高技术通讯 2010 年 第 20 卷 第 4 期:355 ~ 360

doi:10.3772/j.issn.1002-0470.2010.04.005

基于网络处理器的自相似流量仿真系统①

李新磊②* ** *** 郑康锋* ** *** 杨义先③* ** ***

(* 北京邮电大学网络与交换技术国家重点实验室信息安全中心 北京 100876)
(** 北京邮电大学网络与信息攻防技术教育部重点实验室 北京 100876)
(*** 灾备技术国家工程实验室 北京 100876)

摘 要 为了解决大部分网络流量的自相似性研究中缺少高速真实的数据流的问题,提出了一个使用网络处理器 IXP2400 建立的自相似流量仿真系统。该系统利用分形布朗运动(FBM)模型产生自相似序列,以该序列作为数据发送带宽的依据;使用基于微引擎的发送控制算法实现高精度数据发送,以减少输出数据流赫斯特参数的误差,并利用 IXP2400 的硬件资源缩短随机中点置位(RMD)算法的计算时间。实验结果表明,该系统能够根据输入的赫斯特参数产生真实的自相似 TCP 数据流,最高数据发送速度接近 900Mb/s,赫斯特参数的误差控制在 5% 以内。

关键词 自相似,网络处理器,微引擎,Hurst 参数

0 引 言

僵尸网络的出现使得攻击流量不仅在微观特征上无法与合法的用户流量区分,而且宏观上也呈现出明显的自相似性,这使得攻击流量很难与正常的业务流量区分[1,2]。因此,在进行流量型网络攻击(如拒绝服务攻击)的研究时需要一个可控的具有明显自相似性的背景流量。部署一个与真实网络相同的实验网络显然是不现实的,常见的解决方案是从实际网络的测量中提取攻击序列,但是实际流量的大小和自相似特性等难以控制,所以有必要实现人工合成的自相似数据流以满足相关研究及验证工作的需要。目前对网络流量自相似性的研究大部分使用软件仿真的方法进行,如文献[3]中提出使用 OPNET 软件平台合成自相似业务流并研究其特性,这种方法无法产生真实流量,是一种纯虚拟化的研究,缺乏实际环境中的验证。而产生高速真实流量的方法通常是使用大型网络测试设备,如 SmartBit、IpiE 等。这些设备所产生的流量大多类型单一,缺少对完整传输控制协议(TCP)会话的支持,另外这类设备专业化强、系统设置复杂、不易使用,而且价格昂贵。目前在半实物计算机网络仿真领域使用硬件平台产生真实网络流量的方案很少,本文是第一次将网络处理器引入到这一领域,利用网络处理器的高性能处理能力实现自相似流量的仿真,并且具备基本的 TCP 交互能力。与大型测试设备相比,该方案成本较低,配置简单,控制灵活。

1 相关背景

自相似模型比传统模型更能充分地表现网络流量的特性。常用的具有自相似特性的流量发生模型有分形布朗运行(fractional Brownian motion, FBM)模型[4]、分形高斯噪声(fractional Gaussian noise, FGN)模型[5]、分形自回归滑动平均(fractional autoregressive integrated moving-average, F-ARIMA)过程模型[6]以及离散小波模型[7]等。产生自相似流量的方法很多,如基于快速傅立叶变换(FFT)的方法、基于小波变换的方法等。但是大部分自相似模型的算法需要大量计算,不适合在网络处理器上实现。本文结合网络处理器的特点选取了基于 FBM 模型的较为快速简单的随机中点置位(random midpoint displacement, RMD)方法来产生自相似数据流序列,其优点在于占用处理器时间相对较少且实现相对简单。

① 863 计划(2007AA01Z466),973 计划(2007CB310704),国家自然科学基金(60821001)和国家自然科学基金与香港研究资助局联合科研基金(60731160626)资助项目。
② 男,1981 年生,博士生;研究方向:网络安全;E-mail:lixinlei.bupt@gmail.com
③ 通讯作者,E-mail:yxyang@bupt.edu.cn
(收稿日期:2008-12-18)

系统采用 IXP2400 网络处理器，这是 Intel 公司为高速网络数据包处理而设计的一种专用处理器。IXP2400 采用多核、多线程的模式来提高处理能力，其核心部件为微引擎（micro-engine，ME），另外使用一颗 Xscale 处理器负责网络处理器控制层面的处理任务和整个网络处理器的管理任务。

2 软件结构设计

本文使用 Radisys 公司的 ENP-2611 板卡作为验证系统的硬件平台。由于网络处理器由 Xscale 和 ME 两类处理器组成，因此相应的软件结构分为控制层和仿真引擎层两层组成。

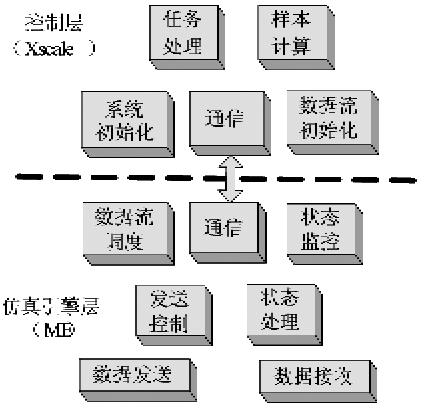

图 1 软件结构示意图

（1）控制层：运行于 Xscale 之上，主要的任务是进行 FBM 序列的计算，另外还要执行外部任务指令的处理以及系统管理等任务。

（2）仿真引擎层：运行于 ME 之上，主要任务是根据 FBM 序列进行带宽分配，精确发送数据包，维护网络会话等。

仿真引擎层上采用的是微码编程，微码是 IXP2400 特有的开发语言，通过编程的方法实现硬件接口的数据接收、发送和处理，能够使性能接近硬件的理论指标。通过编写不同的微码程序可以实现不同的功能，升级微码即可快速实现硬件的升级。

2.1 RMD 算法的加速与设计

文献[8]和[9]阐述了 RMD 算法生成 FBM 序列的原理，并提出了实现 FBM 的快速方法。基本思想是递推的扩展生成序列，通过不断的分割来产生样本值，每次分割使用一个高斯偏移量修正中点的样本值，使最后得到的数据具有很好的边缘分布特性。

Xscale 的主要任务是运行 RMD 算法计算 FBM 序列，虽然 RMD 算法计算量相对较小，但考虑到 Xscale 的主频只有 600M，所以仍需提高计算效率。IXP2400 内置有硬件伪随机数发生器，可以快速产生 32 位的伪随机数，利用伪随机数发生器产生 RMD 算法所需的高斯分布随机数序列可以提高计算速度。需要指出的是计算生成的 FBM 序列可能会有负值出现，需要进行线性变换使序列具备物理意义。基于 IXP2400 的 RMD 算法设计如下：

步骤 1：由硬件生成 n 个伪随机数序列 X_i，除以序列周期 $2^{32}-1$，生成[0,1]上的均匀分布随机数序列 Y_i。

步骤 2：选取 Y_i 中每 k（实验表明 $k \geqslant 12$ 即可取得理想效果）个样本值为一组，利用中心极限定理产生均值为 0，方差为 1 的高斯分布随机数序列 G_j。

步骤 3：迭代分割产生 FBM 序列 Z_i。

步骤 4：对 Z_i 进行变换：$Z[i] = M + a(Z[i] - m)/\sigma$，使 Z_i 具有物理意义，其中 M 表示平均速率，a 为峰值速率的控制系数，m 为 FBM 序列的最小值，σ 表示 FBM 序列最大值与最小值之差。

需要指出的是步骤 4 进行的变换为线性变换，不会改变序列的自相似特性，所以变换后的序列与原序列具有相同的 Hurst 参数。

2.2 仿真引擎的设计及其编程模型

仿真引擎的设计是整个软件设计中的核心部分，其主要任务有两方面：一是提供高精度的数据发送（发送精度与仿真数据流赫斯特参数的精度成正比）；二是进行网络层的连接处理。IXP2400 中有 8 个 ME，分为两组：Cluster0 和 Cluster1。具体使用情况如下：

表 1 微引擎使用分配表

微引擎编号	分配任务
ME0:0	数据流调度
ME0:1	数据发送控制
ME1:0、ME1:1	千兆端口发送
ME1:2、ME1:3	千兆端口接收
ME0:3	通信
ME0:2	状态处理

仿真引擎工作流程如图 2 所示。

数据流调度 ME 是仿真引擎中重要的部分之一，其主要工作是根据自相似序列的值在带宽范围内发起尽可能多的连接。具体实现时在指定的源 IP 地址范围和源端口范围内以一定的规律发起 TCP 连接，可以按照地址和端口顺序进行也可以随机选取地址和端口。

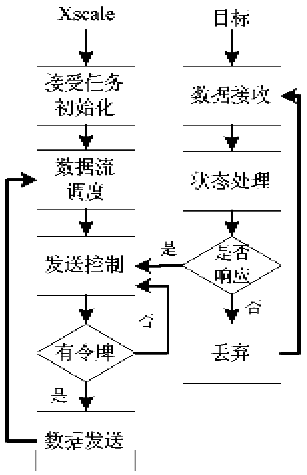

图 2　仿真引擎工作流程图

发送控制 ME 是仿真引擎中的另一个重要部分,控制精度直接影响输出仿真流的赫斯特参数的精度。后面一小节将专门讨论发送控制 ME 的控制算法。

状态处理 ME 维护一个连接状态表,根据接收到的数据包进行标准的 TCP 三次握手处理,将构造好的响应包交给发送控制 ME。

因为在自相似流量仿真应用中数据包的处理过程比较固定,因此 ME 的编程模型采用一种叫做超级任务链(hyper task chaining,HTC)的模型[10]。HTC 实际上是把一连串的处理按流水线方式排列,数据包依次经过所有处理流程,处理的复杂度以及处理时间基本相同。每个 ME 实现一个处理子任务。在本系统中 ME 的编程模型如图 3 所示。

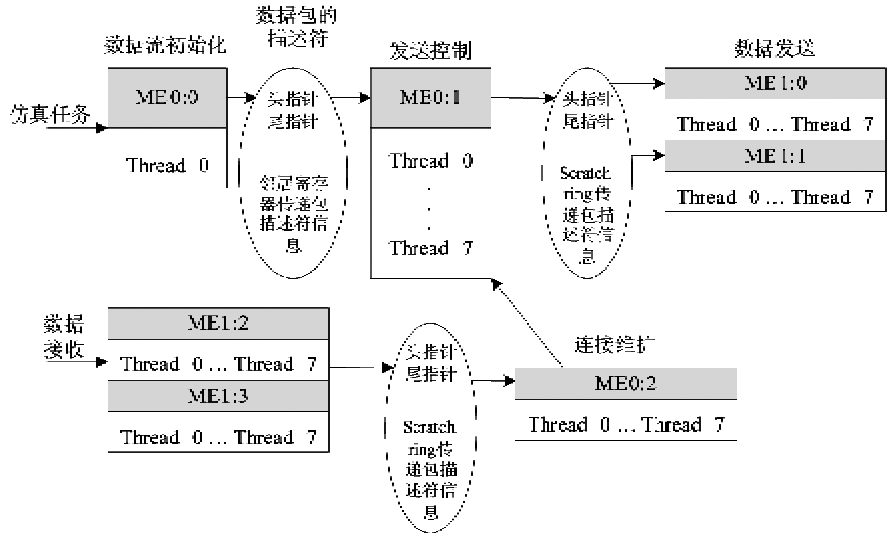

图 3　ME 编程模型示意图

数据流调度 ME 和接收 ME 为每一个需要处理的数据包创建一个数据包描述符,描述符携带各种需要的信息取代数据包本体在不同子任务间进行传递,以此减小系统开销。描述符的传递采用硬件队列方式。IXP2400 的硬件队列分两种。一种硬件队列是使用网络处理器内部存储资源 Scratch 或 SDRAM 建立的,队列的维护使用硬件机制,保证了大吞吐量的队列操作;另一种硬件队列是采用叫做邻居寄存器的寄存器建立的,邻居寄存器可以由本地 ME 或者邻居 ME 访问,特点就是微引擎间传递信息的速度快。在本系统中连接维护和发送控制两个子任务间对延迟的要求最高,因此采用第二种队列方式,其他子任务间采用第一种队列方式。

2.3　基于微引擎的发送控制算法

仿真引擎层是整个软件体系中的核心部分,该层的数据流发送精度是输出数据流的 Hurst 参数精度的一个重要影响因子。为了保证发送数据流的精度,提出了基于 ME 的发送控制算法。对于 FBM 序列的使用,在产生真实数据流时可以有两种用法:一是当作相邻两个数据包之间的间隔;二是作为单位时间内发送的包(或字节)数。本文采用第二种用法,即用 $Z[i]$ 代表第 i 个单位突发时间 Δt 内的发送速率(bps)。算法的基本思想如下:

发送控制微引擎 ME0:1 以一定的间隔时间向发送微引擎 ME1:0 和 ME1:1 发放令牌,时间间隔以线程的延迟时间为参考,发送微引擎依据令牌进行数据发送,以此控制发送微引擎实现高精度的数据

发送。设数据包的长度为 L（字节），IXP2400 发送一个长度为 L 的数据包所用的固有时间为 t_1（包括数据从 DRAM 移到发送缓存以及物理层设备（PHY）发送数据帧占用的时间等），相邻数据包的发送延迟为 t_2，则有如下关系：

$$\frac{\Delta t \cdot Z[i]}{8 \cdot L} = \frac{\Delta t}{(t_1 + t_2)} \tag{1}$$

变换可得

$$t_2 = \frac{8 \cdot L}{Z[i]} - t_1 \tag{2}$$

设 t_3 为 ME0:1 调度时进行一次最小延迟的时间，$n[i]$ 为产生两个相邻令牌时需要进行最小延迟的次数，则有 $t_2 = n[i] \cdot t_3$，t_3 是固定值。对于 t_3 的选择有两种：一是使用 IXP2400 的“Timestamp”寄存器进行计数，每次计数为 16 个指令周期；二是使用通用寄存器自行设定一个计数器，精度可以达到 1 个指令周期，但是需要浪费一个微引擎专门维护这个计数器。为减小系统开销，本文采“Timestamp”进行延迟计数。$n[i]$ 即为在第 i 个突发时间 Δt 内微引擎 ME0:1 产生相邻两个令牌的间隔，即发送微引擎 ME1:0 和 ME1:1 每发送一个包需要延迟的参数。因此有

$$n[i] = \frac{8 \cdot L}{t_3 \cdot Z[i]} - \frac{t_1}{t_3} \tag{3}$$

需要指出的是 Δt 的选取不能太小，和 $t_1 + t_2$ 可比拟时就会影响发送速度的精度。

从式(3)可以看出，在每个单位突发时间 Δt 内 $n[i]$ 是可以提前计算好的，但是 $n[i]$ 在整个 Δt 内并不是不变的，根据精度需要在更微小的时间周期内进行微调，以达到更高的精度。微调的原理如下：

在 Δt 开始的时刻开始记录 PHY 实际发送的比特数，在 $\Delta t/2$ 时刻比对实际已发送比特数 N 和理论发送值 $(\Delta t/2) \cdot Z[i] \cdot 8$，计算差值是否达到需要微调的门限 $t_3 \cdot Z[i] \cdot 8$，如果达到则进一步计算需要调整的延迟间隔 $\Delta n[i]$，计算公式如下：

$$\Delta n[i] = \left| N - \frac{\Delta t}{2} \cdot Z[i] \cdot 8 \right| \bmod(t_3 \cdot Z[i] \cdot 8) \tag{4}$$

微调时仅对 $\Delta t/2$ 后的下一个延迟间隔做调整，若实际发送量大于理论发送量则该时间间隔调整为 $n[i] - \Delta n[i]$，反之调整为 $n[i] + \Delta n[i]$。在 Δt 选取合理的情况下 $\Delta n[i]$ 一般无法和 $n[i]$ 比拟，因此 $n[i] - \Delta n[i]$ 出现负值的可能性很小，但是一旦微调时发现 $n[i] - \Delta n[i]$ 确实为负值则需检测是否出现硬件故障或线程阻塞。

3 实验与结果

3.1 最高数据发送能力试验

为测试在完整处理流程下的数据发送能力，根据 RFC 标准[11] 选取数据帧长为 64、128、256、512、1024、1280、1518 字节的数据包测试系统的发送能力，测试设备为 SmartBits Chassis SMB-6000。SMB-6000 与 ENP-2611 之间采用千兆光纤连接，测试结果如图 4 所示。

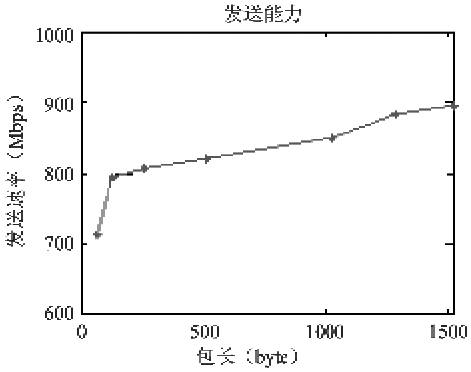

图 4 发送能力测试图

从图 4 可以看出，系统具备了较强的数据处理及发送能力，能够满足进行自相似数据流实物仿真的需求，特别是可以用来仿真流量型网络攻击的特性。图中表现出的长数据包比短数据包发送速率高的现象的主要原因是，在短数据包的处理过程中内部操作较为频繁，因此系统开销比较大。

3.2 基于 ME 的发送控制算法仿真实验

本节对基于 ME 的发送控制算法进行仿真实验，验证算法对发送数据速率的控制精度。仿真使用 Intel 提供的 SDK3.5 工具包，实验平台为 IBM X3250 服务器（配置为 2.8GHz 双核处理器，2GB DDR 内存，Win2000 Server 操作系统），测试数据帧的长度选取典型的 64、256、512、1518 字节 4 种长度，设定发送的单位突发时间为 10s，仿真数据如图 5 所示。

从仿真数据可以看出，在各种不同负载特别是大负载情况下，系统输出的仿真自相似数据流和理论值达到了较高的吻合度，发送精度比较理想，说明发送控制算法具备较高的有效性。但是也可以发现理论值和仿真结果存在一定的误差，造成误差的主要原因一是 SDK 所带仿真工具对输出数据流的捕获不够精确，例如对输出流的采样时间跨越两个相邻的 Δt 时便会造成较大误差；二是发送控制算法本

身的精度还有待提高。

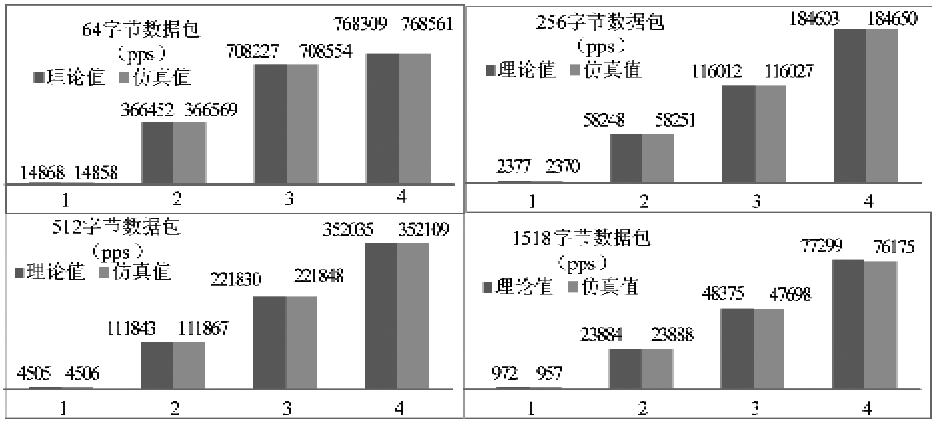

图 5　不同包长和速率的发送控制仿真结果

3.3　数据流仿真实验

本节选取发送控制精度相对较高的 512byte 字节数据包,设定 Δt 为 10s,Hurst 参数分别为 0.65、0.75、0.85、0.95 时用 R/S 法对输出数据流的 Hurst 参数进行验证,结果如表 2 所示。

表 2　Hurst 参数仿真值与设定值对照表

理论 H 参数	仿真结果	误差(%)
0.65	0.6317	2.82
0.75	0.7213	3.83
0.85	0.8199	4.48
0.95	0.9312	2.02

从上面的仿真结果可以看出,输出仿真数据流能够良好地表现设定的输入参数,说明基于网络处理器的自相似数据流模型能够有效产生预期的自相似数据流,数据流的 Hurst 参数能够根据需要进行改变,且与设定值比较吻合。但是输出流的 Hurst 参数还是存在一定误差,误差范围在 5% 以内,下面一节将讨论误差产生的原因。

3.4　仿真结果 Hurst 参数的误差分析

从 3.3 小节的仿真结果看,仿真数据流的 Hurst 参数与设定值存在一定的误差,误差的来源主要有以下 3 方面:

(1) RMD 算法产生的误差。RMD 算法是近似生成 FBM 序列的算法,快速计算的代价是 Hurst 参数误差较大。减小这类误差只能改进 RMD 算法或者采用其他快速、准确的算法。

(2) 发送控制算法产生的误差。发送控制算法的精确度对自相似数据流产生直接影响,但是在整个流量的生成过程中产生的影响是相同的。如果期望 Hurst 参数值为 H_1,可以引入一个修正因子 H',对 RMD 算法的输入 Hurst 参数值 H_2 做修正:$H_2 = H' + H_1$,以此减小对输出数据流的 Hurst 参数的影响,这里 H' 是一个经验型的参数,需要通过大量的实验取得。

(3) 数据测量产生的误差。该类误差对数据流的自相似特性不产生实质影响,是测量方法不当产生的主观偏差。通过更加精确的观测手段和合理的抽样取值可以减小该类误差,例如采用基于硬件的专业测试设备等。

4　结　论

本文的主要工作是提出了使用网络处理器产生以太网自相似数据流的构想,并进行了初步探讨。利用网络处理器较强的数据包底层处理能力实现真实可控的自相似数据流,为相关研究提供数据源。对使用 RMD 算法近似产生 FBM 序列进行了仿真,提出了网络处理器内部数据发送的精确控制算法,并验证了网络处理器精确发送数据流的能力。

然而利用网络处理器进行自相似流量的仿真还只是初步探讨,后面还有大量工作要做,需要改进的方面主要集中在下面三点:一是提高发送控制算法的控制精度,可考虑使用时间精度为单个指令周期的计数器进行控制,但是要解决好减小系统开销和多线程调度的问题;二是实现混合包长的数据流发送,包长为 64 到 1518 字节之间的任意值,需要相应的改进发送控制算法;最后,现有的 RMD 算法在各种自相似模型中已经是计算速度比较快的了,但是

在主频 600MHz 的 Xscale 上运行还是需要耗费相当的时间,如何结合网络处理器更好地改进自相似序列的产生方法,也是需要解决的难题。

参考文献

[1] 韩心慧,郭晋鹏,周勇林等. 僵尸网络活动调查分析. 通信学报,2007,28(12):167-172

[2] 诸葛建伟,韩心慧,周勇林等. 僵尸网络研究. 软件学报,2008,19(3):702-715

[3] Fras M, Mohorko J. Estimating the parameters of measured self similar traffic for modeling In OPNET. In: Proceedings of the 14th International Conference on Systems Signals and Image Processing, and 6th EURASIP Conference Focused on Speech and Image Processing, Multimedia Communications and Services, Maribor, Slovenia, 2007, 78-81

[4] Duncan T E. Some martingales from a fractional brownian motion and applications. In: Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, Seville, Spain, 2005, 1110-1114

[5] Purczynski J, Wlodarski P. On fast generation of fractional Gaussian noise. *Comput Stat Data Anal*, 2006, 50(10): 2537-2551

[6] Liu J K, Shu Y T. Traffic modeling based on FARIMA models. In: Proceedings of 1999 IEEE Canadian Conference on Electrical and Computer Engineering, Edmonton, Alberta, Canadian, 1999, 162-167

[7] Li Y L, Liu G Z, Li H L, et al. Wavelet-based analysis of Hurst parameter estimation for self-similar traffic. In: Proceedings of 2002 IEEE International Conference on Acoustic, Speech and Signal Processing, Orlando, FL, USA, 2002, 2, 2061-2064

[8] Lau W C, Erramilli A, Wang J L, et al. Self-similar traffic generation: the random midpoint displacement algorithm and its properties. In: Proceedings of the 1995 IEEE International Conference on Communications, Seattle, WA, USA, 1995, 466-472

[9] Droz P, Le Boudec J Y. A high-speed self-similar ATM VBR traffic generator. In: Proceedings of the 1996 IEEE Communications Theory Mini-Conference, London, UK, 1996, 586-590

[10] Intel Corporation. Intel Internet Exchange Architecture Portability Framework. Santa Clara: Intel Press, 2003, 11, 23-24

[11] Bradner S, McQuaid J. RFC 2544: Benchmarking Methodology for Network Interconnect Devices. Fremont, California: The Internet Engineering Task Force, 1999

A self-similar flow simulation system based on network processor

Li Xinlei[*,**,***], Zheng Kangfeng[*,**,***], Yang Yixian[*,**,***]

([*] Information Security Center, State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876)

([**] Key Laboratory of Network and Information Attack & Defense Technology of MOE, Beijing University of Posts and Telecommunications, Beijing 100876)

([***] National Engineering Laboratory for Disaster Backup and Recovery, Beijing 100876)

Abstract

In order to solve the problem that most studies of self-similarity of network traffic lack the high-speed and real data flow, a self-similar flow simulation system based on the network processor IXP2400 is presented. The presented system makes use of the fractional Brownian motion (FBM) model to generate the self-similar sequence which is the reference for transmission bandwidth. The transmission control algorithm based on micro-engines achieves the high precision control of data flow so as to reduce the error of the output traffic's Hurst parameter. In addition, the system uses the hardware resources on the network processor to accelerate the efficiency of the random midpoint displacement (RMD) algorithm. The results show that the presented system can generate ideal self-similar TCP traffic according to the input Hurst parameter. The highest speed approaches to 900Mb/s and the error of the Hurst parameter is controlled in the range of 5%.

Key words: self-similar, network processor, micro-engine, Hurst parameter