[0], 0);
        for(j=1 ; j<cnt;j++){
            throughput = (pkt_byte_sum[j] / (end_time[j] -
start_time[0]))*8/1000;
            printf("%s\t%s\n", end_time[j], throughput );
        }
}
```