NS-3 802.11 物理层源代码实现原理分析

王 悦
(中央财经大学信息学院信息管理系 北京 100081)

摘 要 NS-3 是国外近几年发展起来的重要网络仿真软件,它提供了比 NS-2 更低层次的无线功能抽象,更贴近真实的无线物理层的工作原理。分析了 802.11 无线物理层的源代码,对其主要实现机制进行了详细剖析,包括节点状态与接收分组的条件、信道繁忙起止时间的计算、信道支持多个路径损耗衰落模型、误码率与分组接收成功率的计算、多个干扰分组的跟踪管理和以分块为单位的累积干扰计算,并对协议修改给出了建议。该工作为理解 NS-3 无线仿真原理做出了有益贡献。

关键词 NS-3,802.11,无线物理层,源代码分析
中图法分类号 TP391 文献标识码 A

Analyzing Source Code of 802.11 Physical Layer Implementation in NS-3

WANG Yue
(Department of Information Management, School of Information, Central University of Finance and Economics, Beijing 100081, China)

Abstract NS-3 is an important network simulator, offering a lower level abstraction of wireless functionality than NS-2 and is more close to realistic wireless physical layer. In this work, we read the source code and analyzed simulation mechanisms of 802.11 physical layer, including node states and the condition that packets can be received, the computation of the starting and ending time of channel busy state, packet reception power considering the addition effect of multiple path loss and fading models, the computation of bit error rate and packet error rate, and tracking of multiple interfering packets and the computation of interference based on chunk units. We gave some advices for protocol modification. This paper makes a contribution to understand the wireless simulation principle of NS-3.

Keywords NS-3, 802.11, Physical layer, Source code analysis

1 引言

NS-3[1] 是新型开源的离散事件网络模拟器,它对 NS-2 的网络模块做了重新改写,不支持 NS-2 的 API。相比 NS-2,NS-3 提供了对网络协议的更低抽象层次,使得它更贴近真实系统的实现。NS-2 中发现的一些局限(如不支持多类型接口的节点、无线物理层模型过于简化)被修正了。在无线物理层仿真方面,NS-2 的物理层模型主要存在如下问题。

(1)分组接收成功模型过于简化:在 NS-2 中,当 SNR 大于 SNR 阈值时分组接收成功,否则接收失败。而在实际中接收取决于 SNR、发送模式(调制模式、比特速率)的概率值。

(2)NS-2 没有考虑累积干扰强度,为了简化计算,只是在一个分组接收期间对每一个干扰源的分组单独计算其 SNR 值。在实际中,如果多个干扰源的分组同时到达接收节点,它们的接收功率将会累加起来对正在接收的分组产生干扰。

(3)NS-2 只有 Friis 和 Two-ray ground 两种大尺度衰落的信道模型,没有包含多径衰落、阴影效应等信道模型[8]。

NS-3 的 802.11 模块始自 2006 的一篇论文[2],基本解决了上述问题:(1)提供了多种发送模式,真实地模拟了分组成功接收概率[3,4];(2)精细地跟踪计算累积干扰强度随时间变化的情况;(3)提供大量的大小尺度衰落的信道模型。

由于 NS-3 的作者没有对外提供详细的设计文档,我们对上述 3 个问题的解决方案并不明确。而在科研工作中,往往需要扩展一个网络仿真模块,如实现新型无线物理层架构[5,9]或新型干扰消减技术[6],这些需要详细理解 NS-3 无线物理层的数据结构和处理流程,而这方面的技术资料在国内外非常欠缺。继 NS-2 802.11 模块的源代码分析[10]之后,本文通过分析源代码和跟踪调试给出了 NS-3 无线物理层的实现原理,对进一步理解 NS-3 做出了有益的贡献。

2 源代码分析方法

我们在 source insight 软件里查看 NS-3 的源代码。该软件可以快速定位到类、变量和函数(方法)的声明以及函数(或方法)之间的调用关系。NS-3 源代码的结构错综复杂。在源代码分析中,主要采用“场景代入法”,这个想法来自于“走查”(walk through)的测试方法[7],即将一个场景代入手工模拟

本文受国家自然科学基金项目(61309030,61100112),教育部人文社会科学研究一般项目(13YJA630089),教育部人文社会科学研究规划基金项目(15YJAZH066),北京市哲学社会科学规划项目(13JGA004),北京市社科联青年社科人才资助项目(2015SKL009),中财 121 人才工程青年博士发展基金(QBJ1427)资助。

王 悦(1979—),男,副教授,主要研究方向为计算机网络、机器学习,E-mail:yueLwang@163.com。

代码的运行过程。比如,当多干扰源的分组依次到来时,InterferenceHelper::AppendEvent()方法将它们记录在m_niChanges列表中。分析这段代码时,手工产生多个干扰分组,让它们依次调用AppendEvent()方法,从而对m_niChanges列表的更新操作做出清晰解读。实践中发现,"场景代入法"可以准确地分析出大部分源代码的功能,能产生逻辑上自洽、符合网络原理的分析结果。但是由于NS-3的继承关系复杂,对不确定的部分,将用GDB调试工具来跟踪其执行输出。如下代码显示了调试scratch目录下的friis-test.cc脚本的过程:

```
./waf --run scratch/friist-test
./waf --command-template="gdb %s" --run build/sratch/friis-test
```

此例中所有目录都在ns-3.19下。首先,waf命令编译脚本friist-test.cc,编译后的可执行文件friis-test放入目录build/scratch下。然后,以waf的调试模式运行friis-test,在打开的gdb模式中可以对NS-3的源代码设置断点、单步执行和打印变量的值。

3 WiFi物理层实现分析

图1显示了类之间的依赖关系,箭头旁的文字表示成员变量。

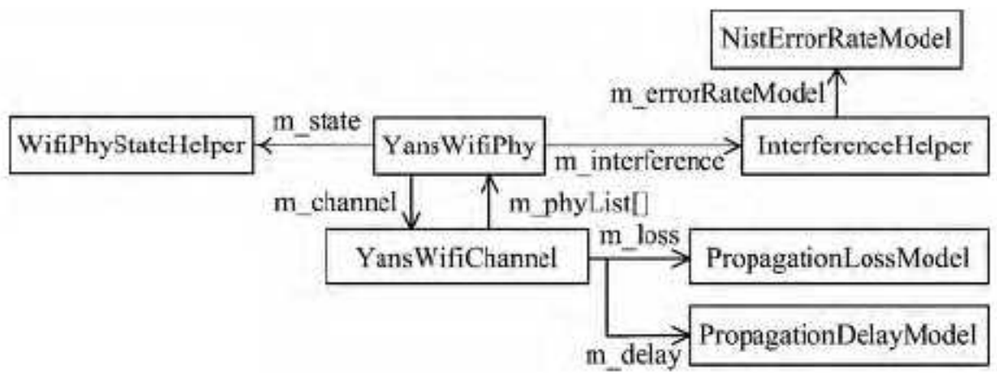


图1 NS-3无线物理层主要类之间的依赖关系

在此给出无线物理层的概要实现框架。在图1所示的类图中,YansWifiPhy处于核心位置,支持802.11的a、b、g、n标准,它联接其它类的对象完成无线物理层的接收和发送过程。从一个分组的发送说起,该节点的YansWifiPhy对象(模拟了无线网卡的物理层)通过调用成员变量m_channel(YansWifiChannel类)的send()方法,将分组的处理流程传递给该节点所在的YansWifiChannel对象。后者模拟了无线信道,它通过成员变量列表m_phyList[]访问到众多YansWifiPhy对象,它们是连接到该信道上的各个无线节点的物理层。在YansWifiChannel的send()方法里,通过调用其成员m_loss(PropagationLossModel类)的CalcRxPower()方法计算出到第i个接收节点(m_phyList[i])的接收功率,通过调用其成员m_delay(PropagationDelayModel类)的GetDelay()方法计算出到此接收节点的传播延迟,这些操作模拟了分组在信道中的传播机制。在经过传播延迟后,该分组被YansWifiChannel的Receive()方法处理,在此方法里进一步调用了m_phyList[i]->StartReceivePacket()方法将分组传递给第i个接收节点的无线物理层。由于无线信道的广播性质,一个接收节点的物理层在"听到"上一个分组到达期间,下一个分组可能又到达了。到底接收(或称作"捕获")哪个分组由接收条件给出,即,当YansWifiPhy对象没有处于接收或发送某一分组的状态,且当前到达的分组的接收功率大于接收阈值时,YansWifiPhy接收此分组。YansWifiPhy的状态通过调用成员变量m_state(WifiPhyStateHelper对象)的GetState()方法来获取。以上分组接收准备的处理流程在StartReceivePacket()方法里给出。同时,YansWifiPhy通过调用其成员变量m_interference(InterferenceHelper类)的Add()方法来维护累积干扰随时间的变化。InterferenceHelper的成员变量m_niChanges是一个C++ vector模板,它记录了每个干扰分组到达和离开时间及其对累积干扰功率的增减。在一个分组接收结束时,通过计算在分组接收期间的各个不同累积干扰值下的误码率,进而计算出该分组的成功接收概率,并按此概率随机选择该分组是接收成功抑或接收失败。各种调制模式的误码率由成员变量m_errorRateModel(NistErrorRateModel类)的相应方法给出。以上分组接收结束的处理流程在YansWifiPhy的EndReceive()方法给出。

此外,需要说明的是,无论在NS-2还是NS-3中,分组发送和接收并不是真实的通信过程,而只是包含分组信息的Packet类的对象通过方法之间的调用从一个处理对象传递到另一个处理对象。NS的仿真时间也不是真实流逝的物理时间,而只是表示事件发生时间的数值变量。NS按照事件发生的先后顺序将其插入到事件队列中(调用Simulator::ScheduleWithContext()方法)。NS的事件调度模块循环地从事件队列中取得下一个事件(比如,分组到达事件)交由相应的网络模块来处理。NS中的当前时刻Now指的是当前正在被程序处理事件的发生时间。NS本质来说就是一个事件调度器。这些概念对于正确理解NS-3很有必要。

3.1 节点状态

节点状态用于跟踪记录节点的收发时序,在何时启动分组接收和记录信道繁忙时段等方面有重要作用。节点在一个时刻只能处于以下5种状态之一。TX:节点正在发送一个分组;RX:节点正在接收一个分组;CCA-BUSY:节点监听到信道繁忙;IDLE:节点监听到信道空闲;SWITCHING:节点正在多信道之间切换,对单信道无需考虑。

节点状态由YansWifiPhy对象的m_state成员变量(WifiPhyStateHelper对象)的GetState()方法获取。状态X的起始与结束时间由成员变量m_start$X$和m_end$X$维护。GetState()的实现方法为:若当前仿真时间位于某状态的起止时间区间,则节点属于该状态。

3.2 WiFi物理层发送过程

此过程由节点的YansWifiPhy对象发起和掌管,调用SendPacket()方法,通过成员变量m_channel(YansWifiChannel对象)的send()方法,将分组的处理流程传递到节点的YansWifiChannel对象,即仿真了将分组发送到信道的过程。

3.3 信道传输过程

此过程由发送节点的YansWifiChannel对象掌管,实现了信道广播机制。成员变量m_phyList[]列表记录着连接到该信道上每个接收节点的YansWifiPhy对象。通过返回m_phyList[i]的MobilityModel成员变量,取得第i个接收节点的位置。由发送和接收节点的位置进一步计算出传输延迟和接收功率,调用方法如下:

```
m_delay->GetDelay(senderMobility, receiverMobility);
m_loss->CalcRxPower(txPowerDbm, senderMobility, receiverMobility);
```

NS-3 为多个接收节点构造多个信道传输事件。NS-3 事件调度器将一个信道传输事件(即分组和它的接收功率、信道传播延迟、事件的处理方法 YansWifiChannel::Receive()以及进一步处理该分组的 m_phyList[]元素的下标)插入到事件队列中。当一个信道传播事件被处理时,YansWifiChannel::Receive()方法调用其接收节点的 m_phyList[i].StartReceivePacket()方法,开始接收节点的物理层对分组的处理。代码如下:

```
m_phyList[i]->StartReceivePacket(packet,rxPowerDbm,txVector,preamble);
```

以上就完成了分组经由信道从发送节点传播到接收节点的仿真过程。在上述计算信道传播延迟时调用了 PropagationDelayModel::GetDelay()方法,延迟一般设为收发节点的距离除以光速。计算接收功率时用到 PropagationLossModel::CalcRxPower()方法。NS-3 支持大量的信道损耗和衰落模型,如大尺度路径损耗的 FriisPropagationLossModel,TwoRayGroundPropagationLossModel 等,小尺度的路径衰落模型 NakagamiPropagationLossModel 等,它们是 PropagationLossModel 类的继承类。

值得注意的是,NS-3 允许多个信道损耗和衰落模型作用于一个信道,这更符合真实情况。在实现中,一个 PropagationLossModel 对象可以指向下一个 PropagationLossModel 对象(通过 m_next 指针)。PropagationLossModel::CalcRxPower()方法的代码如下:

```
double self=DoCalcRxPower (txPowerDbm,a,b);
if (m_next != 0) self=m_next->CalcRxPower (self,a,b);
```

这是一个递归调用,首先由发送功率 txPowerDbm、发送节点 a 和接收节点 b 的位置得到当前信道损耗模型的接收功率,存入到 self 变量中。然后,将 self 作为发送功率调用 m_next 指向的下一个 PropagationLossModel 对象的 CalcRxPower()方法得到叠加新的路径损耗后的接收功率。

3.4 WiFi 物理层接收过程

接收过程分两阶段:开始接收和结束接收。首先声明一些关键点。在 NS-3 中提到的"接收"(receive)有两种含义:第一种情况是指分组的比特流确实被接收节点的物理层接收和解码,当节点接收完最后一个比特时,解码结果可以是成功或失败;第二种情况即是"监听"到该分组,尽管分组的电磁波到达接收节点,但是由于某种原因(详见下文(1.2)),接收节点并没有真正地接收它,但是该分组的电磁波在其发送期间成为接收节点的干扰噪声,干扰强度等于分组的接收功率。在一个时刻可以同时存在多个干扰分组,其累积干扰强度为各个干扰分组的接收功率之和。当累积干扰强度和热噪声之和大于载波监听阈值时,信道处于繁忙状态,否则处于空闲状态。我们一般称第二种情况为"监听",但在下文中为符合 NS-3 的习惯,有时也用"接收",请注意根据上下文区分。

(1)开始接收过程。这是当一个分组首个比特到达 YansWifiPhy 对象时的处理流程,由 StartReceivePacket()方法掌管。该方法的代码大致分为 3 个部分。

(1.1)多个干扰分组的跟踪记录。将这个分组作为一个干扰分组记录下来,将在 3.5 中详细说明。

(1.2)判断节点能否接收当前分组。分两种情况,节点状态为 SWITCHING、RX 或 TX 时,不接收这个分组。值得注意的是,当节点在接收上一个分组的期间(节点处于 RX 状态),不再接收新的分组。当节点状态为 IDLE 或 CCA-BUSY 并且分组接收功率大于接收阈值 m_edThresholdW(此变量名表示 energy detection threshold in watts)时,准备接收该分组,这就是分组接收条件。NS-3 的事件调度器将该分组插入到事件队列中,在分组发送时长(rxDuration)后由 YansWifiPhy::EndReceive()方法处理。

```
m_endRxEvent=Simulator::Schedule
(rxDuration,&YansWifiPhy::EndReceive,this,packet,event);
```

(1.3)节点监听到信道繁忙状态的跟踪管理。如果无线物理层不接收该分组,它将作为干扰分组贡献到累积干扰中。当累积干扰功率大于或等于载波监听阈值 m_ccaMode1ThresholdW(此变量名表示 clear channel assessment threshold in watts)时,接收节点监听到信道繁忙,处于 CCA-BUSY 状态。此部分的操作分为以下 3 个步骤。

(a)信道繁忙状态的触发。在分组的第一个比特到来时,节点状态已处于 SWITCHING、TX、RX 三者之一,或者状态为 IDLE 或 CCA-BUSY 并且分组的接收功率小于接收阈值 m_edThresholdW(这些情况下节点不接收当前分组),那么当前分组将作为一个干扰分组,代码 goto 跳转到 maybeCcaBusy 行号,进行下面两步;否则,接收当前分组,忽略下面两步。

(b)信道繁忙状态延迟的计算。调用 m_interference 成员变量(InterferenceHelper 对象)的 GetEnergyDuration()方法计算从当前时刻(分组第一个比特到达节点物理层的时刻)到信道繁忙终止时刻的延迟时长 delayUntilCcaEnd,代码如下:

```
Time delayUntilCcaEnd=m_interference.GetEnergyDuration(m_ecaMode1ThresholdW);
```

具体计算过程为,从当前仿真时刻开始查找累积干扰功率首次小于 m_ecaMode1ThresholdW 的时刻,两者之差即为 delayUntilCcaEnd。累积干扰功率计算见 3.5。

(c)确定信道繁忙状态的起止时刻。调用 m_state 成员变量的 SwitchMaybeToCcaBusy()方法。该方法更新信道繁忙的起始时刻的代码如下:

```
if (GetState () != WifiPhy::CCA_BUSY) m_startCcaBusy=now;
```

更新信道繁忙的结束时刻的代码如下,这里 duration 是 delayUntilCcaEnd:

```
m_endCcaBusy=std::max (m_endCcaBusy,now+duration);
```

(2)结束接收过程。此过程仅适用于节点接收分组的情况(见(1.2))。当分组的最后一个比特到达无线节点的事件被 NS-3 事件调度器处理时,调用 YansWifiPhy 对象的 EndReceive()方法。该方法的主要功能是计算分组在接收期间的错误率 *PER*(Packet Error Rate),并按照 *PER* 的概率来决定是否接收此分组,即判断 m_random->GetValue()>snrPer.per,其中 m_random->GetValue()产生一个[0,1]之间的均匀分布的随机数。

由于分组接收期间可能有许多个干扰分组到达或离开,

累积干扰功率随之发生变化，因此 NS-3 单独计算在累积干扰功率恒定不变的时间区间里的分组的分块(chunk)的成功接收率 PSR_{chunk}，然后将所有分块的 PSR_{chunk} 累乘起来成为整个分组的 PSR(packet success rate)。有如下公式：

$$PSR_{chunk}=(1-BER)^{chunk-nbits} \tag{1}$$

$$PSR=PSR_{chunk\,1}\cdot PSR_{chunk\,2}\cdot\cdots\cdot PSR_{chunk\,m} \tag{2}$$

$$PER=1-PSR \tag{3}$$

式(1)中的 BER 指比特误码率，取决于调制方法和比特速率；$chunk-nbits$ 是分块的比特数，此计算过程调用 NistErrorRateModel 对象的 GetChunkSuccessRate() 方法。以 1Mbps 的 DSSS 调制为例，该方法进一步调用 DsssErrorRateModel::GetDsssDbpskSuccessRate() 方法，代码如下：

```
double EbN0 = sinr * 22000000.0/1000000.0;
double ber = 0.5 * std::exp(-EbN0);
return std::pow((1.0 - ber), static_cast<double>(nbits));
```

前两行代码为 BER 的计算公式(这里 $SINR$ 是该分段的信噪比)：

$$BER_{1M,DSSS}=0.5e^{-22\cdot SINR} \tag{4}$$

第三行代码是 PSR_{chunk} 的计算公式，即式(1)。

式(2)表示一个分组的 PSR 是它的各个分块的 PSR 的乘积。式(3)表示一个分组的接收失败率 PER(packet error rate)等于 $1-PSR$。

3.5 多个干扰分组的跟踪记录与累积干扰的计算

前面看到，无论计算信道繁忙时间还是分块的 SINR 时，都需要知道多个干扰分组在一个分组接收期间的到达时序，从而能划分分块并计算累积干扰功率。此部分功能由 InterferenceHelper 类的对象(即无线节点的 YansWifiPhy 对象的 m_interference 成员变量)掌管。

每一个干扰分组到达接收节点后，为其创建一个 InterferenceHelper::Event 对象(注意这不是 NS 事件队列中的事件)，由一个 Event 对象又派生出两个 NiChange 对象。

在 InterferenceHelper 类里嵌套定义了 Event 类，将其称为干扰事件类。该类的主要成员变量是 m_startTime，m_endTime 和 m_rxPowerW，分别表示了一个干扰分组的第一个比特的到达时间、最后一个比特的到达时间和干扰强度(即接收功率)。

在 InterferenceHelper 类里嵌套定义了 NiChange 类(词类名表示 Noise & Interference Change)，称为干扰变化类。该类的对象有两个成员变量：m_time 表示此干扰变化的发生时间，m_delta 表示此干扰变化对累积干扰功率的增量。当一个干扰分组的首个比特到达接收节点时，干扰发生，创建一个 NiChange 对象，m_time 为该干扰分组第一个比特的到达时间，m_delta 为该干扰分组的接收功率；当该干扰分组的最后一个比特到达接收节点时，干扰结束，m_time 为最后一个比特到达时间，m_delta 为负的分组的接收功率，表示将从累积干扰里减去该分组的干扰功率。

InterferenceHelper 对象里定义了成员变量 m_niChanges，它是 NiChange 对象的列表(C++ Vector)，维护着当前仿真时刻以后的 NiChange 对象。

以上这些数据结构对多个干扰分组进行了有效的跟踪记录。此部分因代码量较大，只简述其主要功能。

(1)多个干扰分组的跟踪记录。当节点的 YansWifiPhy 对象在 StartReceivePacket() 方法里处理分组第一个比特到达的事件时，间接调用 m_interference 成员变量的 AppendEvent() 方法，将所创建的两个 NiChange 对象(对应干扰开始和结束的干扰变化)插入到 NiChanges 列表。当前仿真时刻的累积干扰记录在 m_firstPower 成员变量里(m_firstPower 记录着一个接收分组的第一个比特到来时的累积干扰功率)，同时移除当前时刻以前的 NiChange 对象。

(2)累积干扰与 PSR_{chunk} 的计算。当节点的 YansWifiPhy 对象在 EndReceive() 方法里处理分组最后一个比特到达的事件时(表明接收完毕该分组)，间接调用 m_interference 成员变量的 CalculatePer() 方法。这个方法在 m_niChanges 列表中取得在该分组接收期间的 NiChange 对象，并将其存入 ni 列表；然后，遍历 ni 列表。在相邻两个 NiChange 对象发生时刻(记为 previous 和 current)之间，累积干扰功率不变。当前累加干扰功率等于 m_firstPower 反复累加当前时刻以前的 NiChange 对象的 m_delta 成员变量。在已知分组接收功率和累积干扰功率后，调用 CalculateChunkSuccessRate() 方法计算该分块(由 previous 和 current 划定)的接收成功率 PSR_{chunk}。由于 previous 和 current 可能位于不同报文首部的发送期间内，而不同报文首部所使用的调制方式不同，因此需要调用不同的误码率计算方法。

4 验证实验

用如下实验来验证 NS-3 的主要物理层功能模块，包括累积干扰、路径损耗模型、载波监听。图 2 是实验拓扑图。发送节点 T 持续地往接收节点 R 发送 UDP 广播分组，两者距离为 50m。干扰节点同时发送分组，它与 R 的距离为 200m。MAC 协议设为 802.11b，发送模式为 DsssRate11Mbps。干扰分两种测试情况：单干扰源，只 I_1 发送；双干扰源，I_1、I_2 同时发送。信道传输模型分两种测试情况：Frris 模型、Frris+Rayleigh 模型，后者在 Friis 的大尺度路径损耗下再叠加 Rayleigh 衰落，使得每个接收或干扰分组的接收功率是一个随机值。发送节点 T 的载波监听分两种测试：关闭载波监听、打开载波监听(载波监听阈值设为 −85dBm)，前者通过将载波监听阈值设为一个极大值来实现。干扰节点始终关闭载波监听，以产生持续干扰。T 发送 1000 个包，在 R 处统计成功接收 T 的分组数，计算出分组成功接收率(PSR)。

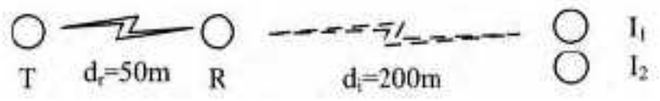


图 2 仿真实验拓扑

如表 1 所列，当关闭载波监听时，双干扰源比单干扰源的 PSR 低，因为累积干扰导致更高的误码率。当其它条件不变时，在发送节点 T 处打开载波监听比关闭载波监听获得显著提升的 PSR，因为载波监听使得 T 有效地避开干扰节点发送区间，避免了碰撞。当其它条件不变时，在信道传输中增加 Rayleigh 衰落导致更低的 PSR，这也与以往研究相符。

表 1 不同设置下的分组成功接收率(PSR)(%)

	关闭载波监听		打开载波监听	
	Friis	Friis+Rayleigh	Friis	Friis+Rayleigh
单干扰源	57	45	95	67
双干扰源	37	26	88	67

(下转第 313 页)

[2] 王军,李宁,胡南,等. 一种基于策略的混合式异构无线资源管理架构[C]//2008 年中国通信学会无线及移动通信委员会学术年会论文集. 2008

[3] 董全,李建东,赵林靖,等. 基于效用最大的多小区异构网络调度和功率控制方法[J]. 计算机学报,2014(2):373-383

[4] Amin R, Martin J, Zhou Xue-hai. Smart Grid communicationusing next generation heterogeneous wireless networks[C]// 2012 IEEE Third International Conference on Smart Grid Communications (SmartGridComm). 2012:229-234

[5] Patel A, Aparicio J, Tas N, et al. Assessingcommunications technology options for smart grid applications[C]// 2011 IEEE International Conference on Smart Grid Communications (SmartGridComm). 2011:126-131

[6] 田霖,翟国伟,黄亮,等. 基于集中式接入网架构的异构无线网络资源管理技术研究[J]. 电信科学,2013(6):25-31

[7] Brueck S. Heterogeneous networks in LTE-Advanced[C]// 2011 8th International Symposium on Wireless Communication Systems (ISWCS). 2011:171-175

[8] 张小朋,邢远. 泛在网络理念下文化遗产信息化建设的研究[C]// 区域特色与中小型博物馆——江苏省博物馆学会 2010 学术年会论文集. 江苏省博物馆学会,2010

[9] 兴荣. 基于效用最大的多小区异构网络调度和功率控制方法[J]. 计算机学报,2015(2):373-383

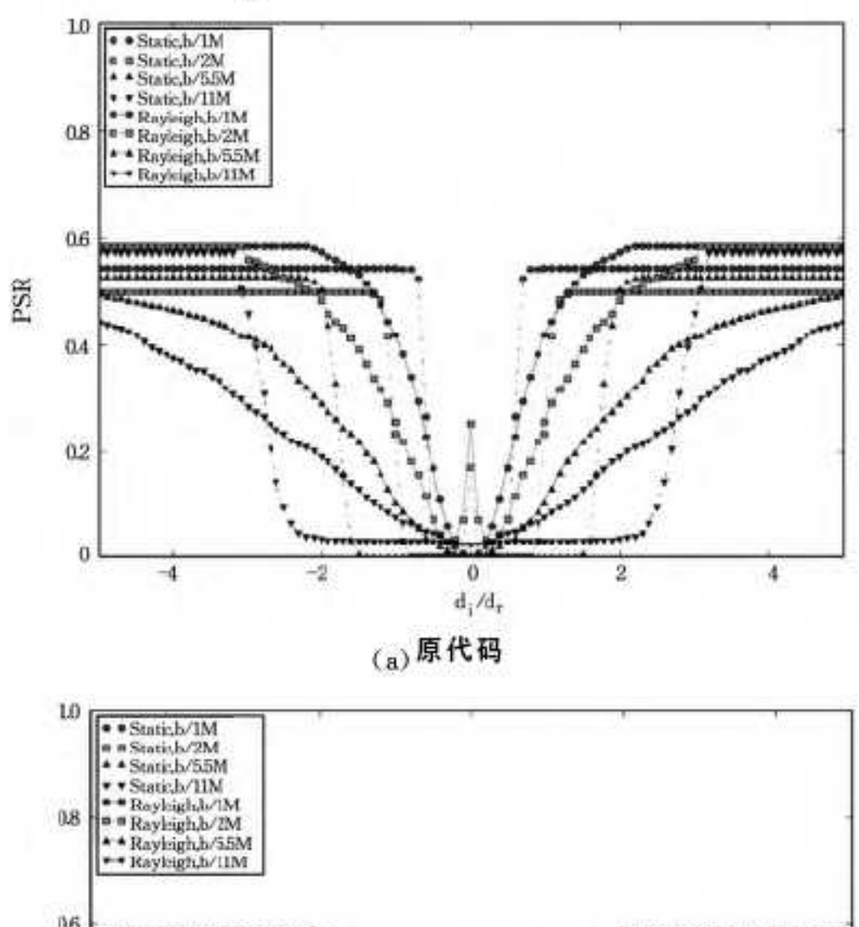
[10] 信息与电子科学和技术综合专题组. 2020 年中国信息与电子科学和技术发展研究[C]// 2020 年中国科学和技术发展研究(上). 中国土木工程学会,2004

(上接第 284 页)

5 协议修改的建议

把 NS-3 的协议修改分为参数级、方法级和类级 3 种。这 3 种修改对源代码的了解程度从低到高。

参数级协议修改:有时我们不满足于默认参数,想对某个网络元素进行参数调整以查看其对性能的影响。NS-3 的仿真配置脚本是 C++ 程序,可以在其中对任何网络元素设置参数。每个网络元素的各种参数通常存储于其所对应的对象的成员变量里。例如,通过调用 YansWifiPhy 对象的 SetEdThreshold() 方法可以设置接收阈值(即成员变量 m_edThresholdW 的值)。参数级的协议修改一般只涉及到对仿真配置脚本的修改。



(a)原代码

(b)代码修正后

图 3 在 $d_i/d_r=0$ 的 2Mbps(方形点)异常大的分组接收成功率问题

方法级协议修改:有时需要对某个类的方法进行改进。这时涉及到修改 NS-3 的源程序。操作步骤如下:首先,在 src 子目录下打开相关的类,定位到相应的方法,对其代码进行改进;然后,在 NS-3 的主目录下运行./build.py,对 NS-3 进行重新编译;最后,运行仿真脚本进行测试。

类级协议修改:有时需要对类进行较大的修改以实现新协议。这类修改的难度大,耗时长。因为 NS-3 的一个网络对象在工作中会用到许多其它对象,所谓"牵一发而动全身",开发人员必须对所涉及的类的源代码有深入细致的理解,才能保证改动不会导致逻辑错误。

这里演示一个方法级协议修改的实例——修改误码率计算方法。在仿真中我们注意到一个异常情况:在图 2 的单干扰源拓扑中,当干扰节点与接收节点在同一位置时($d_i/d_r=0$),$SINR$ 极小,误码率极高,接收成功率接近 0;但是无论在静态信道(static)还是瑞利衰落信道(rayleigh)下,2Mbps 比特速率却给出了高达 25% 的分组接收成功率,如图 3(a)所示。仔细考察发现,计算误码率的 DqpskFunction() 方法在此情况下给出大于 1 的误码率,因此导致了式(1)中的分组接收成功率 PSR 反而大于 1。我们加入了当误码率大于 1 时将其置为 1 的一行代码,纠正了这个问题,如图 3(b)所示。此问题将报告给相关开发者。

参 考 文 献

[1] NS-3[OL]. http://www.nsnam.org

[2] Lacage M, Henderson T R. Yet another network simulator[C]// Proceeding from the 2006 workshop on ns-2: the IP network simulator(WNS2'06). ACM,2006:12

[3] Pei G, Henderson T. Validation of ns-3 802.11b PHY model [Z]. Boeing Research Technology,2009

[4] Pei G, Henderson T. Validation of OFDM error rate model in ns-3[Z]. Boeing Research Technology,2010

[5] Chen B, Yenamandra V, Srinivasan K. FlexRadio: fully flexible radios and networks[C]//NSDI'15. 2015:205-218

[6] Yu H, Bejarano O, Zhong L. Combating inter-cell interference in 802.11ac-based multi-user MIMO networks[C]//Mobicom'14. ACM,2014:141-152

[7] Dale N, Lewis J. Chapter 7: Problem Solving and Algorithm, Computer Science Illuminated (5th edition)[M]. Jones & Bartlett Learning,2012

[8] 王悦,刘灿涛,朱雷,等. NS-2 无线物理层实现与衰落功能扩展研究[J]. 计算机工程,2014,40(7):58-61

[9] 廖海帆. LTE 无线资源调度算法研究及 NS-3 仿真平台实现[D]. 北京:北京邮电大学,2015

[10] Wang Yue, Yue O C. A Tutorial of 802.11 Implementation in NS-2[R/OL]. http://www.docin.com/p-677082566.html