# 高性能网络数据捕获方案的分析和优化

张玉琴 王 亮 李盘文
(中国飞行试验研究院 西安 710089)

**摘 要**：针对目前对高速网络数据监控、分析和性能预测的需求，对实时存取原始网络数据的软件解决方案进行了分析。以 WinPcap 为例，深入其内部运行机制，详细展示了网络数据包捕获中的各个阶段，并给出了每一阶段的详细性能评估，同时指出其中的性能缺陷和瓶颈所在，最后给出了改进各个阶段性能的详细解决方案，使其能明显改进高速网络数据包捕获速率，节省系统开销，为之后的数据包处理和分析留出充裕的资源。最后对改进方案进行了定量的分析和论证。

**关键词**：高速网络；数据包捕获；WinPcap 库函数
**中图分类号**：TP393.092 **文献标识码**：A **国家标准学科分类代码**：520.1050

# Analysis and optimize of full-fledged network data capture solution

Zhang Yuqin Wang Liang Li Panwen
(Chinese Flight Test Establishment, Xi'an, 710089, China)

**Abstract**: For the demand of monitoring and analyzing the high-speed network data and predicting the capability, analyze the solution of the software for saving and reading the original network data. In this paper, take WinPcap as example, particularly prodict each stage of capturing network data with the inner running system, provide the detailed performance evaluation, meanwhile find out the disadvantage of capability and choke point. Then elicit the particular solution which improve the capability of each stage for improving the rate of capturing the packet obviously, saving system cost and leaving abundant resource to process and analyze the pocket. In the end, analyze and demonstrate the solution.

**Keywords**: high-speed network; data package capture; WinPcap

## 1 引 言

现代网络化设备能力的不断提升，网络的带宽和复杂性也随之提升。对网络进行监控、故障分析也变得越来越复杂，需要很高的专业能力和专用工具。一些专用工具(如网络分析仪和监控设备)都是基于专门的硬件，通常都很昂贵，难于部署，同软件相比灵活性非常低[1]。

目前为止，大量为软件应用提供实时存取原始网络数据的基于软件的解决方案(通常以标准操作系统的扩展形式出现)被提出。这些解决方案通常以库函数的形式实现，如广为人知的 WinPcap 和 libpcap，能够应用在很多的操作系统上。这些库函数导出一系列的原生函数，使应用程序能够不经过其他网络协议层直接与网络进行互动。软件组件容易部署，灵活性也高：一个包捕获组件能为大量的应用工具(如网络监控器、流量分析器和实时检查软件等)提供数据包的系统底层存取。另外成本低廉，容易升级，这也是专业人士喜欢使用软件工具进行网络监控和分析的原因。但是，性能一直是基于软件的工具的一个弱点，当处理高速网络时必须使用硬件方案。虽然目前 CPU 的性能非常强大，但还是不能在几倍千兆网络速度的链路上进行软件实时流量分析[2]。

该文指出了网络分析需要的组件并测量了各部分的权重。之后在 WinPcap 协议的基础上实现了一系列的优化并进行测试，将改进的结果以量化的形式给出。

## 2 WinPcap 数据包捕获结构体系

通过分析一个典型的包捕获和流量分析结构体系的各部分组件来作为本文中使用的模型。特别的，专注于由网络接口卡接受，传输到内存，到达最终的应用程序的数据包所走过的路径。WinPcap 作为广泛应用的包捕获协议，其基本原理与其他解决方案是一致的。

### 2.1 网卡和网卡驱动程序

现代网卡有一个很小的板载内存，通常是几千字节。

收稿日期：2015-06

用来在全速状态下接收和发送数据包，独立于主机的处理能力。另外，网卡在数据包还存储在板载内存上时进行一些初步处理，如 CRC 校验、短以太网帧检测，这样无效的帧就可以立即丢弃。

当网卡接收到一个有效的数据包后，网卡向总线控制器产生一个进行数据传输请求。此时，网卡控制总线，将数据包传输到主机内存中的网卡缓存中，释放总线，向高级可编程中断控制器芯片产生一个硬件中断。这个芯片唤醒系统的中断处理程序，触发网卡驱动中断服务例程。中断服务例程通知捕获驱动，进行数据包的过滤和存储[3]。

### 2.2　包捕获驱动

通常来说，包捕获组件对其他软件模块是透明的，如协议栈，因此，不会影响标准系统的行为。通过在系统中插入一个钩子函数，通常是一个叫做 tap()的回调函数，使其能够在新数据包到达时接到通知。

tap 函数进行的第一个动作就是过滤，例如，数据包进行分析是否是用户需要的数据包。WinPcap 提供一个用户级的 API 函数来将高级表达式(如接收所有 UDP 数据包)转换成一系列的伪指令(如“如果以太网头的以太网类型字段是 IP，并且 IP 头部的协议类型字段是 17，那结果为真”)，并将其发送至过滤器，并激活。

过滤通过后，将接收的数据包与物理层信息，如长度和接收时间戳，关联在一起，这对应用程序存取和处理数据包很有用。数据包之后被拷贝进一个缓存，通常被称为内核缓存，存储等待送往用户级的数据。对捕获进程的性能来说，这个缓存的大小和结构是非常重要的。一个大的合理构建的缓存系统可以补偿在突发传输中缓慢的用户级程序所带来的不良影响，减少捕获驱动(内核缓存)向用户级程序传输数据所引起的系统调用的数量[4]。

用户级程序通过类 read 系统调用取得数据包。检查内核缓存的状态：如果缓存不为空，其内容就传输到用户分配的内存，当数据包拷贝到用户级时，程序立即唤醒开始处理数据包。

## 3　性能评估

### 3.1　测试平台

图 1 是进行分析用的测试平台：2 台 PC 通过快速以太网链路直接连接。一台 PC 作为流量发生器，同时另一台为测量目的而加装了分析扩展的 libpcap 用来进行实际的测试。特别的，分析扩展使用了 Pentium 家族微处理器中的性能监控计数器。这个家族中的处理器具有一定数量的内部计数器(类型和数量因型号而不同)可以将其进行编程，用来跟踪事件，如译码的指令条数，接收的中断数量等等。例如 CPU_CLK_UNHALTED 计数器存储给定时间间隔内 CPU 消耗的实际时钟周期数，并能区别在内核级和用户级分别消耗的时钟周期数。程序可以通过 rdpmc 指令获得这个计数器的值。

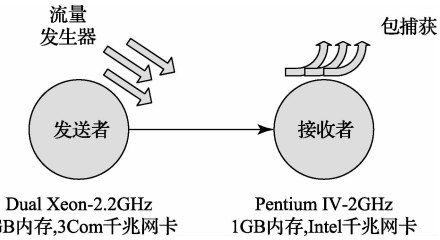


图 1　测试平台

### 3.2　其他的处理开销

处理数据包时要涉及一些其实不是包捕获体系的组件，如网卡驱动和操作系统。这些组件被引入所带来的以时间表示的开销，叫做额外处理开销，占总开销很大的比重。

1)操作系统

网卡收到数据包后第一个启动的软件就是操作系统。系统处理的开销随包速率变化，但主要还是与中断的数量成比例，这个中断就是网卡通知系统一个包已经被接收并等待处理。在测试平台上每一个中断大约需要 2 700 个时钟周期。在测试中，网卡中断产生速率是 2 999 中断/s，帧速率是 148 Kfps，也就是说每一帧数据平均要消费 54 个时钟周期。这个开销与网卡和帧速率有关[5]。

2)网卡和设备驱动

虽然网卡不需要中央处理器的帮助来完成它的工作，但是它的行为会影响其他组件，特别是操作系统和网卡驱动。例如要相应的中断数量(影响操作系统的开销)和存取网卡内部寄存器的 I/O 操作的数量(影响驱动程序的开销)对性能有明显的影响。对后一条，中断服务例程函数(通常每次中断调用一次，是设备驱动程序的第一个函数)非常简单但是开销很大(大约 850 个时钟周期)，因为在网卡上要进行一系列的 I/O 操作来通知驱动正在处理数据包。

当负载流量超过一定值时，可以通过一次中断取回多个数据包来改善性能。这减小了操作系统开销(更少数量的中断)和驱动开销(更少的网卡 I/O 操作)。得到的数据显示，每中断传输的包数量与网络负载成线性关系。测试平台在每秒接收 148 K 帧的条件下，达到了每次中断处理 49.61 帧(对应 2999 中断/s)。这意味着在低包速率下底层组件(中断处理，网卡驱动)在包处理中的相对开销更大一些，并且随着包速率的增加变得不那么重要了[6]。

通过测试，网卡驱动每包的花销(最大包速率 148 809 包/s)1 497 个时钟周期。

### 3.3　捕获驱动

分析了捕获驱动中所有组件的开销，例如数据包从网卡驱动(和操作系统)到用户级程序这条路径上的开销。

1)过滤程序

明显地，过滤程序的开销不仅仅与过滤引擎本身的效率有关，还与过滤指令的复杂度有关，例如，每包需要进行检查的次数。图 2 所示是 3 个过滤器的开销：最简单的是只接收 IP 包(需要 3 条伪指令)，另一个是检查 5 个 TCP

端口(21条伪指令),最后一个更复杂,检查10个IP地址和10个TCP端口(50条伪指令)。

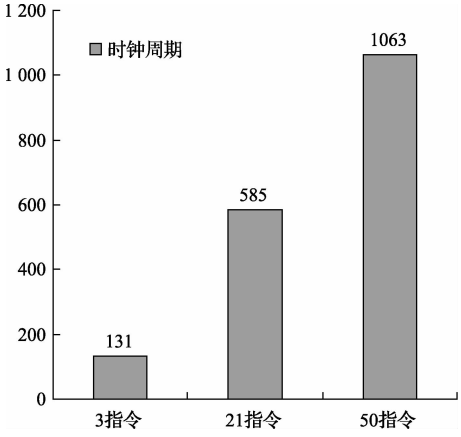


图2 不同复杂度的过滤器

2)内存拷贝

通过第二节的介绍,每个数据包在到达用户程序之前都要在主内存中拷贝2次:第1次将包从网卡缓存传送到内核缓存中,第2次将其传送到用户空间程序缓存。图3所示是测试平台上以CPU周期/字节为单位的2次拷贝的花销。

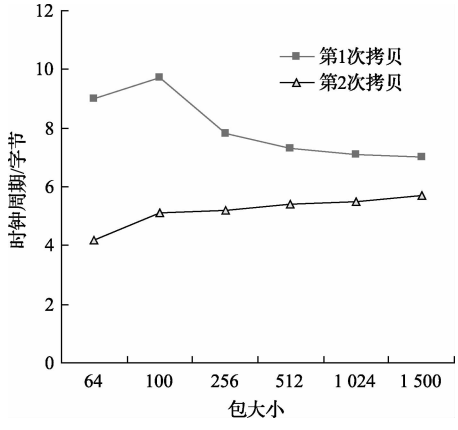


图3 不同包大小下内存拷贝的时钟周期开销

依据NDIS指南,第1次拷贝使用的是NdisTransferData()函数。这个函数的高开销有两个原因。

根据DDK文档,驱动程序必须使用这个函数,因为在网卡驱动将控制权转交给包捕获驱动时,数据包可能是不完整的。因此,这个函数首先检查NIC是否将完整的数据包传输到内存上了;如果没有,就等待,直到传输完毕。

这个函数操作的数据不在CPU cache里。要拷贝的数据包刚刚通过DMA从网卡板载内存传输到主内存中。

第1次拷贝的花销是540～10 500时钟周期/包,第2次拷贝是259～8 550时钟周期/包。事实上,考虑每个包额外的20字节的头部(包含一个时间戳、包长度和其他信息),第2次拷贝的总花销在364～8 664时钟周期/包[7]。

3)与程序的交互

程序与包捕获驱动之间的所有交互都是通过系统调用实现的。windows提供ReadFile()、WriteFile()和DeviceIoControl()系统调用来进行I/O控制。这些调用导致两个上下文切换:第1个是从用户层向内核层传递命令,第2个是向用户层返回控制结果。

在100%CPU负载的情况下,测得每次系统调用传输256 K字节,相当于3 200个数据包。这种情况下,每包的上下文切换大概是10个时钟周期,几乎可以忽略。

4)其他处理组件花销

驱动通过Win32函数KeQueryPerformanceCounter()为数据包获取时间戳,这是内核函数中唯一一个能提供毫秒级精度的。这个函数的花销非常大,因为它要与系统的时间芯片进行交互,在本测试平台上大概是1 800个时钟周期。

其他的花销包括NDIS和内核的交互、内核中使用的内存缓冲区的管理和每个数据包添加头部信息的花销,大概在830个时钟周期。

### 3.4 全部的开销

图4展示的是处理帧速率为148 Kfps的64字节数据包时的每种操作的相应花销。其中,过滤器的伪指令条数为21条;全部花销为每包5 680时钟周期。

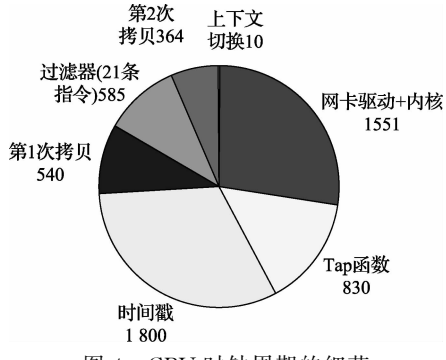


图4 CPU时钟周期的细节

## 4 性能优化

### 4.1 过滤进程

WinPcap使用的过滤系统是BPF(BSD包过滤),于

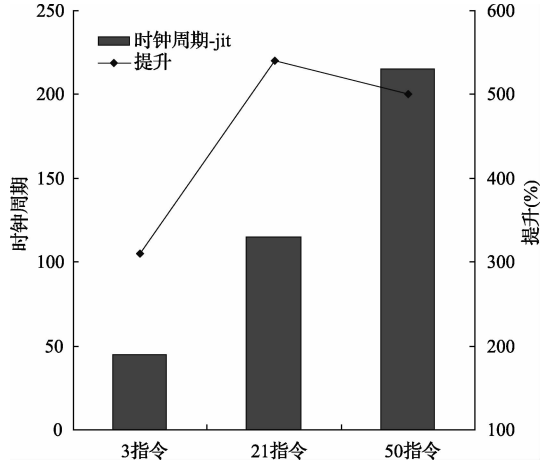


图5 使用JIT的过滤器的开销和提升

1993 年提出。在所有优化 BPF 的方案中，动态代码产生器可以保证有效的性能提升。因此，将 BPF 过滤器转换成 80×86二进制代码的 JIT(just in time)引擎在 NPF 上进行实现。如图 5 所示，本次优化大概提升了 3.1～5 倍性能。也就是总体上得到了 8%的改善[8-9]。

### 4.2　内存复制

如前所述，第 1 次包复制的开销大于第 2 次复制，主要原因是 NdisTransferData()函数所带来的额外检查。但是，目前市面上的网络控制器都是在通知 NIC 驱动前就将完整的数据包传输到了内存，因此，此处可利用标准 C 库函数来完成拷贝，结果如图 6 所示。

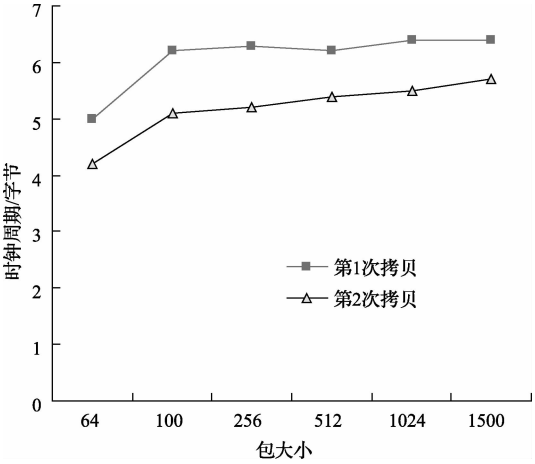


图 6　内存拷贝的开销

通过本次优化，第 1 次拷贝的花销从 540～10 500 时钟周期/包降至 300～9 600 时钟周期/包，第 2 次拷贝不变。相当于提升了总体性能的 4%。

### 4.3　时间戳

通过使用 Intel/AMD 处理器的时间戳计数器(TSC)来代替 KeQueryPerformanceCounter()函数获取时间戳。x86 汇编器提供一个非常快速的(一个时钟周期)指令，rdtsc 来获取时间戳。整个时间戳获取的开销大概在 270 个时钟周期，主要花销在了将计数器的值转换成标准的结构体 timeval 上。本次优化提升了总体性能的 27%。

### 4.4　tap()函数的优化

通过使用标准 C 库函数代替 NdisTransferData()的同时，也简化了 tap()函数。通过 NdisTransferData()传输数据包，需要分配存储包的结构体并在拷贝结束时调用一个回调函数。因此，使用标准 C 库函数可以有效地提高效率。通过实验，tap()函数的开销从 830 降至 560 个时钟周期，相当于 5%的总体性能提升。

### 4.5　优化后的全体开销

图 7 所示是与图 6 在相同试验条件下的经过优化后的全部开销的详图。优化后全体开销为 3164 个时钟周期。与优化前相比有一半的提升。

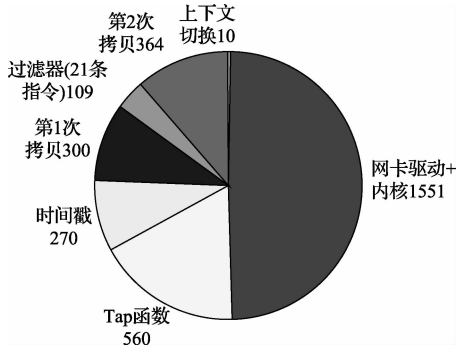


图 7　CPU 时钟周期的细节

## 5　结　　论

展示了分析和优化网络分析和监控工具的基础—包捕获软件。指明了在包捕获过程所引入的各个组件并测量了个部分的 CPU 时钟开销。分析了各个组件的瓶颈所在并进行了优化，结果显示，经过优化，系统开销节省了一半，为之后的数据包处理和分析留出了充裕的资源。

## 参考文献

[1]　王亚. 高速网络环境下数据包捕获技术的分析[J]. 数字技术与应用，2012 (12)：194-195.

[2]　RISSO F，DEGIOANNI L. An architecture for high performance network analysis[C]. Proceedings. Sixth IEEE Symposium on Computers and Communications. IEEE，2001：686-693.

[3]　王建虹. 一种高可靠性双机冗余系统的设计[J]. 国外电子测量技术，2008，27(4)：42-44.

[4]　黄海，张柯，张建明，等. 基于 Agent 技术的网络控制系统研究[J]. 仪器仪表学报，2008，29(8)：1679-1685.

[5]　方亮，喻金科. 基于 Linux 的高速网络数据捕获技术[J]. 计算机与现代化，2010 (9)：30-33.

[6]　PAPADOGIANNAKIS A，VASILIADIS G，ANTONIADES D，et al. Improving the performance of passive network monitoring applications with memory locality enhancements [J]. Computer Communications，2012，35(1)：129-140.

[7]　戚玉华，吴学智，顿新平. 高速网络数据流分类系统[J]. 电子测量技术，2006，29(5)：148-150.

[8]　史久根，胡小博. 高效节能的无线传感器网络数据收集协议[J]. 电子测量与仪器学报，2012，26(5)：437-445.

[9]　李伟，鲁士文. Snort 数据包捕获性能的分析与改进[J]. 计算机应用与软件，2005，22(7)：104-105.

## 作者简介

**张玉琴**，1984 年出生，硕士研究生，工程师。主要研究方向为传感器校准技术及记载测试技术。

E-mail：zyq657@sina.com