NS2 中新协议的实现①

Implementation of New Protocol in NS2

黄镇建 (韩山师范学院 物理与电子工程系 广东 潮州 521000)

摘 要： 网络仿真软件 NS2 是网络性能分析、评估网络设计方案及网络故障诊断的强有力工具。它不仅能实现复杂的网络数据传输和拓扑结构的仿真,还能模拟各种网络环境。NS2 作为开源软件缺少对新协议的模拟能力,因此,在现有基础上对其进行功能扩展极其必要。本文首先分析了 NS2 网络仿真器的体系结构，然后介绍了如何在 NS 环境下编程实现用户的新协议，最后给出了新协议 mydroptail 的仿真结果，仿真实验表明，该协议具有追踪 Droptail 队列长度的功能。

关键词： 仿真 功能扩展 NS2 新协议 队列长度

1 引言

网络仿真[1]是使用计算机技术构造网络拓扑、实现网络协议的模拟网络行为。它能获取特定的网络参数，进而可对参网络性能进行研究和分析。在网络研究中,网络仿真扮演着越来越重要的角色,它给研究人员提供了一个方便、高效的验证和分析方法。在众多的仿真软件中，NS2(Network Simulator Version2)是一个非常优秀的 IP 网络仿真软件。NS2[2]是由美国加州大学的 LNBL 网络研究组于 1989 年开发的一个开放源代码的网络仿真软件，并一直处于不断完善之中，其开放性和灵活的可扩展性受到广大网络研究者的好评。

网络研究人员的任务之一是要不断研究新的网络协议和算法，为网络发展做前瞻性的基础研究。在研究新的网络协议和算法的过程中，仅仅进行理论分析是不够的，还需要对其模拟验证，由于是新的协议,已有的模拟软件往往无法直提供该协议算法，NS2 也不例外。这时，就需要对 NS2 的功能进行扩展，然后才能按照需要，定义网络拓扑结构，设定网络参数，模拟网络行为，对新协议的性能进行验证。然而，国内介绍和研究 NS2 的文献较少，许多初学者对 NS2 知之不多，对于如何在 NS2 平台中开发一个新协议更是不知如何下手。

队列管理机制是当前研究的一个热点，队列管理最重要的目的就是降低隐定状态下队列的长度，因为端到端的延迟主要就是由于分组在队列排队等待造成的。最简单的队列管理机制就是 Droptail,但 NS2 中并没有提供追踪 Droptail 队列长度的功能，研究者往往不能通过追踪队列长度来分析和评价网络性能，因此本文的主要任务就是开发一个能追踪 Droptail 队列长度的新协议 mydroptail,希望对利用 NS2 研究队列管理机制的科研人员有所帮助。

2 NS2 体系结构

NS2 是一种可扩展、可重用、基于离散事件驱动、面向对象的网络仿真工具。NS 由使用 C++实现的编译层和 OTCL 实现的解释层两部分构成。用户使用 OTCL 库中的对象编写并运行仿真脚本[3]。NS2 中的构件一般都是由相互关联的两个类来实现，一个在 C++中，一个在 OTCL 中。构件的主要功能通常在 C++中实现，而对编写仿真脚本研究网络时，由于实验的需要，常常改变协议对象的配置，这时用户需要一种方便改变网络配置的手段，解释性语言 OTCL 能

① 收稿时间:2008-06-21

很好满足这方面的需要[4]。NS2 中，C++中的类和 OTCL 中的类通常具有对应关系，两边类的继承关系也一致，每当实例化一个构件时，都会同时创建一个 OTCL 对象和一个对应 C++对象，并且这两个对象可以互操作。C++对象和 OTCL 对象之间通过 TclCL 机制关联起来，使用如图 1 所示的分裂对象模型。分裂对象模型中的 OTCL 类称为解释类，与之对应的 C++类称为编译类，并称它们互为“影像类”。

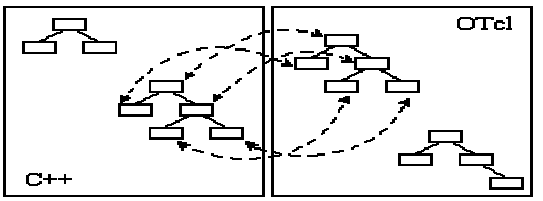

图 1　分裂对象模型

3　NS2 中新协议的开发

通常，用户要开发自已的新协议时，需要用 C++实现具体的算法模块，用 OTCL 脚本来进行网络仿真[4]。下面结合本文的主要任务（开发一个能追踪 Droptail 队列长度的新协议 mydroptail）进行说明。

3.1 定义包头文件

NS 的数据包的包头定义在 packet.h 中，在这个文件中定义了两个结构体 hdr_ip 和 hdr_cmn,其中 hdr_cmn 是 NS 定义用来跟踪数据流和测量其他特性用的，而 hdr_ip 是 TCP/IP 协议 IP 包头的定义。因为本文的主要工作是追踪 Droptail 队列长度，所以不用改变 hdr_ip 和 hdr_cmn 的结构。在 NS 中，凡是派生自 NsObject 的类，都可以实现对成员变量的跟踪记录，为了实现这种机制，我们必须定义一个 Tcl_Channel 类型的成员变量和一个文件关联，然后再定义我们要跟踪的变量类型（TracedDouble 或者 TracedInt）。

```
Tcl_Channel tchan_; // 用以和存放记录的文件关联
TracedInt curq_;  //存放队列长度
```

3.2 定义 C++代码和 OTCL 代码之间的接口

假设用 C++语言编写了一个新协议，现在要通过 OTCL 来调用此新协议，这就需要定义一个 TclClass 类派生出的接口元素，让它生成一个特定的名称的 OTCL 对象，使之与 C++类建立一对一的映射，通过 create 成员函数调用 C++类。本文开发的协议代码如下：

```
static class mydroptailClass:public TclClass{
public:
    mydroptailClass( ):TclClass("Queue/mydroptail") { }
    TclObject *create(int,const char*const*){
    return(new mydroptail);}
}class_my_droptail;
```

当 NS 启动的时候，首先执行静态变量 class_my_droptail 构造函数，于是产生实例 mydroptail,在这个过程中，Queue/mydroptail 及其成员函数均在 OTCL 空间中被创建出来。用户需要生成这个对象的一实例时，只需要利用 OTCL 命令 new Queue/mydroptail，NS2 就会通静态变量 class_my_droptail 调用 create 函数创建一个 mydroptail 对象。

3.3 编写类成员函数和协议算法

如何让新开发的协议 mydroptail 能实现队列长度的追踪呢？我们必须完成两项工作，一是让一个 Tcl_Channel 类型的成员变量和一个文件关联；二是当跟踪的变量的值改变的时候，通过 trace 函数将其记录到文件中。为了介绍其实现方式，下面列出该类成员函数 command 的一部分。

```
int mydroptail::command(int argc,const charconst*argv){
    Tcl&tcl=Tcl::instance( );
    if(argc= =3){
      if(strcmp(argv[1],"attach")= =0){
        int mode;
        const char*id=argv[2];
        tchan_=Tcl_GetChannel(tcl.interp( ),(char*)id,&mde);
        if ( tchan_= =0){
          tcl.resultf("mydroptail:trace:can't attach %s for writin",id);
          return(TCL_ERROR);}
    return(TCL_OK);}
}
```

当用户跟踪的变量的值改变的时候，trace 函数被触发，我们可以将变量改变的时间和相关信息写入到用户所希望的格式文件中。

```
void mydroptail::trace(TracedVar*V)
```

```
{ char wrk[500],*p;
//判断用户希望跟踪的变量本函数是否支持
  if (((p=strstr(v->name( ),"curq"))= =NULL)){
    fprintf(stderr,"mydroptail:unknown trace var %s\n",v->name( ));
    return;}
//将变量值以规定的格式写入到一个字符串中
    if(tchan_){
      int n;
      double t=Scheduler::instance( ).clock( );//获得跟踪变量值改变的时间
      if (strstr(v-name( ),"curq"!=NULL{
      sprintf(wrk,"Q%g%d",t,int(*((TracedInt*)v)));}
    n=strlen(wrk);
    wrk[n]='\n';
    wrk[n+1]=0;
    (void)Tcl_Write(tchan_,wrk,n+1);//将字符串写入到文件中}
  return;}
```

接下来要完成的工作就比较简单了，我们只需将编写的 mydroptail.cc 和 mydroptail.h 放入到 NS 目录下，并且改动 makefile 文件，将 mydroptail.o 文件加入到 OBJ_CC 内，运行 make 命令，重新编译 NS。

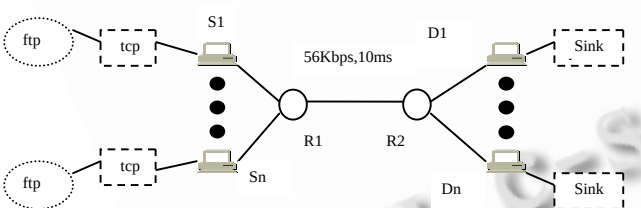

图 2 仿真实验拓扑图

4 仿真实验

完成了协议的编写和 NS 的重新编译，接下来要完成的工作就是编写仿真脚本，利用我们上述的新协议来追踪 Droptail 队列长度了。首先我们建立如图 2 所示的仿真实验网络拓扑，图中 R1 和 R2 是路由器，它们之间的链路带宽为 56k bps,延迟时间为 10ms,队列管理机制为 droptail,R1 和 R2 之间建立 10 条 ftp 连接，这 10 条 ftp 连接的传送时间在 0 到 1 秒之间随机产生。现在我们利用上面编写的 mydroptail 来追踪 Droptail 队列长度，以下为部分 OTCL 脚本。

```
$ns duplex-link $R1 $R2 56k 10ms mydroptail
$ns queue-limit $R1 $R2 50
#跟踪队列变量，R1 和 R2 之间使用 mydroptail 队列
set  q_  [[$ns link $R1 $R2] queue
#创建文件 mydroptail.tr ,将跟踪结果写入到文件中
set  tchan_  [open mydroptail.tr w]
$ q_ trace curq_
$ q_ attach tchan_
```

仿真结果产生一个有关队列长度的数据文本 mydroptail.tr，提取文本的数据并调用 gnuplot 作图，图 3 为显示结果

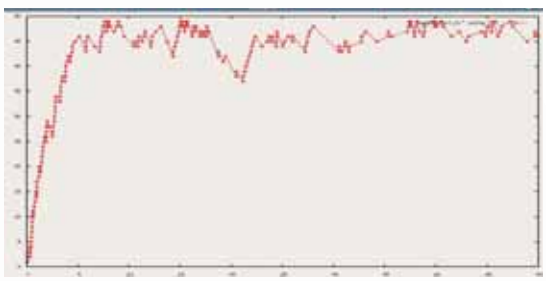

图 3 仿真结果图

5 结束语

NS2 是一个开放的仿真平台，它采用了两级体系结构：编译层和解释层，用户可以通过继承 NS 类来开发自己的新协议，集成到 NS 环境中去。本文详细的说明了在 NS2 中开发一个新协议的方法和步骤，对于需要在 NS 中加入新元素的仿真实验来说是有一定的实际意义。

参考文献

1 杜伟,王行刚.网络仿真在网络性能指标评价中的应用.计算机工程与应用,2003,18:176–180.
2 颜昕,李腊元.NS 的仿真机制及协议扩展.武汉理工大学学报,2004.
3 刘俊,徐昌彪,隆克平.基于 NS 的网络仿真探讨.计算机应用研究,2002,3:54–57.
4 徐雷鸣,庞博,赵耀.NS 与网络模拟.北京:人民邮出版社,2003.