网络仿真器 NS3 的剖析与探究

茹新宇[1,2],刘 渊[1]
(1. 江南大学 数字媒体学院,江苏 无锡 214122;
2. 江苏联合职业技术学院 无锡交通分院,江苏 无锡 214151)

摘 要:随着虚拟机、云计算及大数据等高带宽、异构化网络的普及,当前主流仿真工具 NS2 的局限性已日益凸显,新一代离散事件模拟器 NS3 应运而生,但其相关资料甚少,使用困难。为满足学术研究及教学需求,从多个角度、不同层次对 NS3 作了完整综述。首先简单概述了 NS3 及其与 NS2 的区别,随后系统剖析了其模块体系、基本对象与类、数据传输、代码体系及仿真流程。最后又从理论深度重点探究了 NS3 在研究领域的 TCP 实现过程,详细阐述了类的相互作用、全局变量及其算法实现细节。NS3 虽不具有 NS2 的所有模型,相关研究也不多,但其符合弹性化网络模拟需求,拥有单一的语言、良好的架构及基于底层的开发设计等特点。

关键词:网络仿真器;类;模型;应用程序接口;套接字;继承;带宽
中图分类号:TP393 **文献标识码**:A **文章编号**:1673-629X(2018)03-0072-06
doi:10.3969/j.issn.1673-629X.2018.03.015

Analysis and Research on Network Simulator 3

RU Xin-yu[1,2],LIU Yuan[1]
(1. School of Digital Media,Jiangnan University,Wuxi 214122,China;
2. Wuxi Transportation College,Jiangsu Union Technical Institute,Wuxi 214151,China)

Abstract:With the popularity of heterogeneous networks with high bandwidth like virtual machine,cloud computing and big data,the mainstream network simulator 2 is limited gradually and obviously. The NS3,as a novel generation of discrete event simulator,comes into being,but its related information is few and difficult to use. In order to meet the academic research and teaching needs,we make a complete review of NS3 from a number of different levels. Firstly,we briefly summarize the NS3 and the difference between NS3 and NS2. Then,we analyze systematically the module,basic object and class,data transmission,code and simulation process of NS3. Finally,the TCP implementation of NS3 in the field of research from the theoretical depth is explored,and the interaction of the class,the global variables and the implementation details of the algorithm are elaborated. NS3 does not have all the NS2 models,with not many related studies,but it is in line with the needs of flexible network simulation,with a single language,a good architecture and the underlying-based development and design.

Key words:network simulator;class;model;application program interface (API);socket;inherit;bandwidth

0 引 言

仿真所固有的低成本、虚拟化优势,使其成为极具吸引力的重要科研手段[1]。网络仿真目前已是互联网算法性能、协议拓扑最经济快捷的评价方法之一。网络仿真器主要通过部署虚拟环境,允许对特定节点或路径模拟其网络行为,并提供仿真结果,以便研究、参考或改进。借助仿真的高灵活度和可扩展性,不同网络实体的协议算法、概念模型及其拓扑架构可在虚拟环境中搭建、测试并实施。现有工具,如 QualNet、OPNET[2] 和 REAL 等,由于使用条款、实施成本及代码开源等因素而远未普及。鉴于现场布置及情景再现等客观条件,新网络仿真器 NS3 应运而生。作为 NS2 的继任者,它运用全新理念来解决和减轻 NS2 存在的问题,现已大量部署于实验场景。它支持当前主流协议,为实验提供仿真结果。

文中对 NS3 进行了深入探究,重点阐述了其 TCP

收稿日期:2017-04-16 **修回日期**:2017-08-24 **网络出版时间**:2017-12-05
基金项目:国家自然科学基金(61602213);江苏省自然科学基金(BK20151131)
作者简介:茹新宇(1977-),男,硕士,讲师,CCF 会员(74533M),研究方向为网络拥塞控制、网络安全、流量建模;刘 渊,博导,教授,CCF 高级会员(E200006802S),研究方向为网络测量、网络安全及网络信息系统。
网络出版地址:http://kns.cnki.net/kcms/detail/61.1450.TP.20171205.0904.038.html

实现过程,并提出了具有前瞻性的研究方向。

1　NS3 简介

1.1　NS3 的概述

NS 系列仿真器是由 UC Berkeley 开发的优秀网络仿真软件,使用经典的 Unix 语言作为环境工具,在 GNU/Linux 平台下开发[3]。为便于安装、维护及更新,如今已被移植进 Windows/Cygwin, BSD 及 Mac OSX 等系统。NS2 作为一款开源软件,支持多种协议且易于扩展,但同时也存在较多问题。如它采用分裂对象模型,在 OTcl 中编写脚本,仿真结果由 NAM 实现可视化,却无法单纯从 C++运行。另外,网络层抽象仿真,结果跟踪困难,需借助解析文件提取,模型之间互操作及耦合性不足,分析工具"各自为政"。

目前仍在大规模开发的 NS3 项目始于 2006 年,它广泛汲取主流仿真器 NS2、YANS 和 GTNets 的技术经验,用 C++语言实现,兼容时下流行的 Python,目前已发展至 3.26 稳定版本。NS2 部分模型已移植进 NS3,使之在功能实现、版本更新、用户体验等方面都具有良好表现[4]。其核心及各功能模块由 C++代码完成[5],对外提供丰富的扩展接口[6],可按需更改或增减模块。NS3 包含网络组件模拟接口,拥有事件调度器,可通过执行相关事件,模拟真实通信"行为"。同时它还具备完美的跟踪机制,便于用户解析数据、传输过程及分析结果,详细介绍如表 1 所示。

表 1　NS3 仿真器的优点与特点

优　点	特　点
仿真预测性好、应用成本低、可靠性高、效果好	用 C++设计新核心,可选 Python 脚本接口,用于提高可扩展性、模块化编程和文档编写
能直观观察协议的处理,了解各种环境或因素对网络的影响,便于比较展示各种策略的优缺点	网络节点包括套接字、网络设备等关键接口采用现实化设计,每个节点的多个接口使用 IP 地址
仿真结果可复制、易分析	架构支持软件集成,减少端口或重写模型
预定义构造拓扑对象,通过代码实例化,可配置对象参数,方便使用	通过仿真网络运行本地"进程",支持各类虚拟机

1.2　NS3 与 NS2 的区别

NS2 及其后继者 NS3 具有相同背景、概念和类似目标,是一离散事件模拟器。虽然目前 NS2 仍有很大用户群,但由于 NS3 的特征及设计优势,使之日趋成为 NS2 的可靠替代品。它可较好地规避前者存在的问题,其协议和场景均用 C++编写。第三方软件依赖和源代码构建过程截然不同并优于 NS2,系统集成度较好。两者的部分特征对比如表 2 所示。

表 3 总结了现 NS2 已有模型与 NS3 计划或已开发模型间的对比数据。

表 2　NS2 与 NS3 的部分特征对比

特征	NS2	NS3
首次发布	1996	2008
基于的项目	NS1 & REAL simulators	NS2, GTNets, YANS
体系结构	OTcl & C++	C++ & optional Python 脚本
脚本语言	OTcl	Python
可视化	NAM	NS3-viz, pyviz, nam, iNSpect (all under development)
可扩展性	顺序式仿真	分布式仿真

表 3　NS2 与 NS3 的项目模型对比

模型	现有 NS2 的核心功能	计划加入 NS3 的功能
应用层	ping, vat, telnet, FTP, 多播 FTP, HTTP, probabilistic and trace-driven traffic generators, web-cache	API 套接字(以允许移植的现有应用程序到 NS 环境), peer-to-peer (例如 BitTorrent)
传输层	TCP (many variants), UDP, SCTP, XCP, TFRC, RAP, RTP 多播:PGM, SRM, RLM, PLM	TCP stack emulation (Linux, BSD), DCCP, 另外高速 TCP 变体
网络层	Unicast: IP, Mobile IP, generic dist. vector and link state, IPin-IP, source routing, Nixvector, 多播: SRM, generic centralized, MANET: AODV, DSR, DSDV, TORA, IMEP	全面支持 IPv4、IPv6, 支持全部 NATXORP/Click 支持路由: BGP, OSPF, RIP, IS-IS, PIM-SM, IGMP/MLD
链路层	队列: Diffserv, DropTail, RED, RIO, WFQ SRR, Semantic Packet Queue, REM, Priority, VQ ping, vat, telnet, FTP, 多播 FTP, HTTP, probabilistic and tracedriven traffic generators, webcache, ARP, HDLC, GAF, Aloha 卫星	新的 802.11 模型, 802.11 的变体(网格, QoS), 802.16 (WiMax), TD-MA, CDMA, GPRS
物理层	TwoWay, Shadowing, OmniAntennas, EnergyModel, Satellite Repeater	IEEE802 物理层, Rayleigh 和 Rician 衰落信道, GSM

2　NS3 的系统剖析

2.1　NS3 的模块体系结构

NS3 由一系列层次分明的功能模块拼接而成,其组织结构及模块间的基本依赖关系如图 1 所示。

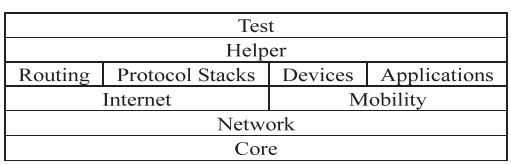

图 1　NS3 的基本模块体系结构

2.1.1　常用模块

Core:内核模块,是 NS3 基本机制的实现,如智能指针(Ptr)、属性(attribute)、回调(callback)、随机变量(random variable)、日志(logging)、追踪(tracing)和事件调度(event scheduler)等内容。

Network:数据分组(packet)模块。

Internet:关于 TCP/IPv4/6 的相关协议族的实现,包括 TCP/IPv4/6、ARP、UDP 及邻居发现等相关协议。

Topolopy-read:读取指定轨迹文件数据,按指定格式生成相应网络拓扑。

Protocol Status:协议框架模块,收集、统计和分析数据。

另有,Tools:统计工具(包括 gnuplot 作图接口);ApplicatioNS:应用模块;Mobility:移动模块;Visualizer:可视化工具;Netanim:动画演示器;Propagation:传播模型模块;Flow-monitor:流量监控模块等。

2.1.2 典型模块

CSMA:基于 IEEE802.3 以太网,包括 MAC 层、物理层和媒体信道。

Point-to-point:实现网络间的点对点通信。

Wifi:基于 IEEE802.11a/b/g 无线网,包括 Ad-Hoc。

其他还有,Mesh:基于 IEEE 802.11s 的无线网络;Wimax:基于 IEEE802.16 的无线城域网;LTE:3GPP 通用移动通信系统(UMTS)技术长期演进;UAN:水声通信网络;MPI:并行分布式离散事件标准信息传递接口;Click:集成可编程模块化路由;Openflow:仿真交换机;Emu:集成实验床和虚拟机环境等。

2.1.3 最新技术

Waf 是基于 Python 框架开发的编译工具,NS3 自身及将要执行的仿真代码都由 Waf 编译运行。Scratch 目录存放用户脚本,运行案例可拷至该目录。Example 举例如何使用 NS3,内含许多模块的使用。Doc 目录是帮助文档,可使用./waf --doxygen 编译本地 Doxygen 文档。Build 编译目录,内含编译文件时使用的共享库和头文件(build/NS3)。Src 是 NS3 的源码目录。

模块中的 wscript 文件结构固定,用来注册模块中的源码和使用其他模块情况。Model 目录包含模块代码的.cc 和.h 文件。Helper 目录存放模块对应的 helper 类代码的源文件。Test 目录包含原模块测试代码,而 examples 目录存放应用该模块的实例代码。Doc 是帮助文档,bindings 目录则被模块用来绑定 Python 语言。

2.2 NS3 的基本对象与类

NS3 以较低层次的抽象来构造组件,模拟真实环境[8],其基本对象是节点、应用、信道和网络设备等,由类表示或实现。

(1)节点(Node):网络上所连接的基本计算单元或终端都抽象为节点,在 C++中由 Node 类描述(如 Nodeontainer 类),用于追踪一组节点指针。用户可通过对节点添加应用、协议和网卡等进行二次开发[9]。

(2)应用(Application):被仿真的用户程序抽象为应用。NS3 以“Time”为参数,记录接收和发送时间。在 C++中抽象成 Application 类表示,实现时需继承该类,将其部署在节点,驱动仿真器运行。由应用程序生成和回显仿真数据包的客户/服务器端程序集,如 Application 类称为“UdpEchoClient Application”和“UdpEchoServer Application”。

(3)信道(Channel):基本的通信子网被抽象为信道。在 C++中由 Channel 类描述,提供管理通信子网对象和节点连接至它们的各种方法。它可模拟简单线缆、无线网,甚至以太网交换机。NS3 包括 Csma Channel、Point To Point Channel 以及 Wifi Channel 等信道。

(4)网络设备(NetDevice):硬件设备和软件驱动的抽象表示,由 C++的 NetDevice 类实现。通过绑定与类型匹配的信道,提供管理节点和信道对象连接。与信道对应,NetDevice 也分为 Csma NetDevice、Point To Point NetDevice 和 Wifi NetDevice 等[10]。若需要一个所有被创建的 NetDevice 对象列表,则需用一个 NetDeviceContainer 对象来存放。

(5)分组(Packet):每个分组包含字节缓冲器、标签和元数据。缓冲器是头部和尾部的逐位串行表示,标签则是任意用户提供的数据结构的集合,而元数据可用以描述已序列化的头和尾的类型。

(6)套接字(Socket):作为应用程序与网络堆栈间的接口。NS3 提供了两种类型的套接字 API,一是 NS3 API,二是使用本机 API 服务来提供类似 POSIX 的 API,作为整个应用程序进程的一部分。

(7)拓扑帮助器(Topology Helper):NS3 用 Topology Helper 类来整合大量分立步骤,使其成为一简单易用的操作。由拓扑生成器调用底层核心完成节点、网络设备、MAC 地址、信道及协议栈等创建与配置。它可实现多节点连接及多子网联网、分配 IP 地址等功能。如 Topology Reader Helper 类可简化配置并使用通用 Topology Reader。Internet Stack Helper 是个安装 Point To Point Helper 对象和点到点网络设备的网络协议栈的拓扑生成器类。

(8)典型容器助手(typical containers and helpers)。NS3 分容器类和助手类,NodeContainer、NetDevice Container 和 Ipv4AddressContainer 是不同容器类,而 InternetStackHelper、WifiHelper、MobilityHelper 和 OlsrHelper 则属不同助手类。

3 NS3 的初步探究

3.1 NS3 的数据传输

NS3 体系结构类似 OSI 模型,便于对网络层次及

协议仿真研究。数据按 TCP/IP 形式发送,以类方式实现。事件由节点触发,当数据包到达节点,仿真器按事件预定的执行时间排序队列、按步处理。数据包传递时,通过指针交互完成转发,传输过程如图 2 所示[11]。

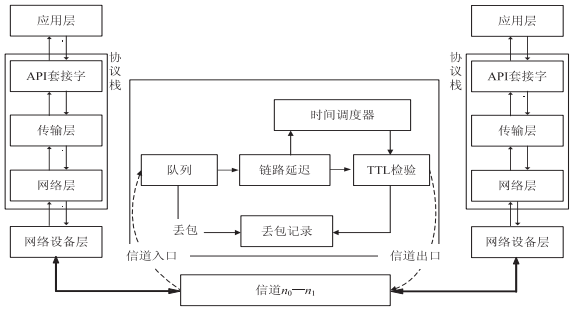

图 2　NS3 数据包传输过程

首先应用层创建数据包并通过套接字传送至传输层(用套接字指针取代应用层指向该数据包)。在该层完成 TCP 或 UDP 头的封装后转交至网络层[12]。由该层判断数据包的目的地址,若非当前节点则查询路由,将数据包交至网络设备层。该层查询 ARP,将数据包转至指定接口,由该口将包发送至信道。信道通过入队、链路延迟处理、TTL 检验及出队等完成数据包的发送。若出现队列已满或 TTL 生命期结束,则丢弃该包。当数据包经过信道传输到达目的节点后,再由网络设备层开始逐层上传,最后通过端口找到 API 套接字,并通过该套接字将包交由应用模块处理。综上所述,NS3 的数据包传输过程与物理网络类似,其仿真可信度较高。

3.2　NS3 的代码体系结构

NS3 从拓扑建立开始,定义所选模型,而后通过设置参数、赋予地址来配置模型并执行代码。仿真生成的 Pcap 包文件可采用 trace 格式跟踪输出,最后用 Wireshark 等工具跟踪结果并分析数据,或由 Net Anim 来实现可视化输出。其代码创建过程如图 3 所示。

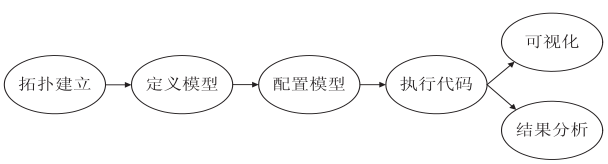

图 3　NS3 代码体系结构

3.3　NS3 的仿真流程

使用 NS3 进行网络仿真时,一般需要 4 步:

(1)选择或开发相应模块;

(2)编写仿真脚本(C++或 Python)。包括:生成节点(如网卡、应用程序和协议栈等);安装网络设备(如 CSMA、WiFi);安装协议(一般 TCP/IP 及应用层协议);其他配置(如节点移动或能量管理等);

(3)启动仿真器;

(4)仿真结果分析(包括输出网络场景和数据图像等)。

4　NS3 的深入探究

4.1　NS3 的数据跟踪

NS3 有两类跟踪:一是用 Logging 系统直接将执行过程显示在命令行,有助于调试仿真脚本;另外常用 Tracing 系统将采集的数据直接存于一文件中,以便后期分析处理。

Logging 系统从低到高可分 7 个等级,高的包含低的消息。通过. cc 文件对程序添加记录,通过环境变量修改系统等级,随后终端运行。而 Tracing 系统则存储大量信息,它包含 3 个基本概念:跟踪源(Tracing Sources)、跟踪宿(Tracing Sink)以及两者的连接机制。该系统基于回调函数收集统计信息,当某跟踪源产生一新事件,可通过回调了解其内部正在发生的模拟情形及设备运行状况。

4.2　NS3 的 TCP 实现

4.2.1　TCP 类及其相互作用

TCP 类在网络模型中与 IPv4/6 协议一起驻留,实现彼此通信的多个类与网络层交互,并向应用层提供可靠的数据传送。下面列出 NS3 中关于 TCP 实现的主要类别以及它们之间的相互作用,如图 4 所示[13]。

TcpSocket:这个抽象类包含 TCP 套接字的基本属性。

TcpSocketBase:此类为应用程序层提供了关键的 TCP 特性和一个套接字接口。它从 TcpSocket 继承,用作所有 TCP 变体的基类。

Tcp Header:此类定义了 TCP 段的头。

TcpTxBuffer:此类提供一个缓冲区,发送方在发送和确认之前,缓冲从应用程序接收的所有数据。

TcpRxBuffer:该类实现一个缓冲区,用于接收从网络层收到的数据,然后将其传递给应用程序。

TcpL4Protocol:是 TCP 套接字和网络层接口类,负责向网络层收发数据包,并且负责传入数据校验和验证。

除了上述主类外,NS3 还包含了基于继承 TCP 套接字的子类,可以实现多种变体。如 Tcp Tahoe、Tcp Reno 和 Tcp NewReno 及 Tcp Westwood 等。

4.2.2　全局变量

现有的变体都是基于继承 TCP 套接字的同一类实现的,它包括以下各全局变量,用以实现拥塞控制算法及其带宽估计等。

m_cWnd 是一 uint32_t 型变量,用于表示拥塞窗口。发送方用来确定可进入网络而不致链路过载的字节数。当发生丢包时,使用 m_cWnd 来估计带宽。

m_ssThresh 也是 uint32_t 型变量,用于标记慢启动结束门限阈值,其值取决于丢包时的链路带宽估计。

m_initialcWnd 是指定 m_cWnd 初值的 uint32_t 变量。

m_inFastRec 是指示快速恢复开始和结束的布尔型变量。

m_prevAckNo 类型为 SequenceNumber32,用以保存最后接收到的 ACK 号。

m_accountedFor 为 uint32_t 型变量,其在丢包时可跟踪 DUP ACK 段数量,并在估计带宽时使用。

m_lastAck 是前一 ACK 到达时间的双精度浮点型变量。

m_currentBW 表示当前带宽估计的双精度浮点型变量。

m_minRtt 是指定的最小 RTT 的时间类型变量。

m_lastBW 是双精度浮点型变量,其在通过 Tustin 滤波器之后,保存最后所估计的带宽值。

m_lastSampleBW 是双精度浮点型变量,其保存测量带宽的最后一个样本值。

m_ackedSegments 是整型变量,它保存当前 RTT 期间已确认的段的总数。

m_IsCount 是一个布尔型变量,用于指示 m_acked 段计数过程的开始。

m_bwEstimateEvent 是指定带宽采样事件类型 EventId 的变量。

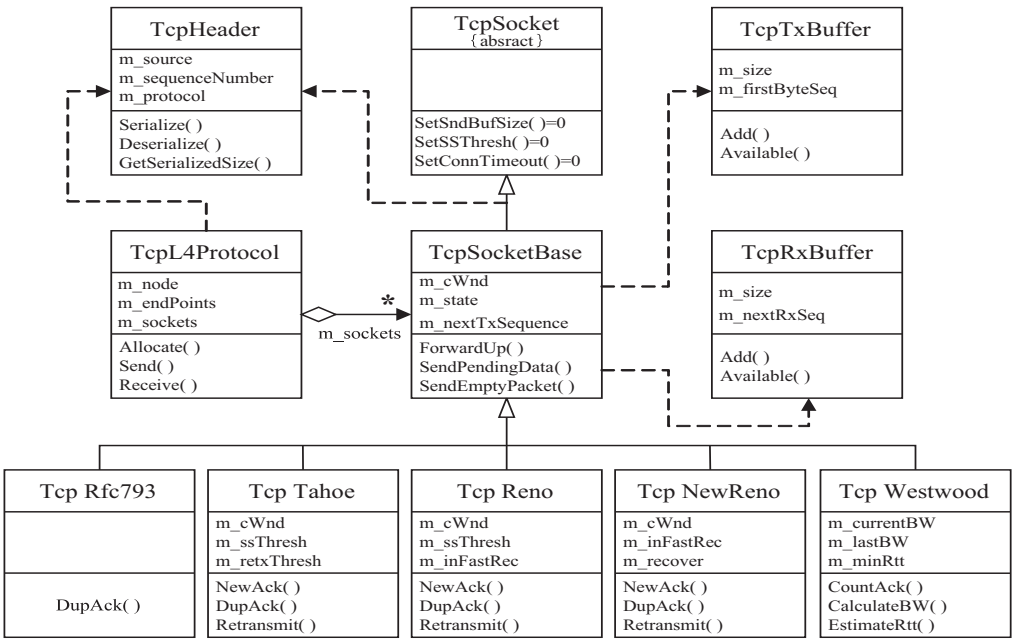

图 4 TCP 类图一览

4.2.3 算法实现

NS3 支持多种类型的 TCP 算法的实现,这些实现继承自 src/network 目录下的一些通用类,这样用户就能以最少的代价实现自己的算法。这里首先描述聚合的概念,在 NS2 中已广泛使用的继承和多态被用来扩展协议模型,如 RenoTcpAgent 就是通过重写方法继承自 TcpAgent 的,这促成了 NS3 的对象聚合系统。然后再介绍两个重要的抽象基类:class TcpSocket,这在 src/internet/model/tcp-socket.{cc,h}中定义。这个类主要用来管理能被不同算法重用的一些属性。例如,属性 InitialCwnd 可用于从 TcpSocket 类派生的任何实现。而 class TcpSocketFactory 则由协议的第 4 层实例使用,以创建正确的 TCP 套接字类型。

目前对于 NS3,有四种方式实现 TCP:NS3 自带的 TCP 实现;支持直接代码执行(DCE);支持网络模拟通讯座(NSC);虚拟机与 NS3 的组合。鉴于复杂程度,文中只介绍第一种实现方式。NS3 自带的 TCP 模型具有建立连接和关闭逻辑的双向 TCP 功能。支持多种拥塞控制算法,如 NewReno、Westwood、Hybla 和 HighSpeed 等,暂不支持多路 TCP 和 TCP SACK 算法。

以前的拥塞控制被认为是通过继承父类的独立的 TCP,每个拥塞控制(如 NewReno)是 TcpSocketBase 的子类,需修改一些继承的方法。但从 NS3.25 开始,TCP 模块经过了重新设计,其中拥塞避免、快速重传与恢复算法已修改,并完成了在 TcpSocketBase 内的合并。架构被重做,以避免之前的继承,让每个拥塞控制都具有一个单独的类,并且建立一个接口,用以在 TcpSocketBase 和拥塞模块之间交换数据,可以为创建和执行自动化测试创造更好的环境。一般在 src/applications/helper 和 src/network/helper 中定义帮助函数,通过 NS3 使用套接字,在应用层设置 TCP 的使用,具体步骤及方法如下:

(1)创建 TCP 接收器。

```
// Create a packet sink on the star "hub" to receive these packets
uint16_t port = 50000;
AddresssinkLocalAddress( InetSocketAddress
(Ipv4Address::GetAny (), port));
PacketSinkHelper sinkHelper("ns3::TcpSocketFactory", sinkLocalAddress);
ApplicationContainer sinkApp = sinkHelper. Install
(serverNode);
sinkApp. Start (Seconds (1.0));
sinkApp. Stop (Seconds (10.0));
```

类似的,以下代码片段将 OnOffApplication 流量源配置为使用 TCP:

```
//Create the OnOff applications to send TCP to the server
OnOffHelper clientHelper ("ns3::TcpSocketFactory", Address ());
```

这里已指定了抽象基类 TcpSocketFactory 的 TypeId。NS3 如何判断脚本需要的是它自带的 TCP 模型还是其他来源,当网络堆栈添加到节点时,聚合到该节点的默认 TCP 实现是 NS3 TCP。因此,默认情况下,在使用 NS3 helper API 时,聚合到具有 Internet 堆栈的节点的 TCP 是 NS3 TCP。

(2)配置 TCP 行为。

通过 NS3 属性,系统导出参数,记录在 TcpSocket 类的 Doxygen 中,例如最大段大小是可设置的属性。要在创建任何 Internet 堆栈相关对象之前设置默认套接字的类型,可在仿真程序的顶部放置以下语句:

```
Config::SetDefault ("ns3::TcpL4Protocol::SocketType", StringValue ("ns3::TcpNewReno"));
```

(3)绑定 Socket 套接字。

对于用户,希望有一个指向实际套接字的指针,需要用函数 Bind()来绑定套接字。设置选项等可在每个套接字的基础上完成,TCP 的套接字可通过使用 Socket::CreateSocket()的方法进行创建。传递给 CreateSocket()的 TypeId 必须是类型 ns3::SocketFactory。因此需要通过与底层 TcpL4Protocol 对象相关联的属性来完成底层套接字类型的配置,也可通过属性配置系统完成。在下面的示例中,访问节点容器“n_0n_1”以获取第零个元素,并在此节点上创建绑定套接字。

```
// Create and bind the socket...
TypeId tid = TypeId::LookupByName ("ns3::TcpNewReno");
Config::Set ("/NodeList/*/$ns3::TcpL4Protocol/SocketType", TypeIdValue (tid));
Ptr<Socket> localSocket = Socket::CreateSocket
(n0n1.Get(0), TcpSocketFactory::GetTypeId ());
```

上面有点号的“*”通配符被传递给属性配置系统,以后所有节点上的套接字都被设置为 NewReno,而不仅仅是节点“n_0n_1.Get(0)”了。如果想要将其限制为仅指定的节点,则必须执行以下操作:

```
// Create and bind the socket...
TypeId tid = TypeId::LookupByName
("ns3::TcpNewReno");
std::stringstream nodeId;
nodeId<<n0n1.Get(0)->GetId();
std::string specificNode = "/NodeList/" + nodeId.str() + "/$ns3::TcpL4Protocol/SocketType";
Config::Set (specificNode, TypeIdValue (tid));
Ptr<Socket> localSocket = Socket::CreateSocket
(n0n1.Get(0), TcpSocketFactory::GetTypeId ());
```

一旦创建了 TCP 套接字,就需要遵循传统的套接字逻辑以及 connect()和 send()(对于客户端)或 bind()、listen()和 accept()(对于服务器端)。有关在 NS3 中使用套接字的信息,可以参阅 Sockets API 相关文献[14]。

若要对已存在算法进行验证,通常 TCP 测试从一个名为 TcpGeneralTest 的类继承,该类提供了涉及 TCP 对象测试场景的操作设置,其位于 src/internet/test 目录下。有关编写新测试的更多信息,请参阅有关 TCP 测试文献。一些测试存于 src/test/ns3tcp 目录,其具有 Internet 模块之外的依赖关系。更多信息可在 NS3 官网的 Wiki 页面上找到。

5　结束语

NS3 具有更好的开发环境(包括 COM 类接口聚合与查询、自动内存管理及对象回调等核心功能),便于仿真全新的复杂模型。它克服了 NS2 的诸多缺陷和明显弱点,越来越多的网络仿真功能现已逐渐推行并植入到 NS3 中,使之适用于更多场景。文中结合项目开发经验,从不同角度和深度对新一代网络仿真器 NS3 作了较为全面的阐述与介绍,对其体系结构和主要功能模块进行了深入剖析,尤其对 NS3 的 TCP 实现机制作了重点探究。为 NS3 的使用及研发者提供了较好的理论指导意义和应用参考价值,为网络相关课题的后续研究及深入学习提供了强有力的支持。目前,NS3 的一些新功能正在积极开发之中,如节点多接口处理、IP 寻址以及更多的因特网协议设计,包括更详细的 802.11 模型等都非常值得期待,NS3 最终必将会取代 NS2。

参考文献:

[1]　梁军学,林昭文,马　严. 未来互联网试验平台[J]. 计算机学报,2013,36(7):1364-1374.

(下转第 82 页)

较高。后续可以根据标签数目变化动态地改变帧长，进一步缩短截断二进制指数退避时所消耗的时间。

此外，RFID 新的防碰撞算法可能将会深入到 RFID 网络物理层（频率、信号调制和数据加密）和数据链路层的特性。在无线协同网络层可以实现信息编码，由此可以更好地实现分集性能，有效提高信息的传输速率，同时具有较低的硬件损耗。而在数据链路层中可以将数据信息自适应组合，并调节发送速率使得与接收端相匹配，使得防碰撞算法结合计算机协议思想得到更好的改进，充分提高 RFID 系统的工作效率。

参考文献：

[1] ULLAH S，ALSALIH W，ALSEHAIM A，et al. A review of tags anti-collision and localization protocols in RFID newworks[J]. Journal of Medical Systems，2012，36(6)：4037-4050.

[2] 李 晶. 一种改进的 RFID 防碰撞时隙 ALOHA 算法[D]. 长春：吉林大学，2009.

[3] 王 勇，李 婷. 改进的基于 ALOHA 的 RFID 防碰撞算法[J]. 电信科学，2016，32(8)：77-81.

[4] DEMIRKOL I，ERSOY C，ALAGOZ F. MAC protocols for wireless sensor networks[J]. IEEE Communications Magazine，2006，44(4)：115-121.

[5] KLAIR D K，CHIN K W，RAAD R. A survey and tutorial of RFID anti-collision protocols[J]. IEEE Communication Surveys and Tutorials，2010，12(3)：400-421.

[6] GALLAGER R G. A perspective on multiaccess channels [J]. IEEE Transactions on Information Theory，1985，31(2)：124-142.

[7] 李宝山，乔 聪. 基于 P-坚持 CSMA 提高 RFID 系统吞吐率的改进算法[J]. 计算机测量与控制，2013，21(12)：3322-3324.

[8] 马 纯，尹小燕，房鼎益，等. 退避算法多负载状况下的退避窗口最优设定[J]. 计算机应用研究，2015，32(1)：175-178.

[9] CHEN Xiaoming，HONG Geok-Soon. A simuliation study of the predictive p-persistent CSMA protocol[C]//Proceedings of the 35th annual simulation symposium. Washington，DC，USA：IEEE Computer Society，2002.

[10] 蒋子峰，陆建德. IEEE802.15.4 动态自适应 CSMA/CA 算法设计与仿真[J]. 计算机技术与发展，2010，20(9)：69-73.

[11] 黄 仁，郜 辉，任军华. 非时隙 CSMA/CA 性能分析与研究[J]. 计算机工程与应用，2009，45(7)：108-110.

[12] 何 伟，南敬昌，潘 峰. 改进的动态 p-坚持 CSMA 协议[J]. 计算机工程，2010，36(21)：118-120.

[13] 苏恒阳，谭英丽. 改进的 RFID 动态帧时隙 ALOHA 算法[J]. 计算机仿真，2011，28(8)：148-152.

[14] 钱东昊，张 琨，张 磊. 基于标签识别码分组的防碰撞算法研究[J]. 计算机应用与软件，2015，32(7)：252-254.

[15] 高金辉，郑晓彦. 新型的 RFID 混合防碰撞算法[J]. 电子技术应用，2011，37(12)：130-132.

[16] SCHOUTE F C. Dynamic frame length ALOHA[J]. IEEE Transactions on Communication，1983，31：565-568.

[17] CHA J Y，KIM J Y. Novel anti-collision algorithms for fast object identification in RFID system[C]//Proceedings of the 11th international conference on parallel and distributed systems. Washington，DC，USA：IEEE Computer Society，2005：604-609.

（上接第 77 页）

[2] 袁 晓，蔡志平，刘书昊，等. 大规模网络仿真软件及其仿真技术[J]. 计算机技术与发展，2014，24(7)：9-12.

[3] 马浩然. 基于 NS3 的分布式消息系统 Kafka 的仿真实现[J]. 软件，2015，36(1)：94-99.

[4] FONT J L，IÑIGO P，DOMÍNGUEZ M，et al. Analysis of source code metrics from ns-2 and ns-3 network simulators [J]. Simulation Modelling Practice & Theory，2011，19(5)：1330-1346.

[5] RILEY G F，HENDERSON T R. Modeling & tools for network simulation[M]. Berlin：Springer，2010.

[6] DAS B，SUBUDHI B，PATI B B. Cooperative formation control of autonomous underwater vehicles：an overview[J]. International Journal of Automation and Computing，2016，13(3)：199-225.

[7] RACHNA C，SHWETA S，RITA K，et al. A study of comparison of network simulator-3 and network simulator-2 [J]. International Journal of Computer Science and Information Technologies，2012，3(1)：3085-3092.

[8] 闵圣天，曾文序，李满荣，等. 基于 NS3 的网络协议分析与模拟[J]. 福建电脑，2014，30(2)：99-100.

[9] 张登银，张保峰. 新型网络模拟器 NS-3 研究[J]. 计算机技术与发展，2009，19(11)：80-84.

[10] 栾 俊，李太浩. 基于 NS-3 的 WiFi 场景仿真[J]. 农业网络信息，2012(1)：18-20.

[11] 李广荣. 基于 NS-3 的虚实网络结合系统的设计与实现[D]. 哈尔滨：哈尔滨工业大学，2015.

[12] 杨鸣亮，张嘉毅，孙 振，等. 关于 NS3 中 GRE 仿真的研究[J]. 电子测量技术，2011，34(1)：22-26.

[13] GANGADHAR S，NGUYEN T A N，UMAPATHI G，et al. TCP Westwood(+) protocol implementation in ns-3[C]// Proceedings of the 6th international ICST conference on simulation tools and techniques. Brussels，Belgium：ICST，2013：167-175.

[14] 袁鹏飞，郑 涛，杨李冬，等. 一种基于 CAPPROBE 带宽估计的 TCP Westwood 算法[J]. 厦门大学学报：自然科学版，2014，53(4)：469-476.