主动网流调度模型A-Flow的研究

黄诚武，王 晟，李乐民
(电子科技大学通信与信息工程学院 成都 610054)

【摘要】分析了第一代主动网在数据包处理中存在的不足，引入新的主动节点数据流调度模型A-Flow，给出了该模型在主动节点传输系统ANTS中的实现方案。新模型基于主动网封装协议ANEP，通过分解调度输入主动节点的数据流，可以实现不同的执行环境在同一节点共存。该模型拥有数据包缓存、认证及净荷调度等功能，提高了主动节点的性能和安全性。

关 键 词 主动网； 流调度； 性能； 互操作性； A-Flow
中图分类号 TP393.03 文献标识码 A

A Flow-Dispatching Model on Active Networks

Huang Cheng-wu，Wang Sheng，LI Le-min
(School of Communication and Information Engineering, UEST of China Chengdu 610054)

Abstract This paper analyses the deficiency of packet handling in active networks, and presents A Flow-dispatching model named A-Flow on active networks. In this paper a project of ANTS over A-Flow is implemented. The new model is based on ANEP, it can make multiple execution environments interoperate on the same node, and it has functions of packet buffering, identification and payload dispatching, which may improve the performance and security of active nodes.

Key words active network; flow dispatching; performance; interoperation; A-Flow

传统网络是基于TCP/IP协议的分组交换网络。在传统网络中，数据包(IP包)内只封装了数据，路由器只是被动地根据路由表处理经过的IP包，这是简单的“存储－转发”模式。模式中，中间节点即路由器对于端用户是完全透明的。由于它的简单实用，其体系结构得到了广泛应用。但是随着网络技术的发展，其沙漏型的体系结构使得对网络的维护和新业务的配置越来越困难。

DARPA组织于1994～1995年在讨论关于未来网络的发展方向时提出了主动网(Active Network)概念[1]。它的主要思想是将网络资源抽象为网络应用编程接口，网络节点能够对流经的用户数据进行复杂的计算处理。在主动网中，原来的路由器或交换机被主动节点(Active Node)所取代。主动节点不再被动地转发它所接收到的数据包，它提供了面向用户的可编程接口，用户可以通过编程指定节点对数据的处理，即“存储－计算－转发”模式。主动网使网络的处理能力得到动态扩展，为解决传统网络中存在的问题提供了新的思路和方法。

在已经开发的主动网项目中，ANTS、SwitchWare、PAN、Smart Packets等可以称为第一代主动网研究项目。其基本上能够做到保护主动节点，但在保护整个网络资源上并不成功。在这方面PAN、ANTS、SwitchWare都试图解决，但没有取得很好的效果。此外，由于不同的主动网平台分别侧重于不同的应用，如ANTS用于协议配置，Smart Packets用于网络管理，因此在实现新的数据流调度策略时，让不同的主动网执

收稿日期：2004－08－30
基金项目：信息产业部“十五”预研基金资助项目
作者简介：黄诚武(1977－)，男，硕士生，主要从事通信网与宽带通信技术方面的研究.

行环境(Execution Environment, EE)在同一主动节点共存也是很有必要的。这些都是在开发第二代主动网时必须考虑的问题。

1 流调度模型A-Flow的设计

1.1 主动网中传统的数据包调度策略及其改善

在主动网中，每个节点都会处理带有程序的数据包。对输入数据包的处理，由于以下两个原因不利于资源管理：1) 当收到数据包时，处理中断所需的时间不能强加给任何应用，因为数据包的目的地址是未知的，需要检查数据包的内容来确定；2) 接收数据包所需的内存不能立即计算、分配。

对这类问题传统的解决方法是，尽量简化对无效数据包的处理。为此，数据包的处理过程分成两个步骤：包分类和后继协议处理。包分类主要是决定数据包的接收者，其思想主要体现在协议分层上：协议栈的某一层使用本层头部的一个域来决定数据包送往上层的哪个协议。后继协议处理则侧重于对接受者的资源控制，网络包的延迟处理技术LRP就是为了解决太多时间花在那些最终需要丢弃的数据包的处理上而导致的网络吞吐率严重下降的问题(尤其在网络过载的情况下)[2]。

LRP是在内核空间用C语言实现的，且仅限于对传统IP包的处理。为了同时调度主动包及传统IP包，且从移动代码的安全性考虑，本文提出一个架构于传统网络技术之上采用Java语言在用户模式实现的A-Flow模型，它为每个主动数据流分配一个流队列，与上层EE相关的主动包将直接递交给上层进行计算处理，其他主动包则交给默认流处理，这样可以大大减少对无效数据包的处理时间。

1.2 A-Flow主要模块设计

在A-Flow模型中如图1所示，流管理器(Flow Manager)采用对数据流注册的方法，可以屏蔽底层网络传输技术对上层主动计算功能的影响；流处理器(Flow Processor)实现对数据包的缓存、认证及净荷调度等核心功能。其他辅助类包括流缓冲队列(Flow Buffer)用于数据包缓存，流调度接口(Flow Dispatcher)声明调度函数原型，流调度器(ANEPDispatcher)则负责将数据包调往正确的流。以下是各主要模块的实现：

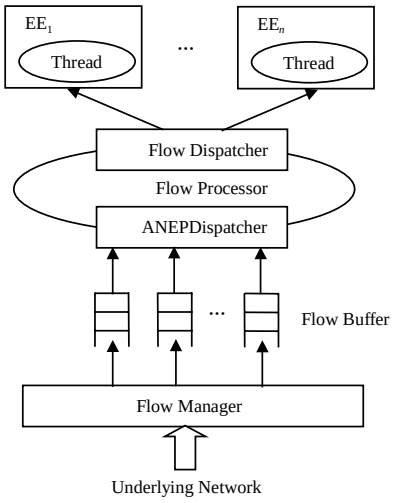


图1 流调度模型A-Flow

1) 流调度接口

首先，需要定义Dispatcher接口以调度输入数据包，该接口声明了一个抽象函数dispatchBuffer()。一个Dispatcher或者缓存一个数据包，或者指向另一个缓存该数据包的Dispatcher。

2) 流调度器

一个实现Dispatcher接口的类ANEPDispatcher可以完成对输入数据流的识别与分发。其算法比较简单：

首先查看Buffer中的主动网标识符(ANid)。

如果该ANid已经在某个EE的Flow Dispatcher中注册，则将此Buffer发往该Flow。

如果一个ANid没有注册并且未设置Discard位(其值由ANEP数据包头部的Flags域标识)，则调度此Buffer到默认Flow。

否则，丢弃该Buffer。

3) 流处理器

这是A-Flow模型的核心类，它有三个功能：(1) 数据包缓存。每个流中都有两个队列：一个是空闲队列(freeBuffers)，另一个是已用队列(usedBuffers)。后者用来存放已经接收但尚未投递的数据包。除非有空闲队列可用，否则不能缓存数据包。使用者必须保证有足够的缓存可使用。(2) 净荷调度。一个Processor和另一个Dispatcher相关联的。因此，分发到某个Processor的所有数据包的净荷也同样要分发到跟它相关联的Dispatcher处，从而实现协议分层。调度之前，Processor会将自己的头长度加到调度的偏移量上。(3) 数据包认证。Processor必须能够识别某个数据包是否属于某个流。这可以通过分析它所携带的数据来完成。

Processor是通过下面的过程来完成对输入流的调度工作的。首先，Processor检查它的freeBuffers队列是

否为空。若为空，则说明这个Flow的Buffer空间已经用完，即未给予Processor以足够的内存。这时，直接丢弃该数据包即可，无需进行额外的处理。若不为空，则将Buffer从队列中取出，并与调度路径中的系统Buffer“交换”。这样，包含数据的Buffer就被插入Processor的接收队列中。

4) 流管理器

该类提供了一套方法以在系统中注册Flow Processor。它提供了直接访问设备的方法，或访问现有的协议栈，比如IPv4/UDP协议。下面是一个注册UDP流的例子：

Initialization:

```
private Hashtable udpFlows = new Hashtable();
```

Register Flow module : registerUDPFlow(int port, FlowDispatcher fd)

```
Integer udpPort = new Integer(port);
UDPFlow udpFlow = (UDPFlow)udpFlows.get(udpPort);
if (udpFlow == NULL) then
    udpFlow = new UDPFlow(port, fd);
 udpFlows.put(udpPort, udpFlow);
```

5) 流缓冲队列

拷贝数据包是Dispatcher不能完成的操作之一。因为一个数据包有可能被传给多个Flow Processor进行处理，所以要求有它自己的可写的数据备份。第一个接收包的Processor将它的freeBuffers跟收到数据包的系统Buffer进行交换，这个Buffer中带有实际的数据，称为原始Buffer(originalBuffer)。其余的Processor也会进行同样的操作，但得到的Buffer中不包含实际数据，称之为从属Buffer(dependentBuffer)。originalBuffer与dependentBuffer之间通过链表连接起来。

当在一个Buffer上进行写操作或匹配操作时，必须先将它声明为“independent”，指的是：将它从链表中移出，必要的话还需拷贝数据。分两种情况：1) 对于dependentBuffer，需先将原始数据拷贝到它的内存中，然后才能将该Buffer从链表中删除；2) 对于originalBuffer，则需先拷贝数据到第一个dependentBuffer，并将它声明为余下Buffer的originalBuffer。将Buffer设置为相关链表的目的是，当一个Buffer需要发往多个目的地时，可以避免进行大量实际的数据拷贝。

2　A-Flow在ANTS中的应用及仿真实验

2.1　基于A-Flow的ANTS模型

为了验证A-Flow的的功能，本文采用应用最为广泛的主动网系统ANTS进行实验，它是基于Linux操作系统，采用Java语言实现的[3]。图2为修改后的ANTS的数据流动路径，其中最大的变动是增加了A-Flow调度，其他阴影模块也作了较大改动。在原来的ANTS中，Channel负责收发Capsule，Channel基类对数据封装与解封，子类实现具体的发送与接收方法；ChannelThread则负责对某个Channel循环接收、解码和执行数据包。改动后的ANTS模型将Channel类的功能一分为二：InChannel启动一个线程，专门负责数据包的接收与解封装；OutChannel负责封装数据包并调用UDPChannel完成数据包发送工作。对输入数据流的调度工作则通过A-Flow模型来完成。

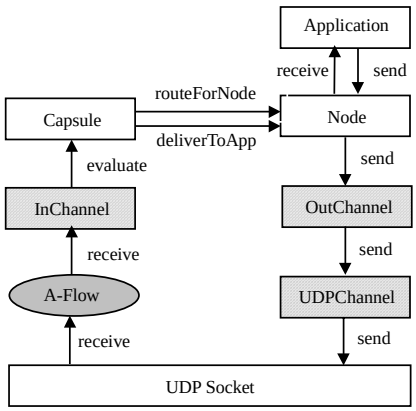


图2　A-Flow在ANTS中的应用

2.2　采用ANEP子层的主动网分发模型

通过ANEP来区分和分配不同的EE，可以在主动网的链路层和EE层之间附加调度功能，作为单独的一个子层[4]。它首先检查到达的主动包的ANEP头的TypeID字段，根据该字段的值(如ANTS为18，PLAN为19等)将主动包分送给相应的EE进行后续操作，其执行过程如图3所示。例如ANTS运行于UDP的3323端口，其他EE在3324端口，ANEP在3322端口侦听主动包的到达。采用这种方式的前提是所有主动包要指定远端的3322端口为目的端口，即

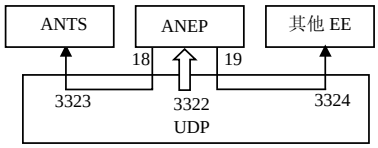


图3 ANEP分发主动包模型

所有到达主动包会送入ANEP子层进行分发，ANEP子层根据TypeID字段值找到对应的执行环境所在的端口，再返回UDP层送入在该端口运行的执行环境执行。

2.3 实验结果及A-Flow的性能分析

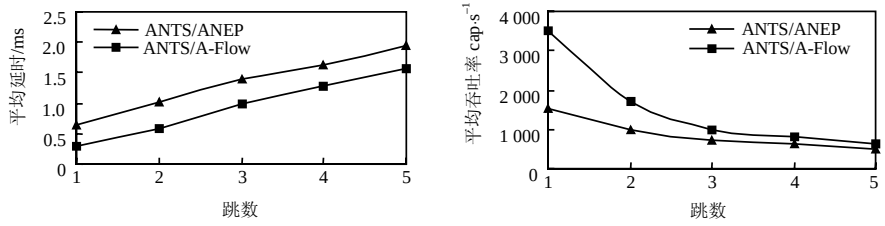
实验环境基于Pentium4 1.8 GHz的处理器，10/100 Mb/s的以太网卡，jdk1.3.1的JVM，内核为2.4.20的Linux操作系统。由于每个主动节点的地址可以通过IP地址和UDP端口号来确定，因此可以在一台机器上模拟运行多个ANTS主动节点。这里IP地址设为Localhost，端口号选择8001至8006，仿真一个直径为5跳共6台机器的简单主动网络。实验对以下两个方面进行测试：

1) 数据包调度 实验使用ANTS的Ping程序进行测试，通过设置ANid和ANEP头部的Discard位，可以检验A-Flow对三类数据包分解调度的效果，如表1所示。对于ANTS主动包，必须既能正确接收，又可以被调度给上层EE计算；对于其他主动包，正确接收后调度给默认流处理；传统IP包将被丢弃。

表1 数据包调度的测试

数据包分类	ANid值	Discard位	Packet received?	Ping succeeded?
ANTS主动包	18	0	是	是
其他主动包	0～256，18除外	0	是	否
传统IP包	任意整数值	1	否	否

2) 性能改善 主动网ANTS/ANEP和ANTS/A-Flow具有相同协议栈且同在用户模式下以Java语言实现，具有可比性。实验对该仿真主动网络分别进行5跳的RTT(Round Trip Time)测试，数据包大小固定为1 500字节。如图4所示，通过对图4a和图4b比较它们的平均延时和吞吐率，得知图4b在性能上有明显的改善。



a 平均延时 b 平均吞吐率

图4 ANTS/ANEP与ANTS/A-Flow的性能比较

实验结果表明，A-Flow模型能有效地完成单节点多EE情况下的数据包调度工作，且很好地解决了这类模型中的性能问题，同时加入了数据包认证功能。

3 结 束 语

第一代主动网的研究至今已有近10年的时间。总的来说，在探索未来网络的体系结构方面，主动网的提出是具有里程碑式意义的。但是在真正要取得广泛应用之前，关于主动网的性能、安全性和互操作性等问题是亟待解决的[5,6]。本文提出的A-Flow模型在为解决这些问题作了一些有益的尝试。

参 考 文 献

[1] Tennenhouse D L, Smith J M, Sincoskie W D, et al. A survey of active network research[J]. IEEE Communications Magazine, 1997, 35(1): 80-86

[2] Druschel P, Gaurav B. Lazy receiver processing[A]. In proceedings of OSDI', 1996, 91-105

[3] Wetherall D J, Guttag J V, Tennenhouse D L. ANTS: a toolkit for building and dynamically deploying network protocols[A]. In: proceedings of IEEE OPENARCH'98, 1998, 117-129

[4] Alexander D S, Braden B, Gunter C A, et al. Active network encapsulation protocol[M]. RFC draft, July 1997

[5] Gelas J P, Lefèvre L. Tamanoir: a high performance active network framework[EB/OL]. http//www.ens-lyon.fr/LIP/ 2004-1-15

[6] Law E, Leung R. Interoperating execution environments in active network[A]. IEEE SoftCOM'02, 2002, 391-395

编 辑 刘文珍