密级：公开

编号：15036043

國防科学技術大学

硕士研究生学位论文

开题报告

论文题目：基于多核 NPU 的 TCP 数据传输卸载

学　　号：13060041　**姓名：**李杰

一级学科：计算机科学与技术

研究方向：网络安全

指导教师：陈曙晖　**职称：**研究员

学　　院：计算机学院

开题时间：2015 年 3 月 17 日

国防科学技术大学研究生院制

二〇一四年十二月

说　明

一、开题报告应按下述要求打印后装订成册：

1.使用 A4 白纸，双面打印；

2.封面中填写内容使用小 3 号仿宋字体；

3.表中填写内容使用 5 号楷体字体。

二、封面中的编号由学院填写，采用八位数编码，前四位为审批日期，精确到年月即可，第五位为院别，后三位为审批流水号（按年）。如 15016100，为 6 院 15 年审批的第 100 位开题申请者，审批时间为 15 年 1 月份。院别代码与编制序列一致，海洋科学与工程研究院代码为 0。

三、开题报告表中学员填写的内容包括学位论文选题的立论依据、文献综述、研究内容、研究条件、学位论文工作计划、主要参考文献等，指导教师认可学员开题报告内容后，对学员学位论文选题价值、对国内/外研究现状的了解情况、研究内容、研究方案等方面予以评价。

四、开题报告评议小组一般由 3-5 名具有正高级专业技术职务的专家（包括导师）组成，其中一名为跨一级学科的专家。

五、博士生开题报告会应面向全校公开举行，评议小组听取研究生的口头报告，并对报告内容进行评议审查。

六、若开题报告获得通过，应根据评议小组意见对开题报告进行修改，并在开题报告会后两周内，将评议表和修订后开题报告纸质版原件交学院备案；若开题报告未获得通过，则填报延期开题申请，由原开题报告评议小组重新组织开题报告会。

1.学位论文选题的立论依据

（1） 课题来源

项目：面向三网融合的统一安全管控网络

（2） 选题依据

TCP(Transmission Control Protocol)协议是互联网中的一个重要协议，它提供了端系统之间的数据可靠传输、多路并传、基本的流量控制和拥塞控制等功能，在互联网中得到了广泛的应用。虽然服务器的处理能力、PCI总线的带宽都有很大的提高，但服务器端TCP服务的性能仍然是内容提供商、游戏提供商、门户网站、搜索引擎所面临的挑战。提高TCP服务的性能，不仅能够降低服务器集群的数量，也能降低功耗，实现更加绿色环保的IT服务。

TCP协议在端系统的处理开销主要包括三部分：中断操作，数据复制和协议处理。

由于数据包到达的时间未知，端系统对数据包的接收一般设计为异步中断的方式。传统的网络接口驱动程序每次和协议栈交互一个数据包，这样的好处在于封装隐藏细节，驱动程序不必关心协议栈如何处理接收到的数据包，协议栈也不必关心网卡驱动程序如何发送数据包。当一个数据包到达网卡时，网卡产生一个中断信号，触发驱动程序对数据包进行接收：将数据从网卡DMA到内存中，构造相应的数据结构（Linux中为skbuff，Unix中为mbuf），传递数据包给协议栈处理。这里端系统会进行上下文切换，当网络带宽较小，数据包到达不频繁时，这样的设计对端系统的性能影响并不明显，然而当网络带宽增至千兆和更高，数据包频繁到达时，频繁的中断操作带来的上下文切换会严重消耗CPU资源，降低端系统的整体性能。后来的Linux网卡驱动程序多采用NAPI的设计：将中断与轮询结合，以谋求一定的平衡。当一个数据包到达网卡后，网卡产生中断信号，在驱动程序的中断处理函数中，首先禁用网卡中断，然后进入一个循环，采用轮询的方式接收一定数量（threshold）的数据包，之后使能网卡中断，将接收到的数据包交给协议栈处理。这样的好处在于不再为每个数据包产生中断信号，减少中断数量，然而threshold的选择将关键影响系统的性能。

TCP为应用层程序提供可靠的数据传输服务，保证数据包的可靠有序。数据发送方面，应用层将数据交给内核之后，就认为数据已经被正确发送，应用程序将释放或者重复使用应用层的发送数据缓冲区，而TCP为了保证数据的可靠有序传输，需要在数据包发送出错或超时的情况下进行重传，所以重传数据只能保持在内核空间。数据接收方面，TCP接收到的可能是错误或者乱序的数据包，TCP需要等待接收到发送方重传错误的数据包，并将乱序的数据包重新排序后，才能通知应用层接收数据，所以TCP接收数据也只能保持在内核空间。这样，发送过程的数据复制步骤为：CPU将数据从用户空间复制到内核空间，驱动程序将数据从内存DMA到网卡；接收过程的数据复制步骤为：驱动程序将数据从网卡DMA到内存中，CPU将数据从内核空间复制到用户空间。在高速网络中，数据频繁的发送和接收，DMA的频繁启动和CPU复制数据的操作将严重影响系统的整体性能。

TCP协议处理开销主要包括：（1）分割发送的应用数据为合适大小的数据块；（2）每次数据发送后启动重传定时器；（3）收到数据后发送延时确认；（4）对收到的数据进行校验和计算；（5）将接收到的乱序数据包重新排序；（6）检测和丢弃重复数据包；（7）进行流量控制和拥塞控制。

目前以太网的发展速度远高于存储器和 CPU 的发展速度，存储器访问和 CPU 处理网络协议已经成为 TCP 的性能瓶颈。随着网络带宽的不断增大，端系统处理网络协议的开销也越来越大，大约需要 1GHz 的 CPU 处理能力对 1Gbps 的网络流量进行协议处理。网络带宽的增大对 CPU 造成了沉重的负担，如能将 TCP 数据发送过程中的分包，校验和计算，以及数据接收过程中的校验和计算，乱序重组任务卸载到网卡中的多核 NPU 中执行，而驱动程序

与网卡之间进行大块的数据交互，则在数据接收过程中，可减少送往协议栈处理的数据包个数，减少协议栈处理数据包头部的开销；同时还可以减少网卡产生中断的数量，减少中断上下文切换带来的 CPU 开销；此外，由于驱动程序与网卡之间进行大块的数据交互，也可减少 DMA 的次数，提升端系统的整体性能。

2. 文献综述

（1） 国内/外研究现状及发展动态

从TCP诞生之日起，人们就没有停止对它的研究。通过优化TCP的处理机制，可以减少主机开销，从而获取更高的带宽[1]。

2.1.1 优化校验和计算

校验和的计算涉及所有的传输数据，访问存储器的操作使得本来并不复杂的校验和计算变得开销很大。Borman对原始的校验和计算进行了包括延迟进位、循环展开、结合校验和的复制操作以及递增的校验和更新优化等四个方面的优化[2]，被广泛的实现在系统中。Mallory和Rijsinghani提出了进一步改进，允许在改变部分数据后，不需要遍历所有的字节而重新计算校验和[3-4]。Kleinpaste通过修改数据复制的语义实现数据与控制信息分开传输，主机计算协议头部的校验和，并将结果通知硬件，硬件计算用户数据的校验和[5]。Henriksson等人设计了可与通用处理器集成的TUCFP芯片，用于并行计算32位校验和数据[6]。

2.1.2 优化大概率事件处理

对大概率事件进行处理优化是许多加速优化方法的基本原则，采用高速缓存实现PCB（Protocol Control Block）的TCP实验结果表明高速缓存命中率可以达到80%[7]。Jacobson基于两种常见情况提出了首部预测：（1）TCP发送数据后，该连接上接收到的下一数据包是对发出数据的应答；（2）TCP接收数据后，该连接上接收到的下一数据包是顺序到达的报文。实验表明，在LAN中首部预测的命中率可达97%，当数据传输跨越WAN时，数据首部预测命中率在83%到90%之间。

2.1.3 减少数据复制

随着网络带宽的增加，TCP数据传输量也增大，相应的数据复制操作开销也增大，存储器带宽逐渐成为约束TCP性能的瓶颈。为此，人们研究了很多方法避免数据的多次复制。

Jerry Chu提出零拷贝技术[8]，实现在Solaris系统中，操作系统预先将NIC的存储空间映射到用户空间或内核空间，应用程序直接访问NIC的存储空间。NIC接收到数据后一直将数据存放在自己的存储空间内，直到应用程序对数据进行访问，数据才经过数据总线到达CPU。零拷贝技术需要修改相应的系统调用，同时要求NIC拥有较大的存储空间存放数据，并能够将数据正确的映射。

Dalton在实验中采用了内核与NIC共享存储空间的方法[9]，内核管理NIC的存储空间，使用DMA或者Programmed IO在NIC存储空间和应用程序缓冲区之间移动数据，这样做的好处在于应用程序不需要修改即可运行。

Druschel和Peterson使用应用程序与内核共享存储区域的技术实现快速缓冲[10]，系统在内核空间和用户空间分别定义了具有共享语义的新API，使用DMA在NIC存储区域、应用程序与内核共享的存储区域之间进行数据移动。然而快速缓冲的技术要求应用程序使用系统定义的共享语义API，缺乏兼容性，共享区域的管理需要应用程序、内核和NIC之间密切复杂的配合。

Kleinpaste通过修改协议处理，增加了支持数据与控制信息分开传输的复制语义[5]，协议在处理过程中只传输协议头部数据和用户数据的索引，而用户数据一直存储在用户空间或者NIC存储空间，复制操作只在用户数据最终进行传输的时候进行。

Greg Buzzard等人提出的Hamlyn机制可以避免由于接收方缓冲区溢出而造成的数据包丢失[1]，该机制由发送方在发送数据之前决定数据在接收方的存储位置。

Rodrignues等人提出提前将接收数据置入应用程序缓冲区的方法对TCP协议处理进行

有效的加速[11]。

Culley ,Garcia和Hilland提出远程直接内存存取协议[12]，在数据包中携带数据在接收方的存储位置信息，接收方的NIC接收到数据后直接将数据存储到应用空间中，避免多次复制操作。

2.1.4 减少中断、cache命中率优化和多线程技术

减少中断对于减轻主机开销，优化TCP具有重要的意义。

Ranjaragan等人在TCP卸载实验中，使用一台设备专门处理TCP，并且将异步触发的中断用轮询代替[13]，此技术的关键在于控制轮询频率，轮询频率过高将导致资源的浪费，而轮询频率过低将导致系统不能及时的处理数据。

Chase等人提出了中断合并的技术[14]，用于减少中断数量。NIC不再为每个接收到的数据包产生中断，而是等待接收到一定数量的数据包后才产生中断，之后主机再处理所有接收到的数据包。中断合并会造成单个数据包的端到端延时不确定。

Juan研究了TCP数据传输过程中浪费掉的中断数量[15]，发现如果以小于MTU的长度对发送数据进行分割，将产生更多的中断，增加单个数据包的传输数据长度，每次以MTU作为长度传输数据，可以有效的减少中断数量。另外，如果NIC可以过滤局域网内的广播报文和某些UDP数据包，也可以有效的减少中断数量。

Mosberger等人通过修改编译器增加指令的cache命中率，在不修改指令的条件下减少每条指令的执行开销，从而减少数据包的处理开销[16]。

由于多个TCP流之间通常表现出弱相关性，可以采用多线程技术对多个TCP流同时进行处理[17]。多线程处理多个不同TCP流可以很好的隐藏存储器访问延时。

2.1.5 TCP Offload Engine

TOE（TCP Offload Engine）可以极大的减少主机开销[18]，提升性能。它利用硬件执行速度快的优点，在硬件上实现TCP协议处理。在TOE的具体实现中，可以采用通用处理器或可编程器件（FPGA等），也可完全采用ASIC（Application Specific Integrated Circuit）进行设计，前者的好处在于灵活性高，而后者可以获得更高的性能[19]。

Henriksson、Nordqrist 和Liu等人设计了一种嵌入式协议处理器，在上面运行实时操作系统，着重于数据包的快速高效接收[20]。

Ang使用i960RN当作NIC的控制处理器，试图实现TOE[21]。系统主机端运行实时操作系统RTX，在数据的发送路径和接收路径上做了优化，避免数据多次复制。实验结果表明，在传输大量数据包时主机CPU的使用率有所降低，但仍然比预期的高很多。Ang指出，硬件设计和缓存管理是影响系统性能的主要瓶颈。

TOE虽然可以降低主机开销，却有许多缺点[22]：（1）在NIC上实现传输层要比只实现链路层功能复杂得多；（2）TOE系统中主机和硬件之间的接口设计特别复杂；（3）TOE需要上层协议的支持才能将数据直接存放到存储器中；（4）TOE需要和主机同时维护每条TCP链接的状态；（5）硬件更新的难度要比软件大；（6）TCP协议的复杂性造成TOE设计难度大，很难定位设计错误。这些缺点弱化了TOE的好处。

目前没有操作系统提供支持TOE的接口或API，对TOE的实现主要在Linux操作系统中，通过修改TCP协议栈来卸载协议，实现方式大致有3种：(1)高性能Socket(High Performance Sockets)[23-24]；(2)屏蔽主机TCP协议栈[25]；(3)替换主机TCP协议栈[26]。TOE产品主要有Chelsio 10G Ethernet TOE[27]、Adaptec TOE[28]和Alacritech SEN 2100系列[29]。Chelsio 10G Ethernet TOE完全卸载了TCP协议栈，包括TCP链接的建立和拆除，而在Kim和Rixner的设计中，将TCP链接的建立和拆除操作保留在主机端，将数据传输卸载到硬件中[30]。

2.1.6 应用级别传输协议

多次数据复制和中断上下文切换给传统TCP协议处理带来很大开销，一些研究人员尝试设计应用级别的传输协议，如Mapp和Pope实现了用户空间传输协议A1[31]，Thekkath等人实现了用户级别传输协议[32]，Edwards和Muir实现了用户级别的TCP协议[33]，Shivam等人实现的EMP系统[34]。

（2） 阅读文献的范围及查阅手段

1) 阅读文献的范围

国内外网络学术会议和期刊论文中相关的论文以及相关的学位论文。XLS416 多核微处理器的相关文档。

2) 查阅手段

借助互联网搜索引擎以及相关文献数据库。

3. 研究内容

（1）研究目标

本课题将使用多核 NPU 作为 NIC，并在 Linux 操作系统中编写 NIC 驱动程序，实现 TCP 方发送数据的分割和接收数据的乱序重组功能卸载。通过减少 CPU 处理报文开销、减少中断数量和减少 DMA 数量三个方面提高主机 TCP 性能。

（2）主要研究内容及拟解决的关键科学问题和技术问题

TCP 数据传输卸载需要驱动程序和 NIC 进行协作完成，本课题中将编写 Linux 操作系统的 NIC 驱动程序，并使用多核 NPU 作为 NIC，进行发送数据的分割和接收数据的乱序重组等操作。本课题的主要工作包括以下两个方面：

1. NIC 驱动程序的设计和实现

网卡驱动程序位于 TCP/IP 协议栈和 NIC 之间，负责将协议栈传递下来的报文交给 NIC 进行发送，以及将 NIC 接收到的报文传入协议栈进行处理。

Skbuff 是 Linux 操作系统中 TCP/IP 协议栈缓存报文的数据结构，在源代码中随处可见，地位重要。Skbuff 有多种缓存报文的存储模型，TCP 数据传输卸载中，驱动程序和 NIC 之间交互大报文（64KB），如何使用 skbuff 存储大报文是需要解决的一个问题。

Skbuff 中的 head 和 end 指针指向低端内存中的一段连续内存，被称为线性区域。当报文较小时，通常报文数据会被放到线性区域中。这种存储模型最为简单，但在 64 位的机器中，使用此模型存储一个 64KB 的大报文需要 16 个连续的页，代价较高。

在协议栈进行 IP 分片重组时，会用到 skbuff 缓存报文的 frag_list 模型，该模型使用 skb_shared_info 中的 frag_list 作为链表头部，通过 skbuff 中的 next 指针将分片数据各自的 skbuff 结构体连接为一个单链表。

当一个大 TCP 报文被分割为多个 MTU 大小的数据包时，会用到 skbuff 缓存报文的另外一种链表模型，它也通过 skbuff 中的 next 指针将分割后的 skbuff 结构体连接为一个单链表，却使用第一个 skbuff 作为链表头部。

Skbuff 另外一种存储报文的模型中将报文数据存储在不同的页中，而在 skb_shared_info 中使用 nr_frags 记录存储数据页的数量，使用 frags 数组记录每个页中的数据长度和偏移量信息。这种模型中使用的物理页可以是分散的，适合作为 TCP 数据传输卸载中存储大报文的方式。

2. 多核 NPU 卡 XLS416 中程序的设计和实现

XLS416 是 Broadcom 公司出品的一款多核 NPU 处理卡，它拥有 4 个 MIPS64 处理核心，16 个硬件处理线程，4GB 的 DDR2 DRAM 存储空间，2 个万兆网口，还有报文分类引擎、压缩解压缩引擎和安全加速引擎。本课题中将使用 XLS416 作为 NIC，进行 TCP 发送报文的分割和接收报文的乱序重组等操作。

TCP 发送报文的分割较为简单，在驱动程序将大报文交给 XLS 卡后，需要逐个修改分割后的报文中 TCP 和 IP 头部中的字段，重新计算校验和，然后将报文发送出去。这里需要避

免数据的大量复制带来的性能降低。

TCP 接收报文的乱序重组较为复杂，首先需要检查数据包的校验和，丢弃校验和错误的报文；其次需要区分设置了 SYN、FIN、RST、URG 等标志的报文和普通的数据报文，并为每条 TCP 链接缓存报文，将接收到的乱序报文重组为 64KB 的大报文，经驱动程序交给协议栈处理。需要解决的问题是 TCP 链接的高效管理，以方便报文到达时进行查找，TCP 乱序报文的重组算法，以及避免数据大量复制，保证性能。

（3）拟采取的研究方法、技术路线、实施方案及可行性分析

本课题研究 TCP 数据传输卸载，研究如如何将原本在主机端 CPU 执行的 TCP 发送数据分割和接收数据乱序重组操作卸载到多核 NPU 中执行。

在系统的设计和实现中，需要大量阅读 Linux 驱动程序、TCP/IP 协议栈、skbuff 等相关资料，找到 skbuff 存储大报文的方法和设计出重组 TCP 乱序报文的算法。

TSO（TCP Segmentation Offload）和 Checksum Offload 已经是当前很多 Linux 网卡具备的特性，只要在网卡驱动中设置相关的标识符，协议栈就不会再分割 TCP 报文，而是将超过 MTU 的 TCP 报文直接交给驱动程序，由网卡将 TCP 数据分割为 MTU 大小的数据包，并计算校验和后发送。

通过阅读 skbuff 的源代码和相关资料，决定使用 skb_shared_info 中 frags 数组，将数据存储在不同页中的方式来存储构造好的大 TCP 报文，交给协议栈处理。

通过阅读 TCP/IP 协议相关资料和考虑各种常见的数据结构，暂定使用哈希表管理 TCP 链接，每条链接上使用链表缓存报文。

暂定 TCP 乱序报文重组算法的流程图如图 1 所示。

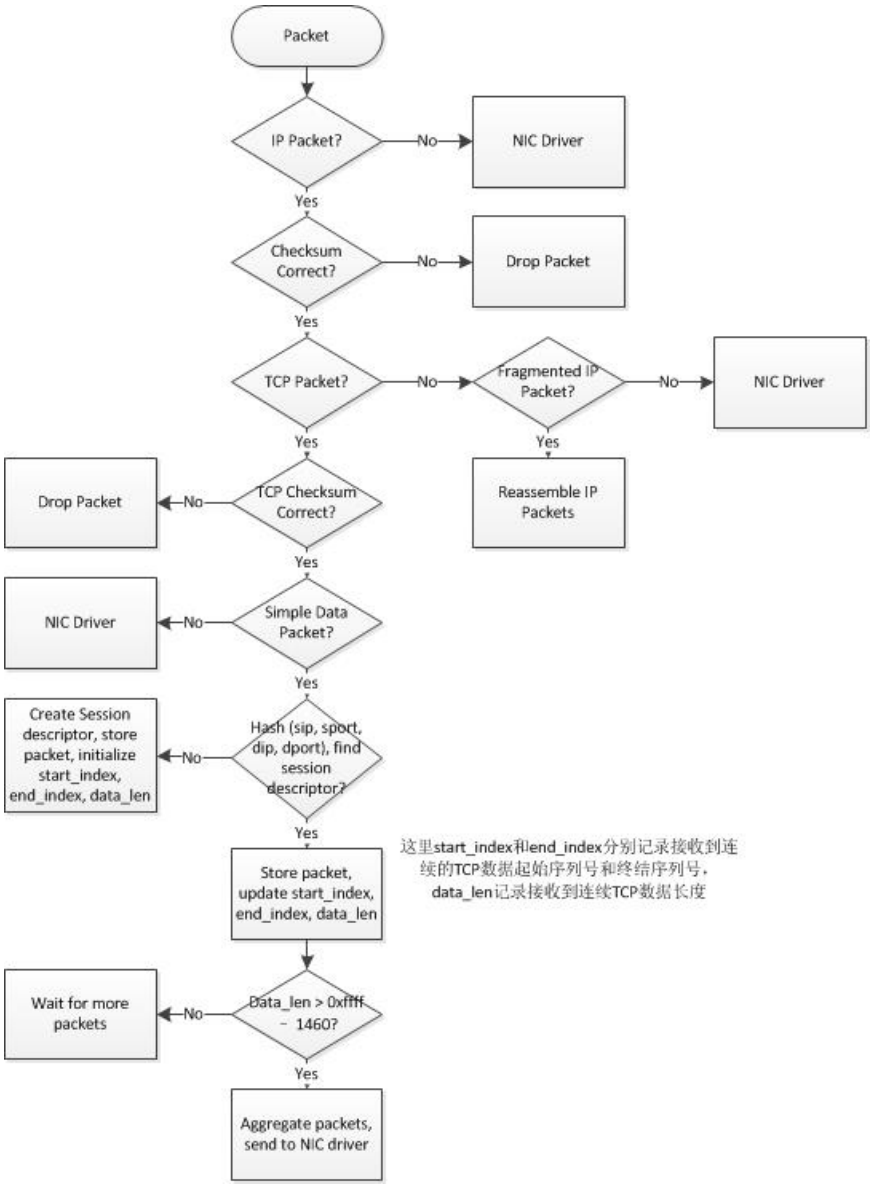


图 1 TCP 乱序报文重组流程

（4）预期创新点

本课题的创新点主要表现在以下几个方面：

（1）不同于传统的基于定制硬件和专用芯片的 TCP 卸载方式，本课题尝试使用商用多核 NPU 提高 TCP 性能，利用多核 NPU 对报文的处理能力，通过增加一级存储，以空间换取时间，达到 TCP 卸载功能。

（2）由 NPU 完成乱序 TCP 报文重组，并合并为一个大 TCP 报文交给驱动程序，进而交由协议栈进行处理，减少 CPU 处理 TCP 报文的开销，减少中断数量，达到提高性能的目的

（3）尝试改进 Linux 中报文缓存数据结构 skbuff，使用 skbuff 存储大块传输数据（64KB），而非单个的报文，以提高 DMA 的性能，优化整体系统；

（4）将 TCP 分割发送数据的操作卸载到作为 NIC 的 NPU 中执行，减少主机开销。

4. 研究条件

1、目前已具备以下研究条件：

1） XLS416 多核 NPU 处理卡，相关编译工具链；

2） 装有 Linux 操作系统的计算机；

3） 高性能应用级测试仪；

2、研究所面临的困难

多核 NPU 编程是一个比传统 X86 平台更为复杂的过程，需要掌握大量的网络与多核编程知识，且每一个过程，如构造大块数据时，如果 skbuff 数据结构稍有错误，将导致协议栈不能正确处理报文。代码中的错误难以被调试和跟踪，运行时发现的 bug 难以重现等困难，将会是研究本课题所面临的主要困难。另外，本课题采用 NPU 进行 TCP 卸载虽然具有创新性，但其结果如何，要完成完整的系统并进行测试才能得到结果，这也是本课题的困难和风险。

5. 学位论文工作计划

起讫日期	主要完成研究内容	预期成果
2014.12-2015.1	阅读相关参考文献和资料	对TCP加速领域有总体认识
2015.2-2015.3	阅读XLS416文档，Linux NIC驱动相关文档	学会Linux NIC驱动和XLS416多核NPU程序编写
2015.4	研究Linux TCP/IP协议栈和skbuff，设计TCP报文乱序重组算法	找到skbuff存储大报文的方法，设计出TCP报文乱序重组算法
2015.5-2015.6	Linux NIC驱动程序和XLS416程序的设计和实现	初步实现系统
2015.7-2015.9	测试系统的TCP性能，找到性能瓶颈并优化	测试、完善系统
2015.10-2015.12	撰写毕业论文，论文预审并准备答辩	完成毕业论文和答辩

注：每个子阶段不得超过3个月；预期成果中必须包含成果的形式、数量、质量等可考性指标该计划将作为论文研究进展检查的依据。

6. 主要参考文献（博士不少于 50 篇、外文不少于 25 篇，硕士不少于 30 篇、外文不少于 15 篇，可附页）

序号	文献目录（作者、题目、刊物名、出版时间、页次）
1	Greg Buzzard, David Jacobson, Milon Mackey, Scott Marovich, John Wilkes. An implementation of the Hamlyn sender-managed interface architecture. Proceedings of the 2nd Symposium on Operating Systems Design and Implementation, New York: ACM Press, 1996.245-259.
2	D. BomaII, Cray Research, C. Panridge. Computing the Intemet Checksum. RFC1071, 1988. http://www.faqs.org/rfcs/rfcl071.html.
3	T. Mallory, A. Kullberg. Incremental Updating of the Internet Checksum. RFC1141, 1990. http://www.faqs.org/rfcs/rfc1141.html.
4	A. Rijsinghani. Computation of the Internet Checksum via Incremental Update. RFC1624, 1994. http://www.faqs.org/rfcs/rfc1624.html.
5	Karl Kleinpaste, Peter Steenkiste, Brian Zill. Software Support for Outboard Buffering and Checksuming. ACM SIGCOMM Computer Communication. 1995, 25(4):87-98.
6	Tomas Henriksson, Niklas Persson, Dake Liu. VLSI IMPLEMENTATION OF INTERNET CHECKSUM CALCULATION FOR 10 GIGABIT ETHERNET. http://www.da.isy.liu.se/pubs/tomhe/DDECS2002_checksum.pdf.
7	Grey R. Wright, W. Richard Stevens. TCP/IP Illustrated, Volume 2: The Implementation. Addison Wesley, 1995.
8	H. K. Jerry Chu, Zero-Copy TCP in Solaris. Proceedings of the USENIX 1996. http://citeseer.ist.psu.edu/chu96zerocopy.html.
9	C. Dalton, G. Watson, D. Banks, C.Calamvokis, A. Edwards, and J.Lumley. Afterburner. IEEE Network, 1993, 7(4):36-43.
10	P. Druschel and L Peterson. Fbufs: A High-Bandwidth Cross-Domain Transfer Facility. Proceedings of the Fourteenth ACM Symposium on Operating Systems Principles, 1993.189-202.
11	Steven H. Rodrigues, Thomas E. Anderson, David E. Culler. High-Performance Local Area Communication With Fast Sockets. In USENIX Annual Technical Conference, 1997. http://citeseer.ist.psu.edu/rodrigues97highperformance.html.
12	P. Culley, D. Garcia, J. Hilland. An RDMA Protocol Specification. IETF Internet-Draft. drift-ietf-rddp-rdmap-00.txt. http://www.ietf.org/proceedings/03mar/I-D. 2003.
13	Murali Rangarajan, Aniruddha Bohra, Kalpana Banerjee, Enrique V. Carrera, Ricardo Bianchini. TCP Servers: Offloading TCP Processing in Internet Servers. Design, Implementation, and Performance. Technical report, Department of Computer Science Rutgers University, 2002. http://citeseer.ist.psu.edu/rangarajan02tcp.html.
14	Jeffrey S. Chase, Andrew J. Gallatin, and Kenneth G. Yocum. End-System Optimizations for High-Speed TCP. IEEE Communications Magazine, 2001, 39(4):68-74.
15	Juan M.. UDP, TCP, and IP Fragmentation Analysis and Its Importance in TOE Devices. 2003. http://mayaweb.upr.clu.edu/crc/crc2003/papers/JuanSola.pdf.

序号	文献目录（作者、题目、刊物名、出版时间、页次）
16	David Mosberger, Larry L. Peterson, Patrick G. Bridges, Sean O'Malley. Analysis of Techniques to Improve Protocol Processing Latency. Proceedings of ACM SIGCOMM `96, ACM, 1996. http://citeseer.ist.psu.edu/mosberger96analysis.html.
17	Addressing TCP/IP Processing Challenges using the IA and IXP Processors. Intel Technology Journal, 2003,7(4):39-50.
18	Eric Yeh, Herman Chao, Venu Mannem, Joe Gervais. Introduction to TCP/IP Offload Engine. 2002. http://www.10gea.org/SP0502IntroToTOE_F.pdf.
19	Anvantages of a TCP/IP Offload ASIC. http://graphics.adaptec.com/pdfs/tcpio_adv_wp.pdf.
20	Tomas Henriksson, Ulf Nordqvist, Dake Liu. Embedded Protocol Processor for Fast and Efficient Packet Reception. 2002. http://www.da.isy.liu.se/pubs/tomhe/ICCD2002.pdf.
21	Boon S. Ang. An Evaluvation of an Attempt at Offloading TCP/IP Protocol Processing onto an i960RN-based iNIC. 2001. http://www.hpl.hp.com/techreports/2001/HPL-2001-8.pdf.
22	Jeffrey C. Mogul. TCP offload is a dumb idea whose time has come. HotOS03, 2003. http://www.usenix.org/events/hotos03/tech/full_papers/mogul/.
23	H. V. Shah, C. Pu, and R. S. Madukkarumukumana. High Performance Sockets and RPC over Virtual Interface (VI) Architecture. In CANPC workshop `99. 1999.
24	J. S. Kim, K. Kim, and S. I. Jung. SOVIA: A User-Level Sockets Layer Over Virtual Interface Architecture. In Cluster Computing `01. 2001.
25	W. Feng, P. Balaji, C. Baron, L. N. Bhuyan, D. K. Panda. Performance Characterization of a 10-Gigabit Ethernet TOE, in Proceedings of HOT Interconnects, August 2005.
26	Adaptec Corporation. Advantages of a TCP/IP offload ASIC. 2004. http://graphics.adaptec.com/pdfs/tcpio_adv_wp.pdf.
27	Ethernet Storage White Papers, http://www.adaptec.com/en-US/products/host_tech/. 2007.
28	10GbE Storage Accelerators, http://www.chelsio.com/products/10G_adapters, 2007.
29	Alacritech SEN2100 Series, http://www.alacritech.com/Products/Network/SEN2100.aspx, 2007.
30	Hyong-youb Kim, Scott Rixner, TCP Offload Through Connection Handoff. Proceedings of the EuroSys Conference(2006): 279-290.
31	Glenford Mapp, Steve Pope. The Design and Implementation of a High-Speed User-Space Transport Protocol. Proceedings of the 1st International Wrokshop on High Performance Protocol Architectures, 1994.
32	Chartdramohan A. Thekkath, Thu D. Nguyen, Evelyn Moyt, Edward D. Lazowska. Implementing Network Protocols at User Level. IEEE/ACM Transactions on Networking, 1993, 1(5):554-563.
33	Aled Edwards, Steve Muir. Experiences implementing a high performance TCP in user-space. In Proc. SIGCOMM. New York: ACM Press, 1995.196-205.
34	Piyush Shivam, Pete Wyckoff, Dhabaleswar Panda. EMP: Zerocopy Osbypass NICdriven Gigabit Ethernet Message Passing. SC2001 November 2001.

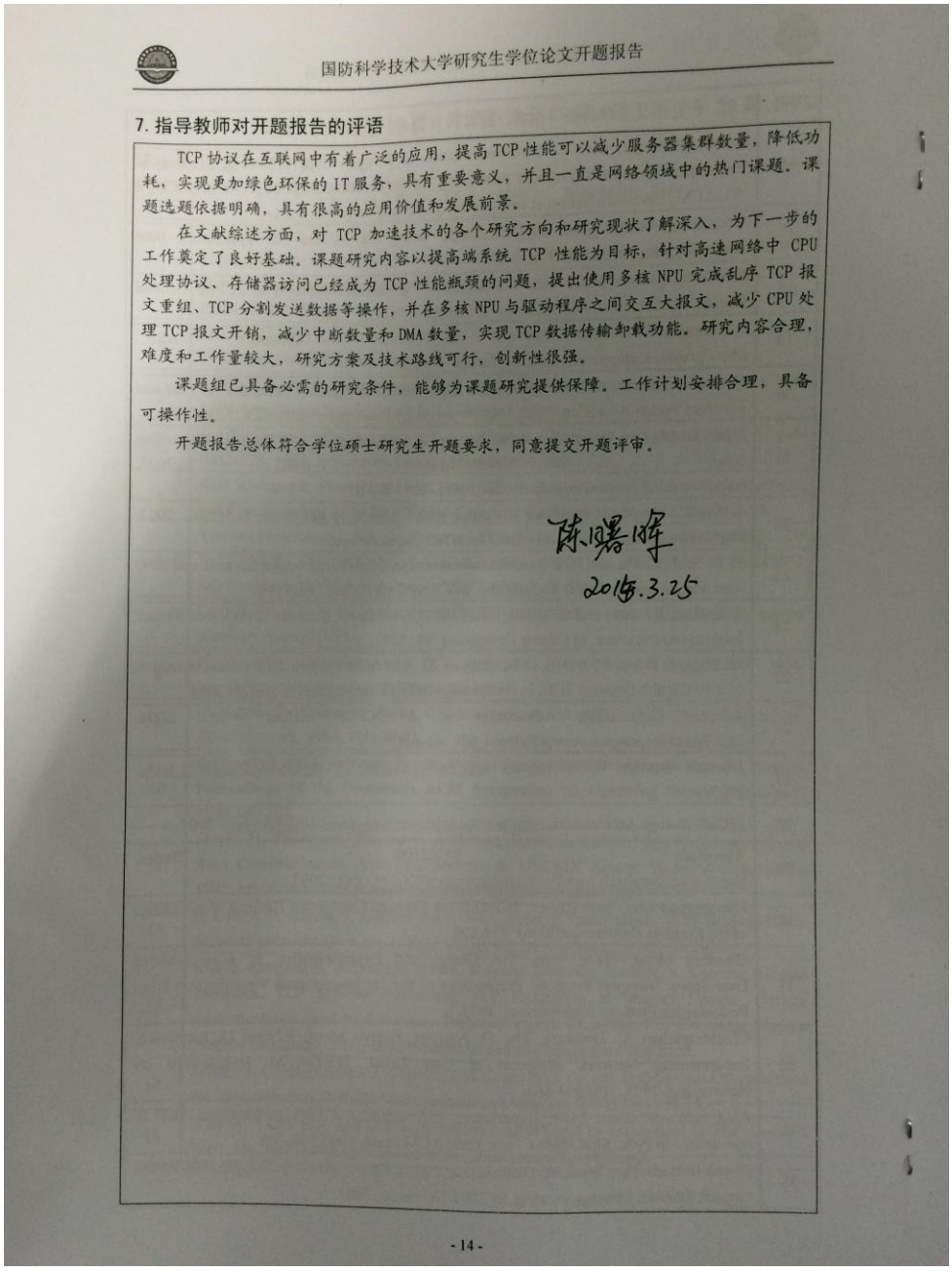
7. 指导教师对开题报告的评语

TCP 协议在互联网中有着广泛的应用，提高 TCP 性能可以减少服务器集群数量，降低功耗，实现更加绿色环保的 IT 服务，具有重要意义，并且一直是网络领域中的热门课题。课题选题依据明确，具有很高的应用价值和发展前景。

在文献综述方面，对 TCP 加速技术的各个研究方向和研究现状了解深入，为下一步的工作奠定了良好基础。课题研究内容以提高端系统 TCP 性能为目标，针对高速网络中 CPU 处理协议、存储器访问已经成为 TCP 性能瓶颈的问题，提出使用多核 NPU 完成乱序 TCP 报文重组、TCP 分割发送数据等操作，并在多核 NPU 与驱动程序之间交互大报文，减少 CPU 处理 TCP 报文开销，减少中断数量和 DMA 数量，实现 TCP 数据传输卸载功能。研究内容合理，难度和工作量较大，研究方案及技术路线可行，创新性很强。

课题组已具备必需的研究条件，能够为课题研究提供保障。工作计划安排合理，具备可操作性。

开题报告总体符合学位硕士研究生开题要求，同意提交开题评审。



国防科学技术大学研究生学位论文开题报告

7. 指导教师对开题报告的评语

TCP 协议在互联网中有着广泛的应用，提高 TCP 性能可以减少服务器集群数量，降低功耗，实现更加绿色环保的 IT 服务，具有重要意义，并且一直是网络领域中的热门课题。课题选题依据明确，具有很高的应用价值和发展前景。

在文献综述方面，对 TCP 加速技术的各个研究方向和研究现状了解深入，为下一步的工作奠定了良好基础。课题研究内容以提高端系统 TCP 性能为目标，针对高速网络中 CPU 处理协议、存储器访问已经成为 TCP 性能瓶颈的问题，提出使用多核 NPU 完成乱序 TCP 报文重组、TCP 分割发送数据等操作，并在多核 NPU 与驱动程序之间交互大报文，减少 CPU 处理 TCP 报文开销，减少中断数量和 DMA 数量，实现 TCP 数据传输卸载功能。研究内容合理，难度和工作量较大，研究方案及技术路线可行，创新性很强。

课题组已具备必需的研究条件，能够为课题研究提供保障。工作计划安排合理，具备可操作性。

开题报告总体符合学位硕士研究生开题要求，同意提交开题评审。

陈曙晖

2016.3.25

- 14 -