广州大学

课程考查报告

课程名称：	无线网络
学　　院：	机械与电气工程学院
专　　业：	电信　电信　通信
班级名称：	1 班
学生姓名：	苏伟强　周宇恒　林敏菲
学　　号：	1507400051　1407400106　1507600006
指导老师：	刘外喜
完成日期：	2018.7

目录

1 任务要求与分工

1.1 任务要求

通过 *NS2(Network Simulator version 2)* 进行无线网络仿真，无线网络结构为 $Ad-hoc$，使用 *AODV* 路由协议，媒体接入方式为 *DCF* 或 *PCF*，分别使用三种不同的 *MAC* 协议对网络进行仿真，即 *802.11a, 802.11b, 802.11g*。各个节点以某种模式发送数据包，发送速率，发送模式自定。各个节点可以固定也可以移动。运行仿真，通过 *trace* 数据文件，分析 *AODV* 的运行过程，比较 3 种不同的 *MAC* 协议的性能，并且说明原因。

1.2 分工

小组分工见表1

表 1: 小组分工表

组员	分工内容
苏伟强	802.11a/b/g 参数收集,编写仿真程序（*.tcl*）,*.tr* 仿真文件网络吞吐量分析（*Matlab*）,AODV 过程分析，报告撰写
周宇恒	*ns2* 安装，无线网络仿真架构以及程序资料的收集以及初步仿真，仿真数据分析相关资料的收集，*.tr* 仿真文件网络丢包率，网络时延分析。
林敏菲	对仿真结果，分析比较 802.11a/b/g 各协议性能

2 理论介绍

2.1 WLAN 的定义

WLAN(无线局域网)，通常指采用无线传输介质的计算机局域网，其利用无线电等无线方式，提供对等或者点对点的数据通信 [1]。

2.2 IEEE 802.11 协议体系结构

2.2.1 IEEE 802.11 层次结构

802.11 是 *WLAN* 主要的标准之一，其层次结构如图1所示 [2]，物理层包括物理汇聚子层（PLCP，负责将 MAC 帧对映到传输介质），物理介质相关子层（PMD，负责传送 MAC 帧）。数据链路层分为介质访问控制子层（MAC），逻辑链路控制子层（LLC）。其中 MAC 子层主要控制节点获取信道访问权，而 LLC 子层负责建立和释放逻辑连接，提供高层接口，差错控制等。

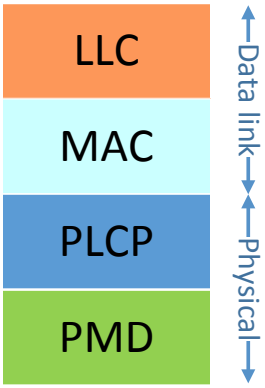

图 1: IEEE 802.11 层次结构

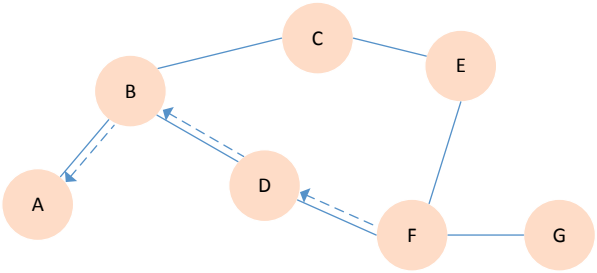

图 2: AODV 示意图

物理层定义了包括直接序列扩频，OFDM 等规范。

MAC 层接入协议包括 DCF 和 PCF 两种。DCF 是基础协议，核心是 CSMA/CA；PCF 用于超帧无竞争期，是可选协议。为避免冲突，MAC 子层规定所有站在完成发送后必须等待一个短时间才能发送下一帧，该时间称为 IFS，IEEE 定义了 4 种 IFS 以实现不同的访问优先级控制，其时间长度关系为 SIFS<PIFS<DIFS<EIFS。

其基本帧结构如图3所示，其中 Preamble 是一个前导标识，用于接收设备识别 802.11，而 PLCP 域中包含一些物理层的协议参数，显然 Preamble 及 PLCP 是物理层的一些细节。MAC 层处理的是帧数据。[3]

图 3: 802.11 基本帧结构

5 种物理层标准（直序扩频 DSSS、跳频扩频 FHSS、正交频分复用 OFDM、高速率直序扩频 HR/DSSS、红外 TR），也就是 5 种编码与调制方法，每种对应的物理层帧格式都是不同的。也就是说，虽然这个 802.11 标准对外的接口（MAC）是一样的，但是根据底层采用的不同物理层标准，底层的从帧格式到编码、调制都是不一样的。

2.2.2 IEEE 802.11 若干子标准简介

802.11 不断发展，包括多个子标准，如表2所示，为部分标准的性能比较

表 2: 部分 802.11 子标准性能比较

子标准	802.11a	802.11b	802.11g	802.11 n
物理层	OFDM	DSSS	OFDM	OFDM
频带	5GHz	2.4GHz	2.4GHz	2.4/5GHz
传输速率	6 54Mbps	1,2,5.5,11Mbps	最高 54Mbps	最高 600Mbps
传输距离	室内 30m 室外 45m	室内 30m 室外 100m	室内 30m 室外 100m	室内 70m 室外 250m

3 Ad Hoc 网络

WLAN 的协议 802.11 定义了两种工作模式：Ad Hoc （对等）模式和 Infrastructure （架构）模式。在 Ad Hoc 模式中，至少需要包含两个 STA ，每两个 STA 之间直接相连实现资源共享，不需要 AP 和分布式系统。Ad Hoc 网络是一种非蜂窝对等式无线移动网络，广泛应用于各种无线通信场所。

3.1 MANET

MANET 是一种移动式 Ad Hoc 网络，可以在有限范围内实现多个移动节点临时互联互通的网络，是一种灵活的通信方式。

3.2 AODV 路由协议

AODV（Ad Hoc On-demand Distance Vector）是 MANET 的一种路由协议，是一种按需路由 [4]。采用逐跳转发 RREQ 分组，加入组播路由协议扩展，从路由查找回复 RREP。其过程如图2所示，这里假定 A 需要与 G 通信，RREQ 请求包由 A 创建，所有节点都会转发 RREQ，直到自身有到 G 的路由或者有直接邻居 G，实线表示 RREQ 路由，虚线表示 RREP 逆向路由。

4 系统设计与实现

该系统的设计与实现，分析遵从图4流程，其中仿真模型参数配置包括：网络组件配置，节点配置等。[5]

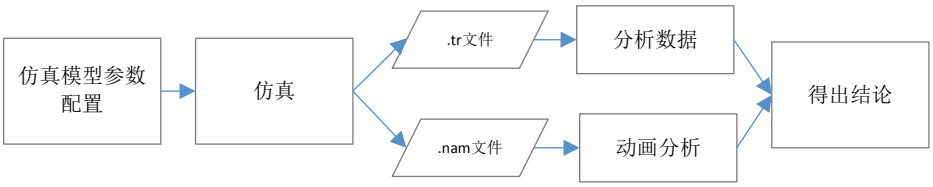

图 4: 系统实现流程图

4.1 802.11a/b/g 仿真

仿真目标：创建一个简单的无线场景：100 个固定节点，定义边界为 200m*300m 的区域，在 100 个节点中建立 30 对 TCP 连接，并且使用 FTP 进行数据传输。[5]。仿真并且进行生成追踪文件（后缀.tr），仿真动画文件（后缀.nam），编写脚本语言进行数据分析。

如图4，首先进行仿真模型参数配置。配置网络组件，在无线模拟开始时，我们需要为每个网络组件定义类型，如下表3所示意；同时，配置无线节点，如表4，具体实现见附录代码：为了可以仿真 802.11a/b/g，我们需

表 3: 仿真网络组建配置

组件名称 (变量名)	配置
通道类型（chan）	无线通道
天线传播类型（prop）	两径模型 [7]
网络接口类型（netif）	无线物理层
MAC 类型 (mac)	IEEE 802.11
接口队列类型 (ifq)	尾部丢弃，优先级排队 [8]
链路层类型 (ll)	LL [10]
天线类型 (ant)	全向天线 [9]
接口队列最包长 (ifqlen)	50
移动节点数 (nn)	100
路由协议 (rp)	AODV
拓扑边界长度 (x)	300m
拓扑边界宽度 (y)	200m

表 4: 无线节点配置

参数	配置
Ad Hoc 路由协议 (adhocRouting)	AODV
链路层类型（llType）	LL
MAC 类型（macType）	IEEE 802.11
接口队列长度（ifqLen）	50
接口队列类型（ifqType）	尾部丢弃，优先级排队
天线类型（antType）	全向天线
天线传播类型（propType）	两径模型
网络接口类型（phyType）	无线物理层
通道类型（channelType）	无线通道
拓扑实例	见附录程序
应用层追踪	ON
网络层追踪	ON
MAC 层追踪	ON
移动追踪	OFF

要修改表3中 MAC 类型以及网络接口类型（物理层）中的参数，具体参数见表格5，详细请见附录代码。

配置好节点之后创建节点，初始化节点位置，节点间距为 30m，建立 30 对 TCP 连接，FTP 连接（从第 0.1s 开始发送数据到仿真结束前 1s 停止发送数据包），开始仿真，生成.tr 文件，.nam 文件。如表4所示，这里设置了应用层追踪，网络层追踪，MAC 层追踪，所以最后产生的.tr 文件必定是含有应用层，网络层还有 MAC 层的追踪条目。结果也验证了我们的猜想，如代码片段 AODV100.tr 所示为仿真结束后生成的追踪文件的（后缀.tr）的部分记录。一下对追踪记录的各个部分进行解释。

各个位分别表示：事件 时间 _ 节点 _ 层次 —— 分组 UID 分组类型 分组大小 [MAC 层信息] —— [IP 层信息]。其中事件，有 4 种类型：s,r,d,f，分别表示分组的发送，接收，丢弃，转发事件；节点，表示事件发生的节点；层次，表示事件发生的层次：有 AGT （业务层）RTR （路由层）MAC（MAC 层）；MAC 层信息，包含四项：a、发送节点在无线信道上发送该分组所期望的时间，16 进制表示，单位为秒，b、接收节

点的 MAC 地址，c、发送节点的 MAC 地址，d、MAC 层封装的分组类型，0x800 表示 IP 分组，0x806 表示 ARP 分组；IP 层信息，包含四项：a、分送分组的源节点地址，格式为“节点号：端口号”，b、接收分组的目的节点地址，格式为“节点号：端口号”，c、分组的 TTL 值，d、源节点到目的节点的跳数。[15] 我们观察到，还有带有 AODV 字样的记录条目，这是 AODV 的路由信息，其中标记为 REQUEST 的为路由请求，标记为 REPLY 的为路由回复。

知道了.tr 文件格式，我们就可以编写脚本来分析网络性能了。

表 5: 802.11a/b/g 物理层，MAC 层参数

	参数	802.11a	802.11b	802.11g
物理层	频带/Hz（freq_）	$5*10^9$	$2.4*10^9$	$2.4*10^9$
	通信感知范围 [13]（CSThresh_）	$3.17291*10^9$	$2.79*10^9$	$2.79*10^9$
	可通信范围 [13]（RXThresh_）	$6.5556*10^{10}$	$5.76*10^9$	$5.76*10^9$
	传输功率/W（Pt_）	0.281838	0.281838	0.281838
MAC 层	竞争窗口最小值 [12]/s（CWMin）	15	31	15
	竞争窗口最大值 [12]/s（CWMax）	1023	1023	1023
	时隙 [11]/s(SlotTime_)	0.00005	0.00002	0.000009s
	SIFS[14][11]/s(SIFS_)	0.000016	0.00001	0.000016s
	前导码长度 [14](PreambleLength)	96	144	120
	PLCP 头部长度 [14]PLCPHeaderLength_）	24	48	24
	PLCP 数据率 [14]/bps（PLCPDataRate_）	$6*10^6$	$1*10^6$	$6*10^6$
	最高速率 [14]/bps（dataRate）	$5.4*10^7$	$1.1*10^7$	$5.4*10^7$
	最低速率 [14]/bps（basicRate_）	$6*10^6$	$1*10^6$	$6*10^6$

AODV100.tr

```
1
2  s 0.100000000 _0_ AGT  --- 0 tcp 40 [0 0 0 0] ------- [0:0 50:0 32 0] [0 0] 0 0
3  r 0.160559345 _50_ AGT  --- 0 tcp 40 [3f 32 28 800] ------- [0:0 50:0 26 50] [0 0] 5 0
4  s 0.160559345 _50_ AGT  --- 1 ack 40 [0 0 0 0] ------- [50:0 0:0 32 0] [0 0] 0 0
5  r 0.167370926 _0_ AGT  --- 1 ack 40 [3f 0 a 800] ------- [50:0 0:0 26 0] [0 0] 5 0
6  s 0.167370926 _0_ AGT  --- 2 tcp 1540 [0 0 0 0] ------- [0:0 50:0 32 0] [1 0] 0 0
7  s 0.167370926 _0_ AGT  --- 3 tcp 1540 [0 0 0 0] ------- [0:0 50:0 32 0] [2 0] 0 0
8  s 0.200000000 _1_ AGT  --- 4 tcp 40 [0 0 0 0] ------- [1:0 51:0 32 0] [0 0] 0 0
9  r 0.235411595 _50_ AGT  --- 2 tcp 1540 [3f 32 28 800] ------- [0:0 50:0 26 50] [1 0] 5 0
10 s 0.235411595 _50_ AGT  --- 5 ack 40 [0 0 0 0] ------- [50:0 0:0 32 0] [1 0] 0 0
11 s 0.300000000 _2_ AGT  --- 6 tcp 40 [0 0 0 0] ------- [2:0 52:0 32 0] [0 0] 0 0
12 D 0.354898690 _3_ IFQ  ARP 0 AODV 44 [3f 3 3 800] ------- [51:255 1:255 24 2] [0x4 7 [51 4] 10.000000] (REPLY)
13 s 0.367370926 _0_ AGT  --- 7 tcp 1540 [0 0 0 0] ------- [0:0 50:0 32 0] [1 0] 0 0
14 r 0.367857815 _0_ AGT  --- 5 ack 40 [3f 0 a 800] ------- [50:0 0:0 26 0] [1 0] 5 0
15 s 0.367857815 _0_ AGT  --- 8 tcp 1540 [0 0 0 0] ------- [0:0 50:0 32 0] [2 0] 0 0
16 ......
17 ......
```

4.2 AODV 仿真

为了更加清楚的观察到 AODV 从逐跳广播 RREQ 路由到找到路由之后沿着逆路由回传 RREP 的过程，在进行仿真的时候，沿用前述 802.11 仿真的架构，仅适用 802.11g 进行仿真（因为这里仅需观察 AODV 路由，参考节点间距，适用那种协议可以适当调整）。但是改为 8 个节点排位一行的拓扑结构，如图11建立节点 2 与节点 6 的 TCP 连接并进行 FTP 通信（数据方向 2->6），由于 AODV 是按需路由，只有在需要的时候才会发，所以在节点 2 寻找节点 6 路由的过程中，可以很明显的观察其 AODV 过程。这里我们是通过仿真生成的.tr 文件来分析 AODV 过程。

值得注意的是，为了方便我们分析 AODV 的过程，在仿真的时候，我们只接收网络层的数据（在节点配置中将 routerTrace 调为 ON，其余 OFF）。详情请见附录代码。

5 实验分析

5.1 实验设置

5.1.1 电脑配置

CPU: *Intel*® Core i7-4710MQ @2.50GHz, 运行内存 8G *Kingston*® ，64 位 *Windows*®7 操作系统。ns2 仿真基于 VMWARE 虚拟机平台，运行 64bit-ubuntu 16.04LTS。

5.1.2 实验配置

在第 4.1 节所述的 802.11a/b/g 的仿真实验中，其网络拓扑结构为网格排列的 100 个固定节点，节点间距为 30m，拓扑图如图6。算法中具体参数设置已经在第 4.1 节有详细叙述。该实验设置以下三个实验：1. 其他参数不变，不同协议吞吐量对比；2. 其他参数不变，改变节点个数对网络丢包率，时延的影响；3. 其他参数不变，改变网络中 FTP 代理数对网络丢包率，时延的影响。使用脚本分析，输出数据在 *Matlab* 中进行绘图，详细脚本文件，请见附录代码。

在第 4.2 节所述的 AODV 仿真实验中，其网络拓扑如图11所示为均匀水平排列的 8 个固定节点，为了配合 802.11g，节点间距为 60m，以确保节点广播只能临节点接收到，算法中具体参数设置已经在第 4.2 节有详细叙述。。

5.2 实验结果

5.2.1 802.11a/b/g 仿真分析

如第 4.1 节配置仿真模型，运行仿真，生成追踪文件（.tr 后缀）。这里使用命令行接收参数 p 可以为 a/b/g，表示进行 802.11a/b/g 的仿真。命令行格式为

命令行参数

```
1 $ ns AODV100.tcl [p]
```

参考第 4.1 节所述.tr 文件格式，编写 *Matlab* 脚本，*awk* 脚本，分析数据，绘制折线图。如图5-10。

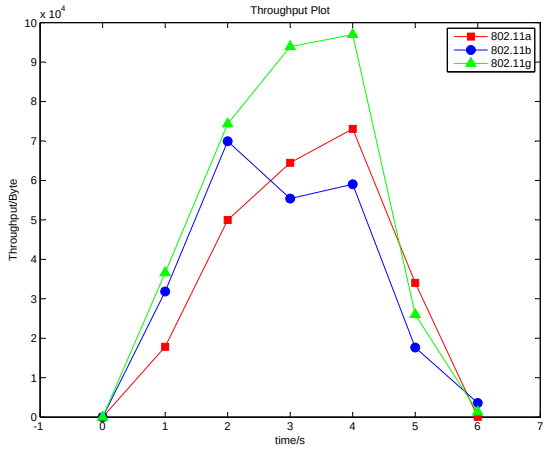

图 5: 不同协议吞吐量对比

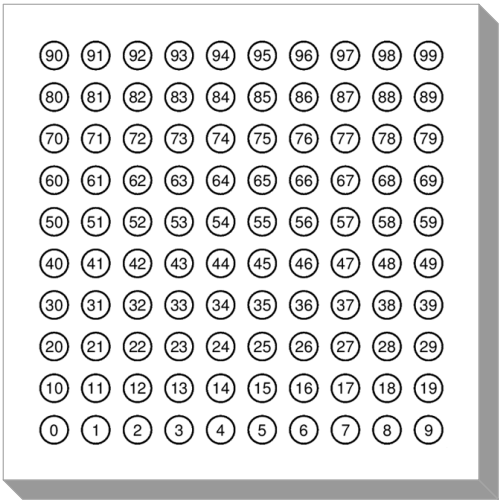

图 6: 802.11a/b/g 仿真拓扑图

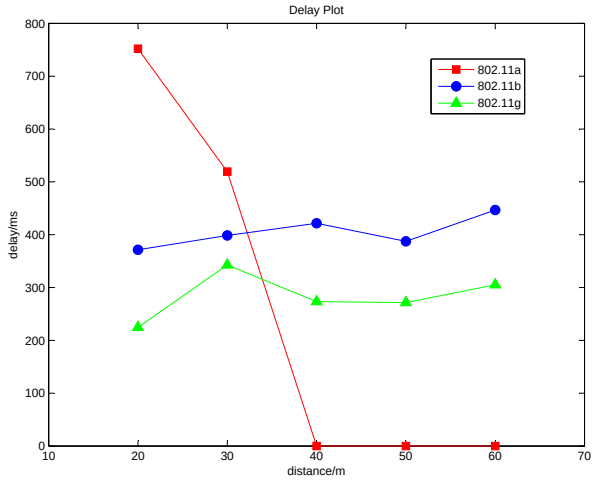

图 7: 不同协议下不同节点间距时延的对比

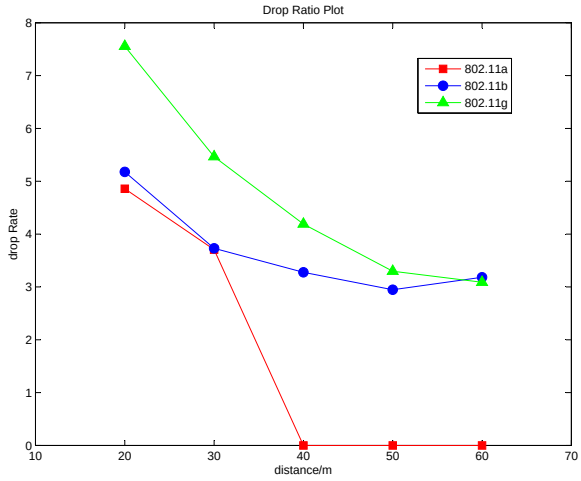

图 8: 不同协议下不同节点间距丢包率的对比

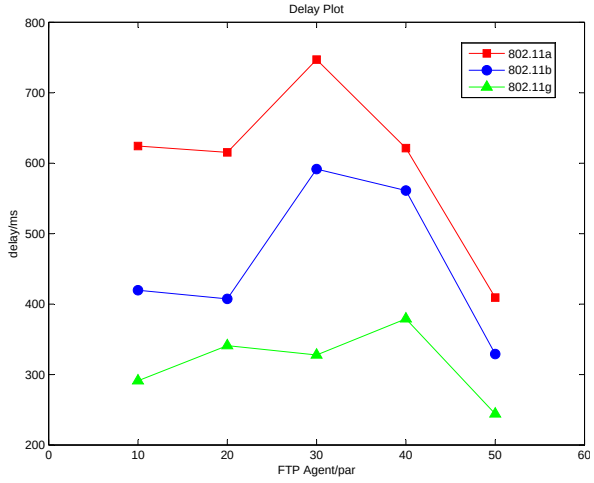

图 9: 不同协议在不同 FTP 连接数下时延对比

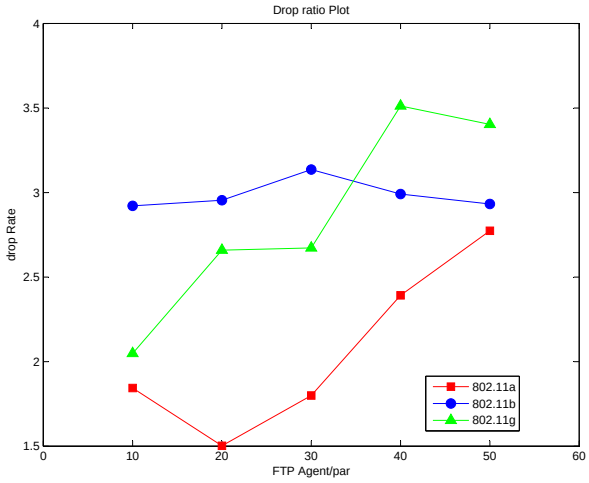

图 10: 不同协议在不同 FTP 连接数下丢包率对比

如图5所示为不同 MAC 协议下全网数据吞吐量的对比图，横坐标为时间段，例如 1 就表示第 1 秒内，可以看吞吐量 802.11g > 802.11a > 802.11b。这是因为 802.11g 和 802.11a 虽然都具有最高 54MHz 的速率，并且都采用了 OFDM 系统，但是如表2所示，802.11g 得益于工作于 2.5GHz 频段，具有更强的信号抗干扰能力，可以传输更远的距离，所以有更高的数据吞吐量。相比较之下，虽然 802.11b 也工作在 2.5GHz 频段，有较强的抗干扰能力，较远的传输距离，但是由于采用了 DSSS 直接扩频，传输速率不占有优势，吞吐量也受到影响。802.11a 工作频率更高，达到 5GHz，其抗干扰能力较差，但是得益于 OFDM，其传输速度较高，综上所述，其吞吐量稍微优于 802.11b。

如图7所示为，不同协议在不同节点间距下时延的对比，网络的 TCP 连接数不变。网络节点间距相同的时候，可以看到，时延：802.11a > 802.11b > 802.11g，这是因为 802.11a 的工作在 5GHz 的频带上，信号衰减快，传输距离短，所以这里其表现出最高的时延。另外，因为传输速度过慢，802.11b 传输时延也比较大，但是要优于 802.11a，这是因为其传输距离的优势在一定的情况下弥补了传输速度的劣势。802.11g 的表现最好，得益于其更高的传输速率，还有更大的传输范围。当节点间距增大的时候，802.11b 和 802.11g 表现稳定，时延略有上升，802.11a 在节点间距为 40m 的时候已经没办法正常通信，这是因为其工作在 5GHz 频段，抗干扰能力较弱的原因，而 802.11b 和 802.11g 工作在 2.5GHz 上，抗干扰能力较强，传输距离较远。

如图8所示为，不同协议在不同节点间距下丢包率的对比，网络的 TCP 连接数不变。网络节点间距相同的时候，可以看到，丢包率 802.11g > 802.11b > 802.11a，其中 802.11a 在信道可用性方面更具优势。这是因为 802.11a 工作在更加宽松的 5GHz 频段, 拥有 12 条非重叠信道。而 802.11g 只有 11 条 (802.11b 同为 11 条), 并且只有 3 条是非重叠信道 (信道 1、信道 6、信道 11)。因此 802.11g 在协调邻近接入点的特性上不如 802.11a。

由于 802.11a 的 12 条非重叠信道能给接入点提供更多的选择, 因此它能有效降低各信道之间的冲突问题 [17]。当节点间距变大的时候，三种协议的丢包率下降，这是因为相同时间接收到的包少了，丢失的包也变少了，但是发送的包数基本不变，导致丢包率下降。802.11a 在 40m 的时候已经无法正常通信。

如图9所示为不同协议在不同 FTP 连接数下的时延对比，节点数量和位置固定。FTP 连接数相同的时候，可以看到时延 $802.11a > 802.11b > 802.11g$，这是因为 802.11a 的工作在 5GHz 的频带上，信号衰减快，传输距离短，并且，该仿真节点距离为 30m，本身对 802.11a 的传输也不是很有利，所以这里其表现出最高的时延。另外，因为传输速度过慢，802.11b 传输时延也比较大，但是要由于 802.11a，这是因为其传输距离的优势在一定的情况下弥补了传输速度的劣势。802.11g 的表现最好，得益于其更高的传输速率，还有更大的传输范围。当 FTP 连接数上升的时候，系统通信压力上升，802.11g 由于其有更好的传播距离，系统容量，时延的表现稳定且优秀，同样是有较高的系统容量，802.11a 由于通信的距离问题，时钟达不到比较好的时延表现，802.11b 表现中规中矩，其传输速率，系统容量的优势并不占优，但是其传输距离要大于 802.11a。

如图10所示为不同协议在不同 FTP 连接数下的丢包率对比，节点数量和位置固定。FTP 连接数相同的时候，802.11a 有较低的丢包率，可靠度比较高，这是因为 802.11a 的 12 条非重叠信道能给接入点提供更多的选择, 因此它能有效降低各信道之间的冲突问题 [17]，反之 802.11b 和 802.11g 都只有 11 条，不具备有这样的特性。随着 FTP 连接数的增加，网络流量增加，802.11a 系统容量的优势得以体现，表现了最低的丢包率，而 802.11g 虽然也有比较大的容量，但是其传输速度慢，丢包率高，802.11b 表现稳健，随着流量数的增大，其丢包率没有太大波动。

5.2.2 AODV 仿真分析

如第 4.2 节配置仿真模型，运行仿真，生成追踪文件（.tr 后缀）和仿真动画文件（.nam 后缀）。打开.nam 文件查看拓扑图，如图11所示，节点间距离为 60m。

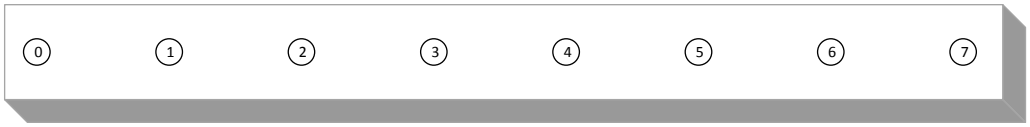

图 11: AODV 仿真网络拓扑

仿真，输出.tr 文件，如下所示，分析 AODV 的过程。

设置了 2 节点发送数据给 6 节点，如下所示，展示了节点 2 寻找节点 6 的路由的过程。首先节点 2 广播 RREQ（第 0.5s），1 节点和 3 节点同时收到了 RREQ，1 节点和 3 节点由于也不知道节点 6 的路由，遂继续转发 RREQ，于是节点 0 和节点 2，节点 2 和节点 4，都收到了 RREQ，节点 2 由于是 RREQ 发起者，忽略该 RREQ，节点 0 和节点 4 由于也没有到 6 的路由，继续转发 RREQ，节点 3 和节点 5 收到了 RREQ，节点 5 有到节点 6 的路由，节点 5 继续广播，当节点 6 通过邻节点的广播，接收到节点 5 的 RREQ 后，返回 RREP 消息给节点 2 以建立通信 (第 0.511309270)。首先按照路由搜索到的路径返回 RREP 给节点 5，然后收到 RREP 的节点继续按照路由搜索到的路径（6→5→4→3→2）依次将 RREP 消息返回给节点 2。节点 2 接收到 RREP 后知道节点 6 的路由，开始 FTP 传输数据 (第 0.523862870s)。

```
 1  s 0.500000000 _2_ RTR  --- 0 AODV 48 [0 0 0 0] ------- [2:255 -1:255 30 0] [0x2 1 1 [6 0] [2 4]] (REQUEST)
 2  r 0.500786200 _1_ RTR  --- 0 AODV 48 [0 ffffffff 2 800] ------- [2:255 -1:255 30 0] [0x2 1 1 [6 0] [2 4]] (REQUEST)
 3  r 0.500786200 _3_ RTR  --- 0 AODV 48 [0 ffffffff 2 800] ------- [2:255 -1:255 30 0] [0x2 1 1 [6 0] [2 4]] (REQUEST)
 4  s 0.501667427 _1_ RTR  --- 0 AODV 48 [0 ffffffff 2 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [6 0] [2 4]] (REQUEST)
 5  r 0.502435627 _0_ RTR  --- 0 AODV 48 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [6 0] [2 4]] (REQUEST)
 6  r 0.502435627 _2_ RTR  --- 0 AODV 48 [0 ffffffff 1 800] ------- [1:255 -1:255 29 0] [0x2 2 1 [6 0] [2 4]] (REQUEST)
 7  s 0.505197934 _3_ RTR  --- 0 AODV 48 [0 ffffffff 2 800] ------- [3:255 -1:255 29 0] [0x2 2 1 [6 0] [2 4]] (REQUEST)
 8  r 0.506074134 _2_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 29 0] [0x2 2 1 [6 0] [2 4]] (REQUEST)
 9  r 0.506074134 _4_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [3:255 -1:255 29 0] [0x2 2 1 [6 0] [2 4]] (REQUEST)
10  s 0.509164924 _4_ RTR  --- 0 AODV 48 [0 ffffffff 3 800] ------- [4:255 -1:255 28 0] [0x2 3 1 [6 0] [2 4]] (REQUEST)
11  r 0.510041124 _3_ RTR  --- 0 AODV 48 [0 ffffffff 4 800] ------- [4:255 -1:255 28 0] [0x2 3 1 [6 0] [2 4]] (REQUEST)
12  r 0.510041124 _5_ RTR  --- 0 AODV 48 [0 ffffffff 4 800] ------- [4:255 -1:255 28 0] [0x2 3 1 [6 0] [2 4]] (REQUEST)
13  s 0.510469070 _5_ RTR  --- 0 AODV 48 [0 ffffffff 4 800] ------- [5:255 -1:255 27 0] [0x2 4 1 [6 0] [2 4]] (REQUEST)
14  s 0.510600387 _0_ RTR  --- 0 AODV 48 [0 ffffffff 1 800] ------- [0:255 -1:255 28 0] [0x2 3 1 [6 0] [2 4]] (REQUEST)
15  r 0.511309270 _4_ RTR  --- 0 AODV 48 [0 ffffffff 5 800] ------- [5:255 -1:255 27 0] [0x2 4 1 [6 0] [2 4]] (REQUEST)
16  r 0.511309270 _6_ RTR  --- 0 AODV 48 [0 ffffffff 5 800] ------- [5:255 -1:255 27 0] [0x2 4 1 [6 0] [2 4]] (REQUEST)
17  s 0.511309270 _6_ RTR  --- 0 AODV 44 [0 0 0 0] ------- [6:255 2:255 30 5] [0x4 1 [6 4] 10.000000] (REPLY)
18  r 0.511530587 _1_ RTR  --- 0 AODV 48 [0 ffffffff 0 800] ------- [0:255 -1:255 28 0] [0x2 3 1 [6 0] [2 4]] (REQUEST)
19  r 0.514243670 _5_ RTR  --- 0 AODV 44 [9c 5 6 800] ------- [6:255 2:255 30 5] [0x4 1 [6 4] 10.000000] (REPLY)
```

```
20  f 0.514243670 _5_ RTR  —— 0 AODV 44 [9c 5 6 800] ———— [6:255 2:255 29 4] [0x4 2 [6 4] 10.000000] (REPLY)
21  r 0.517480070 _4_ RTR  —— 0 AODV 44 [9c 4 5 800] ———— [6:255 2:255 29 4] [0x4 2 [6 4] 10.000000] (REPLY)
22  f 0.517480070 _4_ RTR  —— 0 AODV 44 [9c 4 5 800] ———— [6:255 2:255 28 3] [0x4 3 [6 4] 10.000000] (REPLY)
23  r 0.520824470 _3_ RTR  —— 0 AODV 44 [9c 3 4 800] ———— [6:255 2:255 28 3] [0x4 3 [6 4] 10.000000] (REPLY)
24  f 0.520824470 _3_ RTR  —— 0 AODV 44 [9c 3 4 800] ———— [6:255 2:255 27 2] [0x4 4 [6 4] 10.000000] (REPLY)
25  r 0.523862870 _2_ RTR  —— 0 AODV 44 [9c 2 3 800] ———— [6:255 2:255 27 2] [0x4 4 [6 4] 10.000000] (REPLY)
26  s 0.523862870 _2_ RTR  —— 0 tcp 40 [0 0 0 0] ———— [2:0 6:0 30 3] [0 0] 0 0
27  r 0.525141470 _3_ RTR  —— 0 tcp 40 [9c 3 2 800] ———— [2:0 6:0 30 3] [0 0] 1 0
28  f 0.525141470 _3_ RTR  —— 0 tcp 40 [9c 3 2 800] ———— [2:0 6:0 29 4] [0 0] 1 0
29  r 0.526357070 _4_ RTR  —— 0 tcp 40 [9c 4 3 800] ———— [2:0 6:0 29 4] [0 0] 2 0
30  f 0.526357070 _4_ RTR  —— 0 tcp 40 [9c 4 3 800] ———— [2:0 6:0 28 5] [0 0] 2 0
31  r 0.527626670 _5_ RTR  —— 0 tcp 40 [9c 5 4 800] ———— [2:0 6:0 28 5] [0 0] 3 0
32  f 0.527626670 _5_ RTR  —— 0 tcp 40 [9c 5 4 800] ———— [2:0 6:0 27 6] [0 0] 3 0
33  r 0.529058270 _6_ RTR  —— 1 ack 40 [0 0 0 0] ———— [6:0 2:0 32 0] [0 0] 0 0
34  s 0.529058270 _6_ RTR  —— 1 ack 40 [0 0 0 0] ———— [6:0 2:0 30 5] [0 0] 0 0
```

6 结论与展望

6.1 结论

本课程考查设计对无限网络中的 MAC 层进行了 NS2 仿真，并分析在不同 MAC 协议，即 802.11a/b/g 下网络的吞吐量，时延，丢包率等性能指标。经过实验数据分析和整合可以得到以下结论：802.11b 和 802.11g 都工作在 2.5GHz 的频段上，而 802.11a 工作在 5GHz 上，考虑无线电波的衰减，抗干扰性 802.11b 和 802.11g 要优于 802.11a，所以通信距离也会更远。802.11a 和 802.11g 都是用了 OFDM 技术，极大的扩大了频谱利用率还有系统容量，具有更高的数据吞吐量，而 802.11b 采用了直接序列扩频 DSSS，性能无法无前两者相比。802.11b 的速率是三者之中最慢的 [17]，然后是 802.11a 和 802.11g。信道方面，802.11b 在 2.5GHz 上有 3 条 11MHz 带宽的信道，802.11a 在 5GHz 上有 8 条 54MHz 带宽的信道，802.11g 在 5GHz 上有 3 条 54MHz 带宽的信道，这使得 802.11a 有更大的系统容量，可以有更大的数据吞吐量 [18]。

同时对 AODV 协议进行了仿真，通过 trace 文件，结合其原理，分析其工作过程。得到以下结论：AODV 基于传统的距离向量路由机制，算法清晰简单，易于编程实现。发送数据的时候，按需寻找路由，AODV 协议会帮助其一步步进行路由发现，直到找到目标路由。使用目标序列号防止循环发生，解决了无穷计数的问题。并且，AODV 要求周期性的广播报文，以更新路由信息，这会消耗一定的能量和网络带宽。

本实验的创新点是使用 *Matlab* 编写脚本文件，直接解析.tr 文件，分析相关数据，并且可以直接绘图，直观简洁。

6.2 展望

本实验还可以进一步测试网络节点移动的情况下网络的性能，这是非常有趣的，值得尝试。另外，因为仿真的时候是节点之间是没有障碍的，这在显示生活中太理想了，是不存在的，那么是否可以增加一些障碍物来测设一下网络的性能呢，值得尝试。

本课程考查一路下来收获颇丰，从 ns2 的安装还有在 Linux 下仿真，不仅学习了 ns2 仿真无线网络的知识，而且还巩固加深了基本 Linux 操作的指令与相关的方法。更重要的是，在这个过程中，加深了对无线网络原理的认识，清楚的意识到学习和动手的重要性。在仿真验证分析数据的过程中，虽然一开始无从下手，但是最后经过耐心的思考还有查找大量网络资料，验证了相关数据的准确性之后，编写了脚本程序进行数据分析。在实验过程中，我们还学会了筛选网上的资料，网上资料纷繁复杂，正确与否不得而知，必须经过标准的验证，才算是准确的数据，这一点非常重要，不能乱用网上的代码，必须要独立思考，解决问题，即使最后的结论是错的，也不要灰心，认真分析原因，踏踏实实改正。

参考文献

[1] 金光, 江先亮. 无线网络技术教程原理、应用与仿真实验 [M]. 清华大学出版社, 2011.

[2] 802.11 协议帧基本格式详解 https://blog.csdn.net/sinat_22991367/article/details/73005140

[3] 802.11 概述及帧结构分析https://blog.csdn.net/ruanjianruanjianruan/article/details/50411057

[4] 物聯網中 AODV 的探索-國立嘉義大學https://www.youtube.com/watch?v=TdHDpGoygDg

[5] NS2 相关学习——在 ns 中模拟无线场景 https://blog.csdn.net/ljm1995/article/details/65939314

[6] ns2 无线局域网基本仿真测量https://blog.csdn.net/kgn28/article/details/5330694

[7] 无线传播模型-FreeSpace、TwoRayGround、Shwdowing 模型 https://tieba.baidu.com/p/1868382145?red_tag=1482954996&traceid=

[8] Tail-Drop_ 百度百科https://baike.baidu.com/item/Tail-Drop/3155653?fr=aladdin

[9] Omnidirectional antenna - Wikipediahttps://en.wikipedia.org/wiki/Omnidirectional_antenna

[10] NS2 LL-layer https://www.isi.edu/nsnam/ns/doc/node155.html

[11] 802.11 协议帧间间隔-SIFS,DIFS,PIFS,EIFS 及 slottime https://blog.csdn.net/rs_network/article/details/17954697

[12] MAC 中的竞争窗口 https://blog.csdn.net/chenkai619/article/details/9389881

[13] 无线通信中的信号传输范围 https://blog.csdn.net/xuewuzhijin2012/article/details/53406691

[14] 無線通信協定 page.47, 89, 96, 108, 120http://www.cs.nchu.edu.tw/~hwtseng/Wireless%20Networks/2CSMA_CA.pdf

[15] NS2 中.tr 分析 http://hanbo31.blog.163.com/blog/static/122821968201111269291024O/

[16] 802.11a-5G-5.8G(5.8GHz)-802.11g-2.4G(2.4GHz) 基本知识 https://wenku.baidu.com/view/93f63a5dc281e53a5802ffed.html

[17] 无线网卡 802.11a b g n 详解及区别https://wenku.baidu.com/view/d576392abd64783e09122bd2

[18] 全国计算机等级考试-网络技术三级教程高等教育出版社 [M], 2018.

7 附录

AODV100.tcl

```
1
2 proc getopt {argc argv} {
3 puts "params: DataRate "
4 global opt
5 for {set i 0} {$i < $argc} {incr i}
6 {
7         set opt($i) [lindex $argv $i]
8 }
9 }
10 getopt $argc $argv ;#读取命令行参数
11
12 set MAC_TYPE $opt(0)
13
14 # Define options
15 set val(chan)           Channel/WirelessChannel    ;# channel type
16 set val(prop)           Propagation/TwoRayGround   ;# radio-propagation model
17 set val(netif)          Phy/WirelessPhy            ;# network interface type
18 set val(mac)            Mac/802_11                 ;# MAC type
19 set val(ifq)            Queue/DropTail/PriQueue    ;# interface queue type
20 set val(ll)             LL                         ;# link layer type
```

```
21 set val(ant)             Antenna/OmniAntenna          ;# antenna model
22 set val(ifqlen)          50                           ;# max packet in ifq
23 set val(nn)              100                           ;# number of mobilenodes
24 set val(rp)              AODV                         ;# routing protocol
25 set val(x)               300                          ;# X dimension of topography
26 set val(y)               200                          ;# Y dimension of topography
27 set val(stop)            5                           ;# time of simulation end
28
29 if {$MAC_TYPE == {a} } {
30
31 Phy/WirelessPhy set freq_ 5.0e+9
32 Phy/WirelessPhy set RXThresh_ 3.17291e-09 ;
33 Phy/WirelessPhy set CSThresh_ 6.5556e-10 ;
34 Phy/WirelessPhy set Pt_ 0.281838
35
36 Mac/802_11 set CWMin 15 ;#竞争窗口大小
37 Mac/802_11 set CWMax 1023
38 Mac/802_11 set SlotTime_ 0.000050
39 Mac/802_11 set SIFS_ 0.000016
40 Mac/802_11 set PreambleLength 96 ;
41 Mac/802_11 set PLCPHeaderLength_ 24 ;
42 Mac/802_11 set PLCPDataRate_ 6.0e6
43 Mac/802_11 set dataRate 54e6
44 Mac/802_11 set basicRate_ 6.0e6
45 }
46 elseif {$MAC_TYPE == {b}} {
47
48 Phy/WirelessPhy set freq_ 2.4e+9
49 Phy/WirelessPhy set RXThresh_ 2.78869e-09  ;
50 Phy/WirelessPhy set CSThresh_ 5.76175e-09  ;
51 Phy/WirelessPhy set Pt_ 0.281838
52
53 Mac/802_11 set CWMin 31
54 Mac/802_11 set CWMax 1023
55 Mac/802_11 set SlotTime_ 0.000020
56 Mac/802_11 set SIFS_ 0.000010
57 Mac/802_11 set PreambleLength 144
58 Mac/802_11 set PLCPHeaderLength_ 48
59 Mac/802_11 set PLCPDataRate_ 1.0e6
60 Mac/802_11 set dataRate 11e6
61 Mac/802_11 set basicRate_ 1.0e6
62 } elseif {$MAC_TYPE == {g} } {
63
64 Phy/WirelessPhy set freq_ 2.4e+9
65 Phy/WirelessPhy set RXThresh_ 2.78869e-09
66 Phy/WirelessPhy set CSThresh_ 5.76175e-09
67 Phy/WirelessPhy set Pt_ 0.281838
68
69 Mac/802_11 set CWMin 15
70 Mac/802_11 set CWMax 1023
71 Mac/802_11 set SlotTime_ 0.000009
72 Mac/802_11 set SIFS_ 0.000016
73 Mac/802_11 set PreambleLength 120 ;
74 Mac/802_11 set PLCPHeaderLength_ 24 ;
75 Mac/802_11 set PLCPDataRate_ 6.0e6
76 Mac/802_11 set dataRate 54e6
77 Mac/802_11 set basicRate_ 6.0e6
78 }
79 else {
```

```
puts "␣error␣MAC_TYPE"
}

set ns           [new Simulator]
set tracefd        [open AODV100.tr w]
set namtrace       [open AODV100.nam w]

$ns trace-all $tracefd
$ns namtrace-all-wireless $namtrace $val(x) $val(y)

# set up topography object
set topo        [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#
#  Create nn mobilenodes [$val(nn)] and attach them to the channel.
#

# configure the nodes
$ns node-config -adhocRouting $val(rp) \
-llType $val(ll) \
-macType $val(mac) \
-ifqType $val(ifq) \
-ifqLen $val(ifqlen) \
-antType $val(ant) \
-propType $val(prop) \
-phyType $val(netif) \
-channelType $val(chan) \
-topoInstance $topo \
-agentTrace ON \
-routerTrace ON \
-macTrace OFF \
-movementTrace OFF

for {set i 0} {$i < $val(nn) } { incr i } {
set node_($i) [$ns node]
$node_($i) set X_ [ expr 30*($i%10) ]
$node_($i) set Y_ [ expr 30*($i/10) ]
$node_($i) set Z_ 0.0
}


set max_num 30
#建立tcp连接
for {set index 0} {$index < $max_num} {incr index} {

set tcp_($index) [new Agent/TCP/Newreno] ;#新建TCP源代理
$ns attach-agent $node_($index) $tcp_($index) ;#将TCP源代理与源节点关联

set sinkNum [expr $index + $val(nn)/2]
set sink_($sinkNum) [new Agent/TCPSink]                    ;#新建TCP宿代理
$ns attach-agent $node_($sinkNum) $sink_($sinkNum) ;#将TCP宿代理与目的节点关联

$ns connect $tcp_($index) $sink_($sinkNum)  ;#连接TCP代理与TCP槽代理

$tcp_($index) set packetSize_ 1500  ;#设置TCP代理的最大包数为1500
}

```

```
#建立FTP连接
set FTP_START_TIME 0.1
set FTP_END_TIME [expr ($val(stop) - 1)]

for {set index 0} {$index < $max_num} {incr index} {

set ftp_($index) [new Application/FTP] ;#设置FTP代理
$ftp_($index) attach-agent $tcp_($index) ;#在FTP和TCP代理建立连接

$ns at $FTP_START_TIME "$ftp_($index) start"
$ns at $FTP_END_TIME "$ftp_($index) stop"

set FTP_START_TIME [expr $FTP_START_TIME + 0.1]
}


# Define node initial position in nam
for {set i 0} {$i < $val(nn)} { incr i } {
$ns initial_node_pos $node_($i) 10
}

# Telling nodes when the simulation ends
for {set i 0} {$i < $val(nn) } { incr i } {
$ns at $val(stop) "$node_($i) reset";
}

# ending nam and the simulation
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "stop"
$ns at $val(stop) "puts \"end simulation\" ; $ns halt"
proc stop {} {
global ns tracefd namtrace
$ns flush-trace
close $tracefd
close $namtrace
}
$ns run
```

Matlab 计算吞吐量代码

connect.m

```
function [ThroughputSerial, TimePointSerial] = connect(pack_time_num, pack_size_num)
ThroughputSerial = [0];
TimePointSerial = [0];
pack_time_num = floor(pack_time_num);
serial = [pack_time_num' pack_size_num'];
%统计每一秒的数据吞吐量
for i = 1:max(pack_time_num)+1
        class = serial(serial(:,1)==(i-1), :);
        Throughput = sum(class(:, 2));
        TimePoint = class(1, 1) + 1;%第TimePoint秒
        ThroughputSerial = [ThroughputSerial Throughput];
        TimePointSerial = [TimePointSerial TimePoint];
end
% disp(ThroughputSerial);
```

Extract.m

```
function [pack_size_num, pack_time_num] = Extract(fileName)
fid=fopen(fileName, 'r');

line_iter = 1;
while 1
        line = fgetl(fid);
        if line == -1 %若是读取到末尾，退出该循环
                break;
        end
        %读取包大小
        for i = 1:length(line)
                if line(i)=='['
                        pack_size_str = [];
                        k = 1;
                        space_num = 0;
                        while 1
                                if line(i-k) ~= '␣'
                                        pack_size_str = [line(i-k) pack_size_str];
                                end
                                if line(i-k) =='␣'
                                        space_num = space_num + 1;
                                end
                                if space_num ==2
                                        pack_size_num(line_iter) = str2num(pack_size_str);
                                        %disp(pack_size_num(line_iter));
                                        break;
                                end
                                k = k + 1;
                        end
                break;
                end
        end

        %读取时间
        for i = 1:length(line)
                if line(i)=='_'
                        pack_time_str = [];
                        k = 1;
                        space_num = 0;
                        while 1
                                if line(i-k) ~= '␣'
                                        pack_time_str = [line(i-k) pack_time_str];
                                end
                                if line(i-k) =='␣'
                                        space_num = space_num + 1;
                                end
                                if space_num ==2
                                        pack_time_num(line_iter) = str2num(pack_time_str);
                                        %disp(pack_time_num(line_iter));
                                        break;
                                end
                                k = k + 1;
                        end
                        break;
                end
        end
        line_iter = line_iter + 1;
end
fclose(fid);
```

analysis.m

```
clc;close all;clear;
[pack_size_num_a, pack_time_num_a] = Extract('newtra/AODV100a.tr');
[pack_size_num_b, pack_time_num_b] = Extract('newtra/AODV100b.tr');
[pack_size_num_g, pack_time_num_g] = Extract('newtra/AODV100g.tr');

[ThroughputSerial_a, TimePointSerial_a] = connect(pack_time_num_a, pack_size_num_a);
[ThroughputSerial_b, TimePointSerial_b] = connect(pack_time_num_b, pack_size_num_b);
[ThroughputSerial_g, TimePointSerial_g] = connect(pack_time_num_g, pack_size_num_g);


figure;
h_a = plot(gca,TimePointSerial_a, ThroughputSerial_a,...
        's-r','MarkerSize', 8,...
        'MarkerFaceColor', 'r','MarkerEdgeColor', 'r');
hold on;

h_b = plot(gca,TimePointSerial_b, ThroughputSerial_b,...
        'o-b','MarkerSize', 8,...
        'MarkerFaceColor', 'b','MarkerEdgeColor', 'b');

h_g = plot(gca,TimePointSerial_g, ThroughputSerial_g,...
        '^-g','MarkerSize', 8,...
        'MarkerFaceColor', 'g','MarkerEdgeColor', 'g');
legend('802.11a', '802.11b', '802.11g');

ylabel('Throughput/Byte');xlabel('time/s');
title('Throughput Plot');xlim([-1 7]);
disp('over');
```

绘制不同协议，不同节点距离下的延时和丢包率折线图

dropAndDelay.m

```
clc;clear;close all;
x =[20 30 40 50 60];
a_drop = [4.858 3.70586 0 0 0];
a_delay = [752.007 519.336 0 0 0];

b_drop = [5.17711 3.73134 3.27645 2.94711 3.18218];
b_delay = [371.43 398.617 421.546 387.389 446.676];

g_drop = [7.5543 5.46529 4.19017 3.2974 3.08904];
g_delay = [224.633 342.678 273.245 271.466 305.223];

%时 延
figure;
h_a = plot(gca,x, a_delay,...
        's-r','MarkerSize', 8,...
        'MarkerFaceColor', 'r','MarkerEdgeColor', 'r');
hold on;

h_b = plot(gca,x, b_delay,...
        'o-b','MarkerSize', 8,...
        'MarkerFaceColor', 'b','MarkerEdgeColor', 'b');

h_g = plot(gca,x, g_delay,...
```

```
        '^-g','MarkerSize', 8,...
        'MarkerFaceColor', 'g','MarkerEdgeColor', 'g');
legend('802.11a', '802.11b', '802.11g','location','best');

ylabel('delay/ms');xlabel('distance/m');
title('Delay Plot');xlim([10 70]);
%丢包率

figure;
%,'MarkerFaceColor', 'r','MarkerEdgeColor', 'r'
h_aa = plot(gca,x, a_drop,...
        's-r','MarkerSize', 8,...
        'MarkerFaceColor', 'r','MarkerEdgeColor', 'r');
hold on;

h_bb = plot(gca,x, b_drop,...
        'o-b','MarkerSize', 8,...
        'MarkerFaceColor', 'b','MarkerEdgeColor', 'b');

h_gg = plot(gca,x, g_drop,...
        '^-g','MarkerSize', 8,...
        'MarkerFaceColor', 'g','MarkerEdgeColor', 'g');
legend('802.11a', '802.11b', '802.11g','location','best');

ylabel('drop Rate');xlabel('distance/m');xlim([10 70]);
title('Drop Ratio Plot');
```

FTPnode.m

```
clc;clear;close all;
x = [10 20 30 40 50];
a_drop = [1.84397 1.50215 1.79974 2.39219 2.77411];
a_delay = [624.368 615.401 747.132 621.422 409.255];
b_drop = [2.92165 2.95476 3.13636 2.99116 2.93276];
b_delay = [419.644 407.44 591.637 561.15 329.134];
g_drop = [2.04873 2.65957 2.67272 3.51258 3.40349];
g_delay = [291.145 341.008 327.76 379.04 244.125];
%时延
figure;
h_a = plot(gca,x, a_delay,...
        's-r','MarkerSize', 8,...
        'MarkerFaceColor', 'r','MarkerEdgeColor', 'r');
hold on;

h_b = plot(gca,x, b_delay,...
        'o-b','MarkerSize', 8,...
        'MarkerFaceColor', 'b','MarkerEdgeColor', 'b');

h_g = plot(gca,x, g_delay,...
        '^-g','MarkerSize', 8,...
        'MarkerFaceColor', 'g','MarkerEdgeColor', 'g');
legend('802.11a', '802.11b', '802.11g','location','best');

ylabel('delay/ms');xlabel('FTP Agent/par');
title('Delay Plot');xlim([0 60]);
%丢包率

figure;
h_aa = plot(gca,x, a_drop,...
```

```
        's␣-r','MarkerSize', 8,...
        'MarkerFaceColor', 'r','MarkerEdgeColor', 'r');
hold on;

h_bb = plot(gca,x, b_drop,...
        'o-b','MarkerSize', 8,...
        'MarkerFaceColor', 'b','MarkerEdgeColor', 'b');

h_gg = plot(gca,x, g_drop,...
        '^-g','MarkerSize', 8,...
        'MarkerFaceColor', 'g','MarkerEdgeColor', 'g');
legend('802.11a', '802.11b', '802.11g','location','best');

ylabel('drop␣Rate');xlabel('FTP␣Agent/par');xlim([0 60]);
title('Drop␣ratio␣Plot');
```

延时与丢包率计算

delay.awk

```
1
2 BEGIN {
3         seqno = -1;
4         count = 0;
5 }
6
7 {
8         if($4 == "AGT" && $1 == "s" && seqno < $6) {
9                 seqno = $6;
10                }
11                else if(($4 == "RTR") && ($1 == "r")&&($10 != "ffffffff")) {
12                receivedPackets++;
13                } else if ($4 == "RTR" && $1 == "D" && $10 != "ffffffff"){
14                droppedPackets++;
15                }
16
17                #end-to-end delay
18
19                if($4 == "AGT" && $1 == "s") {
20
21                start_time[$6] = $2;
22
23                } else if(($7 == "tcp") && ($1 == "r")) {
24
25                end_time[$6] = $2;
26                } else if($1 == "D" && $7 == "tcp") {
27                end_time[$6] = -1;
28                }
29
30        }
31
32        END {
33
34                for(i=0; i<=seqno; i++) {
35
36                if(end_time[i] > 0) {
37
38                delay[i] = end_time[i] - start_time[i];
39
40                count++;
```

```
41                  }
42
43                  else
44
45                  {
46
47                          delay[i] = -1;
48
49                  }
50          }
51
52          for(i=0; i<=seqno; i++) {
53
54                  if(delay[i] > 0) {
55
56                  n_to_n_delay = n_to_n_delay + delay[i];
57
58                  }
59
60          }
61
62          n_to_n_delay = n_to_n_delay/count;
63          print "\n";
64          print "丢包率␣␣␣␣␣␣␣␣␣␣␣␣␣: ␣" droppedPackets/(receivedPackets+droppedPackets)*100 "␣%
                ";
65          print "平均时延␣␣␣␣␣␣␣␣␣␣␣␣: ␣" n_to_n_delay * 1000 "␣ms";
66          print "\n";
67  }
```