第 35 卷 第 1 期 计 算 机 工 程 2009 年 1 月
Vol.35 No.1 Computer Engineering January 2009
·网络与通信· 文章编号：1000—3428(2009)01—0120—03 文献标识码：A 中图分类号：TP393

# 主动网络节点的 CPU 资源管理策略

李雅红[1]，王建国[2]，魏必凡[2]
(1. 武警工程学院军事经济系，西安 710086；2. 西安工业大学计算机科学与工程学院，西安 710032)

**摘 要：**在 CPU 管理机制的基础上，根据主动网络的特点，提出一种多优先级短信包优先算法。该算法实现简单，能够满足主动网络的需求，同时在仿真实验中对其进行验证。实验结果表明，该算法具有一定的优越性。

**关键词：**主动网络；主动节点；策略

# CPU Resource Management Policy for Active Network Node

LI Ya-hong[1], WANG Jian-guo[2], WEI Bi-fan[2]
(1. Dept. of Military Economy, Armed Forces Police Institute of Engineering, Xi'an 710086;
2. School of Computer Science & Engineering, Xi'an Institute of Technology, Xi'an 710032)

**【Abstract】**On the basis of CPU management mechanism, and according to the characteristic of active network, a Multi-priority Shortest Packet First (MSPF) algorithm is put forward. It is easy to implement this algorithm, which satisfies the requirement of active network. This algorithm is also simulated and the experimental results show it has the superiority to some extent.

**【Key words】**active network; active node; policy

## 1 概述

主动网络[1](Active Network, AN)是种新的网络，其网络节点不但能转发信包，还能对通过这些节点的信包进行计算和处理。它是相对于当前传统被动网络(Passive Network, PN)而言的。传统网络只是被动地传输数据，网络中间节点(如 Router, Switch)对数据本身的语义并不进行分析和理解，其计算功能十分有限。传统被动网络一般只管理中间节点的物理资源和必要的逻辑资源。与传统网络相比较，主动网络由于要运行处理主动信包中携带的主动代码，所需耗费更多、资源更复杂。因此，在主动网络中需要管理的资源范畴比传统网络中的要大得多，除传统网络的资源管理范畴外，主动网络还要管理中间节点和终端节点的各种物理资源以及逻辑资源。

系统资源管理一般由底层的操作系统(OS)完成，在主动网络中，由节点操作系统(NodeOS)对系统资源进行管理。目前，多数主动网络开发者都简单地将现存 PC 机上的 OS(如 Windows, Linux, Unix)充当 NodeOS，由于这些 OS 有完善的资源管理和调度算法，且对上层编程语言(如 Java)的支撑能力较强，因此在实现上较为简单。但主动网络仍然是种通信传输网络，与其他通信传输网络相同，其根本目的不是为了复杂的计算处理而是快速地传输信包。PC 机上的 OS 主要针对各种复杂应用提供最优的资源管理、调度和其他基础设施服务，这些资源管理和调度算法虽然完善、有效，但一般都很复杂。相对于复杂的应用而言，主动代码一般比较简单。美国麻省理工学院的 Tennenhouse D L 等人已经明确指出多数主动信包到达主动节点后，并不需要复杂的资源模型。因此，这些复杂的资源管理和调度算法对主动代码的运行并不适合，而且会降低信包的传输速率。所以，研究和开发一种简单实用的主动网络资源管理机制显得十分必要。本文主要就主动网络节点的 CPU 资源管理机制进行探讨。

## 2 设计方法

### 2.1 主动节点资源管理体系结构

在体系结构设计中，本文采用组件化思想，把基本的、相对独立的功能设计成一个组件，如图 1 所示。

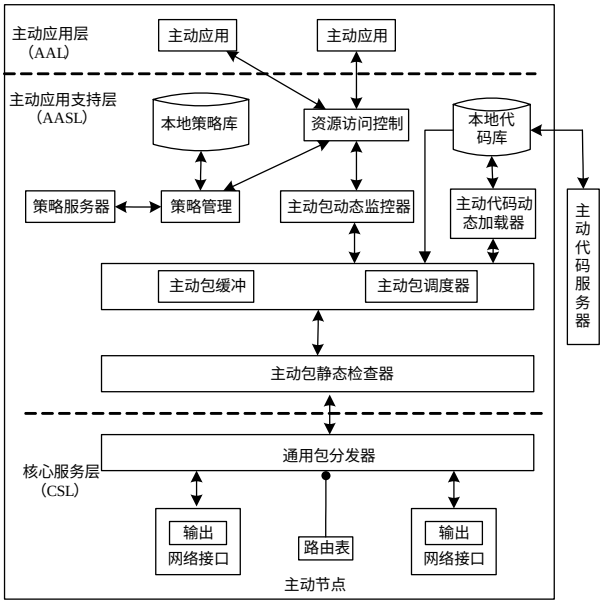


**图 1 主动节点资源管理体系结构**

**基金项目：**陕西省教育厅基金资助项目(02JK147)；陕西省科技厅基金资助项目(2003F24)
**作者简介：**李雅红(1977－)，女，讲师、硕士，主研方向：计算机网络；王建国，博士；魏必凡，硕士
**收稿日期：**2008-05-08 **E-mail：**lhj_lyh@163.com

对图 1 的说明如下：

(1)核心服务层(CSL)

为高层应用提供基本的网络服务，并完成网络硬件设备的仿真。

(2)主动应用支持层(AASL)

向主动应用提供服务，并对主动节点中主动代码的运行进行解析、调度和管理。其中：

1)主动包动态监控器：实时监控主动包的运行时间。

2)主动包静态检查器：完成主动包大小和估计运行时间的检查。

3)主动包调度器：主要完成主动包的调度管理。每当一个主动应用运行完毕，主动包调度器将从缓存的主动包队列中调度出一个主动信包进行处理，而缓冲队列根据最短主动包优先策略建立。

4)本地代码库：用于存放主动代码的本地库。

5)主动代码服务器：提供动态加载主动代码的服务器。

6)主动代码动态加载器：完成主动代码的动态加载。

7)资源访问控制：主要负责审核主动代码的资源访问权限，以保护主动节点的内部资源和状态。

8)策略管理：主要负责向网络上的可靠的策略服务器申请裁决。

9)本地策略库：存放本节点中控制资源访问的权能。

10)策略服务器：主要存放网络中控制资源访问的权能。存在于网络上的某些主动节点。

(3)主动应用层(AAL)

一些具体的主动应用。

### 2.2 CPU 资源管理

一个主动节点可同时运行多个主动应用，即同时处理/运行多个主动信包。一方面为了使各种主动信包能够得到较快的响应和处理，必须保证各个主动应用能够尽快地运行完成。另一个方面为了避免某个主动应用占用过多的 CPU 时间而影响整个主动节点的处理性能，必须保证各个主动应用能够公平地共享 CPU，防止某些主动应用被“饿死”，即某些主动包长时间得不到处理。而信包的处理行为直接影响 CPU 资源的调度，因此，必须有一个合理的调度策略。

(1)信包调度算法原理

所谓信包调度就是要尽快地调度处理主动信包，同时保证一定的公平性。根据主动网络的特点，进行信包调度的基本思想是：引入队列管理原理，构建一个广义队列系统(下文简称队列系统)，如图 2 所示。

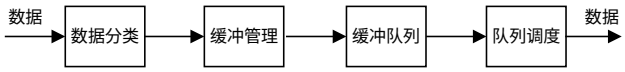


**图 2　广义队列系统**

首先将需要传输的数据根据某种规则进行分类，然后将分类的数据项依照特定的协议插入到相应的缓冲队列中等待被调度处理，最后队列调度算法依照调度策略调度信包处理。其中：

1)数据：也称为队列项，是个抽象概念，在不同的具体环境下有不同的具体含义，如在 IP 网络中是指分组(Packet)，在主动网络中是主动信包(Capsule)。

2)数据分类器：完成数据的分类，并将分类结果传递给缓冲管理器，分类标准可以因队列元素的不同而不同，如可以按服务的优先级分类，也可以按信包的 IP 地址分类。

3)缓冲管理：是指对网络传输节点中队列缓冲资源的管理。在信包传输过程中，其流经的网络传输节点通常采用队列缓存、延迟转发的服务方式以提高输出链路的带宽利用率。缓冲管理机制在信包到达队列前端时依据一定的策略和信息决定是否允许该分组进入缓冲队列，从另一个角度看，也就是做出是否丢弃该分组的决策，因此，也称为丢弃控制。

4)缓冲队列：指狭义队列，用它们来真正缓冲存储数据。

5)队列调度：负责信包的调度，将遵循调度策略的信包传输给 CPU 进行处理，即遵循调度策略从哪个队列中选择信包进行调度。

(2)多优先级最短信包调度策略

在各种作业的调度算法中，短作业优先(Shortest Job First, SJF)算法的调度性能较好，且在实现上较为简单。因此，本文引入 SJF 的调度原理，依照上述信包调度的基本思想，并结合主动网络的特点，提出一种多优先级短信包优先(Multi-priority Shortest Packet First, MSPF)算法，即总是优先调度具有最高优先级且运行时间最短的信包作为下一次处理的对象。MSPF 算法中使用的队列(广义队列)如图 3 所示。

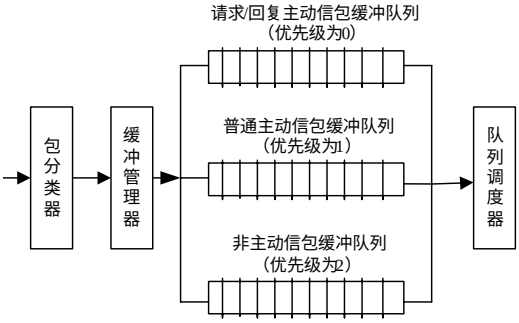


**图 3　MSPF 算法使用的队列系统**

对图 3 的说明如下：

包分类器

在原型系统中采用离散方法实现信包的封装[2]，使用 Capsule代替传统网络中的Packet，并在原型系统中定义 2 大类Capsule：

1)非主动包(Non Active Capsules)：使用非主动包来仿真传统网络中的分组(Packet)，只有目标节点才接收该信包，中间节点只进行转发。该类信包的优先级最低(优先级为 2)。

2)主动包(Active Capsules)：所有主动节点均对它进行处理。主动包又被分为 3 个子类。

①普通主动包(NORMAL Capsule)：携带有指示主动代码指针的主动信包。该类信包的优先级要高于非主动包(优先级为 1)。

②主动代码请求主动包(REQUEST Capsule)：根据主动代码的分发机制，REQUEST Capsule 用于请求主动代码服务器获取主动代码。

③主动代码回复主动包(REPLY Capsule)：用于回复 REQUEST Capsule，传输请求的主动代码。REQUEST Capsule 和 REPLY Capsule 的优先级相同，它们的优先级最高(优先级为 0)。

包分类器根据上述信包分类的规则和优先级的约定，将接收的信包进行解析分类，确定信包的优先级，其返回值为 0~2，并提取信包的长度(通过分析信包长度字段获得)。

缓冲管理器

目前缓冲主要采用 2 种不同的控制策略[3]：基于动态资

源预留策略和基于选择性丢弃策略。本文算法采用基于选择性丢弃的策略。缓冲管理器(又称对列管理器)根据包分类器返回的优先级将信包分发到相应的队列，并根据该信包长度遵循从小到大的顺序插入到队列中相应的位置。当队列容量(包总个数或包总大小)满时选择丢弃队列中新到的包时，即使用弃尾法(Drop Tail)。

队列调度器

队列调度指按照一定的规则来决定从等待队列中选择哪个信包进行处理，使所有队列中的信包能够按照预定的方式输出。队列调度规定了信包从每个队列离开时的规则，通过这些规则使得不同类型的信包得到不同等级的服务。信包调度的主要目的有 2 个：1)相对优先级控制：调度器使得重要的分组得到最好的服务，次要分组得到较差的服务，表现出使用共享资源的一定的相对优先级。这种优先级关系可以在系统初始化时设置为静态优先级，也可以在运行过程中根据系统状态进行调节，即计算动态优先级。2)时延控制：某些实时应用要求严格的时延范围保证和抖动控制，而某些非实时应用则时延的要求较为宽松。

本文提出的 MSPF 算法根据多优先级和短信包优先的规则，在相对优先级和时延 2 个方面都进行了控制。当执行 Dequeue(出队)操作时总是使排序值最小的元素出队。队列调度器决定优先选择哪个队列进行 Dequeue(出队)操作，对于多优先级、短信包优先调度而言，队列调度器先查寻优先级最高(即优先级为 0)的非空队列，然后对该队列执行 Dequeue 操作。当高优先级为空时选择较低优先级队列。具体的调度流程是：

Step1 从具有最高优先级的请求/回复主动信包缓冲队列中调度信包进行处理；

Step2 判断请求/回复主动信包缓冲队列是否为空时，若不为空，则转 Step1；否则，说明请求/回复主动信包缓冲队列中不存在需要处理的信包，就继续执行 Step3；

Step3 从普通主动信包缓冲队列中调度信包；

Step4 当处理完一个普通主动信包时，判断请求/回复主动信包缓冲队列是否为空，若不为空，则转 Step1；否则，继续从普通主动信包队列中调度信包进行处理，直到该缓冲对列为空；

Step5 当普通主动信包缓冲队列为空时，从非主动信包缓冲队列中调度信包；

Step6 当处理完一个非主动信包时，判断请求/回复主动缓冲队列是否为空，若不为空，则转 Step1；否则，判断普通主动信包缓冲队列是否为空，若不为空，则转 Step3；如果请求/回复主动信包缓冲队列和普通主动信包缓冲队列均为空，则继续从非主动信包队列中调度信包进行处理，直到该缓冲对列为空。

## 3 具体实现

在图 1 中，主动信包调度器实现了主动信包的管理，即用主动信包调度器完成主动节点中 CPU 的管理。主动信包调度器由 PktScheduler 类实现，PktScheduler 使用 PriorityCalendarQueue 完成了信包的缓存，下面分别给出这 2 个类的实现。

(1)PriorityCalendarQueue 类。PriorityCalendarQueue 类实现了图 4 中的队列系统。

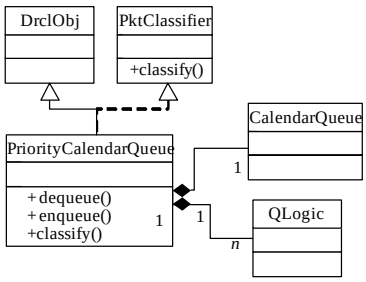


**图 4 PriorityCalendarQueue 类的 UML 示意图**

从图 4 可以看出，Classify 方法继承自 PktClassifier 接口，它实现了主动包分类器的功能，并根据主动包的类别对其优先级进行划分；QLogic 完成缓冲管理功能，当前只实现弃尾算法，当队列满时将后来到的主动包丢弃；Calendar Queue 完成主动包的缓存，它可自动将进入队列的信包进行排序，在 CPU 管理这里的排序依据是信包的长度；Enqueue 方法实现了主动信包的入队(进入 CalendarQueue)，在信包进队之前先判断对应优先级队列是否已满，如果该队列已满则调用 QLogic 完成进行丢弃操作；Dequeue 方法实现了主动信包的出队。

(2)PktScheduler 类。PktScheduler 类是主动信包调度器的具体实现，它继承自 Module。dataArriveAtDownPort 方法处理来自主动包静态检查器(见图 5)上传的主动包。dataArrive AtUpPort 处理来自主动代码获取及分发器的主动代码请求主动包和主动代码回复主动包。

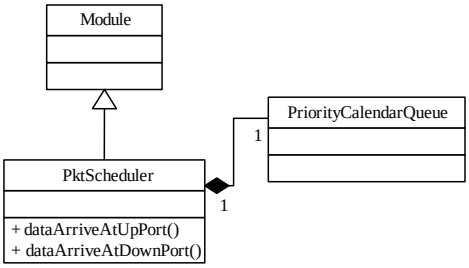


**图 5 PktScheduler 类的 UML 示意图**

## 4 结束语

本文根据主动网络的特点，并在主动信包调度的基础上引入队列管理原理，构建一个广义队列系统。先将需要传输的数据根据某种规则进行分类，然后将分类的数据项依照特定的协议插入到相应的缓冲队列中等待被调度处理。由于主动代码一般比较简单，需要运行的时间很短，因此在各种队列作业的调度算法中，采用短作业优先的调度算法。对主动节点而言，还存在许多关于内存、带宽等资源管理机制的问题，这是下一步的研究内容。

## 参考文献

[1] Tennenhouse D L, Smith J M, Sincokie W D, et al. A Survey of Active Network Research[J]. IEEE Communications Magazine, 1997, 35(1): 80-86.

[2] 王建国. 主动网络关键技术研究[D]. 西安: 西安交通大学, 2002.

[3] 林 闯, 单志广, 任丰原. 计算机网络的服务质量[M]. 北京: 清华大学出版社, 2004.