基于 NS－2 的网络仿真与扩展

Network Simulation and Protocol Extension based on NS－2

陈亚军　肖建华　（武汉科技大学信息科学与工程学院　430081）

摘要：由于网络本身的复杂性，要分析网络性能显得比较困难。然而随着计算机技术的发展，仿真已经成为一种分析复杂系统的有效的工具。该文介绍了目前应用较为广泛的网络仿真器 NS－2。首先详细介绍了网络仿真器 NS－2 的结构、功能及使用方法，给出一应用实例，其次介绍 NS－2 的扩展方法。

关键词：网络仿真　NS

1　引言

网络仿真技术是一种通过建立网络设备和网络链路的统计模型，并模拟网络流量的传输，从而获取网络设计或优化所需要的网络性能数据的仿真技术。

网络仿真作为一种行之有效的、对实际网络进行模拟与分析的方法，具有可信度高、使用范围广、应用成本低、可以提供真实和实用系统的可行性依据等特点，在通信网络技术的研究中有着重要的意义。未来数年将是网络仿真技术蓬勃发展的时期，它已经成为数据网络规划设计不可缺少的工具。

2　NS－2 的功能与特点

NS－2 仿真器作为仿真领域受欢迎和使用范围广的共享软件，主要特点有：

2.1　NS－2 对仿真类进行了封装

NS－2 对仿真类的封装为其在使用 OTcl 进行脚本语言的编程时提供了极大的方便。目前 NS 仿真器用到了 6 种 TCL 类，它们是：

（1）Tcl 类是 C＋＋代码与 Tcl 代码之间的桥梁。

（2）TclObject 类是所有仿真对象的基类。

（3）TclClass 类定义了解释类的类层次，并允许用户实例化 TclObject，与 TclObject 一一对应。

（4）TclCommand 类封装了 C＋＋代码和 Tcl 代码相互调用命令的方法。

（5）EmbeddedTcl 类封装了装载更高级别的内置命令的方法。

（6）InstVar 类定义了实现绑定机制的方法。

2.2　派生类

NS－2 对网络实体的仿真和各种功能模块都封装在派生类中。按照仿真模块来介绍比较重要的派生类，其中 Simulator 类是对整个仿真器的封装，含成员类 Node、Link、Agent、Package、LAN 等。Node 类主要仿真网络中的节点，它的主要组分是 Classifier 类。Classifier 类实现了部分路由功能。Link 类仿真链路，它的主要组分是 Connector、Queue、Delay。对各层协议的仿真由 Agent 类实现，它与计时器 timer 合作。Package 类模拟数据包，由于局域网的特性，NS2 又实现了 LAN 类。

（1）Simulator 类是一个解释类，没有相应的编译类。但它是由许多更小的类构成的，这些类有相应的编译类。

（2）Node 类也是一个由 OTcl 实现的解释类，也没有相应的编译对象。

（3）Link 类是 OTcl 中的一个标准类，提供了一些简单的功能。一个简单的链路对象是由一些连接器（connector）组成的。

（4）Agent 可以在不同层上实现各种协议。

（5）Package 类是仿真对象间交换数据的基础单元，它提供了足够的信息，可以将一个包联入一个列表，可以查询数据包头缓冲，可以查询包数据缓冲。

3　NS 仿真模型

网络建模是实现网络仿真的基础，NS 采取对真实网络元素进行抽象、保留其基本特征，并运用等效描述

的方法来建立网络仿真模型. NS 的仿真模型包括拓扑模型、协议模型和流量模型三个部分,它们由大量的仿真组件(由 C + +和 OTcl 编写)所构成,用于实现对真实网络的抽象和模拟。

3.1 拓扑模型

由节点和链路构成。从本质上讲,节点是分类器(classifier)的集合,链路则是连接器(connector)的集合。

节点主要由地址分类器、端口分类器、多播分类器和复制器等仿真组件构成。

目前,NS 所能支持的队列类型包括 Drop – Tail(FIFO)队列、RED 缓冲管理、CQB 队列(包括优先权调度和 RR 调度)以及公平队列,如 FQ,SFQ 和 DRR 等。

3.2 协议模型

由协议代理及其辅助功能模块组成。协议代理是依附于节点、用于模拟网络各层协议工作过程的仿真组件,辅助功能模块则是协议代理的协调器和管理器。

在节点上,配置不同的代理可以实现相应的协议或其它模型仿真。如 TCP 代理,发送代理有:TCP,TCP/Reno,TCP/Vegas 等,接收代理有:TCPSink,TCPSink/SACK1 /DelACK 等。此外,还提供有 UDP 代理及接收代理 Null(负责通信量接收)、LossMonitor(通信量接收并维护一些接收数据的统计)。

3.3 流量模型

由流量发生器组成。流量发生器是应用层基类 Application 的派生类,其作用是模拟应用程序产生网络通信量。NS 提供了 4 类流量发生器:(1) Expoo;(2) Pareto;(3) CBR;(4) Trace 发生器可以通过读取跟踪文件中记录流量的数据来复制通信量。

4 用 NS –2 进行网络仿真

仿真一般要经过建立模型、模拟实现和结果分析三个过程。NS 的仿真分为两个层次:一是用户层次,用户直接利用 NS 已有的仿真组件和模型进行网络的模拟和分析,而无需对 NS 本身作任何修改和扩展,即通常所言的 NS 仿真;二是系统层次,用户使用 OTcl 和 C + +语言对 NS 本身进行开发和扩展,实现新的协议和算法。

4.1 OTcl 实现仿真的一般过程

(1) 创立仿真器并配置或构造仿真网络拓扑(包括链路和节点)。仿真之前首先要构造一个基本的网络拓扑平台。此时,可以确定链路的基本特性,如延迟、带宽和丢失策略等。

(2) 建立协议代理,包括端设备的协议绑定和通信量模型的建立。

(3) 给节点进行特性化配置。根据仿真具体要求对节点进行代理、路由协议等的初始化。

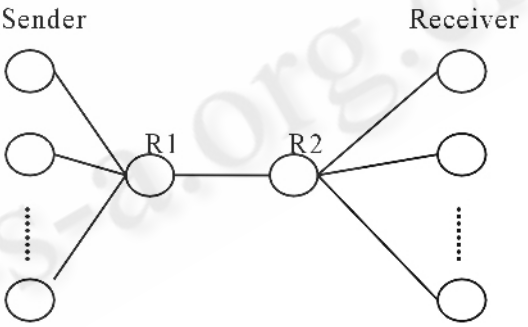
(4) 编写必要的 OTcl 过程或构造可能需要的 OTcl 类,如记录过程等。

(5) 进行仿真结果的追踪。通过记录跟踪变量和建立追踪文件来保存仿真期间网络性能的参数变化。仿真完成后,调用相应观察器(如 Xgraph,Nam)对结果文件进行分析、研究。

(6) 在建立了上述代码后,设定通信量应用和时间相关过程的发送/结束时间,然后运行仿真。

4.2 一个 NS 仿真实例

在本例中,我们使用下面的拓扑图进行仿真,R1 与 R2 之间是一瓶颈链路(5Mb),共 10 个连接,我们每 0.1s 统计流过瓶颈链路的数据包并将其写到文件 P1.txt 中。




```
# 创建仿真器实例,设置连接数,生成瓶颈节点
set ns [new Simulator]
set conn 10
set nodenum [expr 2 * $conn]
for { set i 0} { $i < $nodenum } { incr i } { set node( $i) [ $ns node]}
set node(r1) [ $ns node]
set node(r2) [ $ns node]
# 设置节点之间的链路
for { set i 0} { $i < $conn } { incr i } {
  $ns duplex – link $node( $i) $node(r1) 10Mb 0.1ms DropTail
```

```
 $ns duplex – link $node([expr $conn + $i])
$node(r2) 10Mb 0.1ms DropTail
}
$ns duplex – link $node(r1) $node(r2) 5Mb 5ms
RED
$ns queue – limit $node(r1) $node(r2) 10
# 设置瓶颈链路流监视器
set fmon [$ns makeflowmon Fid]
$ns attach – fmon [$ns link $node(r1) $node
(r2)] $fmon
# 统计流过瓶颈链路以及被丢弃的包
set retransmit 0
proc stats {} {
 global fmon tcp retransmit conn
 set fclassifier [$fmon classifier]
 set parrtcp [$fmon set parrivals_]
 set pdropstcp [$fmon set pdrops_]
 for {set i 0} {$i < $conn} {incr i} {
# 利用 NS – 2 的跟踪变量,我们可以获取很多变量的
值,这里我们监视重传的包数
  set tmp [$tcp($i) set nrexmitpack_]
  set retransmit [expr $tmp + $retransmit]
 }
 puts "Enqueue:$parrtcp Drops:$pdropstcp Retrans-
mit:$retransmit"
}
# 设置发送端与接收端代理,生成业务源
for {set i 0} {$i < $conn} {incr i} {
 set tcp($i) [new Agent/TCP]
 set sink($i) [new Agent/TCPSink]
 $sink($i) set window_ 300
 $ns attach – agent $node($i) $tcp($i)
 $ns attach – agent $node([expr $conn + $i])
$sink($i)
 $ns connect $tcp($i) $sink($i)
 set ftp($i) [new Application/FTP]
 $ftp($i) attach – agent $tcp($i)
 $ns at 0 "$ftp($i) start"
}
$ns at 10.0 "finish"
set P1 [open P1.txt w]
 $ns at 0 "record"
# 一个周期执行过程,统计离开瓶颈链路的数据包个
数
proc record {} {
    set ns [Simulator instance]
    global tcp node fmon P1
    set now [$ns now]
    set time 0.1
    set now [$ns now]
    puts $P1 [$fmon set pdepartures_]
    $ns at [expr $now + $time] "record"
}
# 调用 stats 进行统计
proc finish {} {
    stats
    global ns P1 conn
=
    $ns flush – trace
    close $P1
    exit 0
}
$ns run
```

程序运行将输出:

Enqueue:6631 Drops:608 Retransmit:613,**同时还将生成一个文件** P1.txt。

5 NS 的扩展

在 NS **下进行仿真模块开发的第一步是用一种能和现有** TCP/IP **仿真器比较容易地集成的方法来表示网络,为此目的需要设计不同的类和各种函数来表示网络拓扑结构、集成整个系统。第二要考虑的问题是用户接口,获得用户输入,用户进入网络之后,整个网络及其资源、传送的业务类型、交换等都要初始化。第三要建一个仿真引擎来驱动整个仿真过程。仿真引擎可以选择的驱动方式有时间循环、分组循环和事件驱动。一般地说,事件驱动是一种更有效、更有弹性的模型。**

5.1 代码编写

用户应该根据目标模块的功能,详细分析编译层

的类层次结构,然后确定类的继承关系。例如用户通过 Agent 类创建了一个新类 MyAgent,代码略。

由于用户的开发是在编译层,为在解释层灵活地引用新的类对象,集成新的对象就需要在 NS－2－2 中建立与 C＋＋代码的 Otcl 连接,也就是在编译层与解释层之间通过建立 Otcl 连接实现对象的映射,这样,用户就能够在仿真脚本中方便地使用该对象了。

5.2 Otcl 连接

(1) 实现类的映射。在 Otcl 中创建 MyAgent 类的实例需要在编译层定义一个连接对象,称之为"MyAgentClass",这个对象是从 TclClass 继承而来的,其代码略。

MyAgentClass 在它的构造函数中创建了一个名为"Agent/MyAgent"的 Otcl 对象,与 MyAgent 类具有对应关系。当 NS－2 启动时,执行 MyAgentClass 类的构造函数,从而生成了 MyAgentClass 类的实例,在这个构造函数中,创建了 Otcl 环境的 Agent/MyAgent 类及其方法。如果用户在 Otcl 脚本中使用"new Agent/MyAgent"命令,系统就会调用 MyAgentClass 的 create 方法,产生一个 MyAgent 类对象,这样就在 C＋＋对象和 Otcl 对象之间建立了映射关系。

(2) 实现变量的映射。如果要在 Otcl 脚本中对 C＋＋成员变量进行访问或配置,用户必须建立变量自己的映射关系(即变量绑定),例如上例中的 MyAgent 类有两个变量 myval 和 myval2 下面的代码将这两个变量映射到了 Otcl 空间:

```
MyAgent::MyAgent():Agent(PT－UDP){
 bind("myval_",&myval);
 bind("myval2_",&myval2);
}
```

变量的绑定过程一般是在编译类对象初始化时执行构造函数完成的。构造函数利用绑定函数 bind() 在 Otcl 空间的对应类结构中为 myval 和 myval2 分别创建了一个指定的成员变量 myval_和 myval2_,在 C＋＋与 Otcl 之间建立了两对成员变量的映射。用户在 NS－2/tcl/lib/ns－default. tcl 中设置了变量的默认值后,就可以在仿真程序中使用 myval_和 myval2_来访问或修改变量 myval 和 myval2。

(3) 在 Otcl 中实现对 C＋＋对象的控制或在 C＋＋中通过使用 Tcl 类的实例执行 Otcl 命令。上例中,通过 command 函数可以实现对 C＋＋对象的控制。

这样用户可以在 Otcl 中调用 my－func 功能,同样,C＋＋可以通过使用 Tcl 类的实例执行 Otcl 命令,例如函数 MyPrivFunc 的实现:

```
void MyAgent::MyPrivFunc(void){
 Tcl &tcl=Tcl::instance();
……
}
```

5.3 使用自定义模块

```
set my [new Agent/MyAgent]
 $my set myval_ 2
 $my set myval2_ 3.1
 $my set my－func
……
```

通过上述的分析,用户可以很容易通过扩展 NS－2。

总结而言:在 NS 中扩充新协议(包括代理,流量模型,队列管理算法等)的一般步骤如下:

(1) 定义或者继承 C＋＋协议类
(2) 定义 TCL 相关的类和变量
(3) 把 C＋＋绑定到 TCL
(4) 在 ns－default. tcl 中为新增参数设置缺省值
(5) 重新编译 NS－2

6 总结

文章对 NS－2 仿真器的原理、系统知识、使用、扩展等进行了介绍。同时,我们利用 NS－2 仿真器进行了一个仿真实验。对于研究 TCP 拥塞控制,优化网络设计、开发新的队列管理算法,设置合理的网络参数等有实际的参考价值。

参考文献

1 http://www.isi.edu/nsnam/ns/
2 http://nile.wpi.edu/NS/
3 刘俊、徐昌彪、隆克平,基于 NS 的网络仿真探讨[J],计算机应用研究,2002,19(9):54－57。
4 王宇、赵千川,用网络仿真软件进行 IP 网络的仿真[J],计算机应用与软件,2003,20。
5 颜昕、李腊元,NS 的仿真机制及协议扩展[J],武汉理工大学学报(交通科学与工程版),2004,2。