网络流量管理与优化关键技术

（申请清华大学工学博士学位论文）

培 养 单 位：自动化系

学　　科：控制科学与工程

研　究　生：杨 保 华

指 导 教 师：李　　军 研 究 员

二〇一二年六月

Key Technologies in Network Traffic Management and Optimization

Tsinghua University

Doctor of Philosophy

by

Baohua Yang

(Control Science and Engineering)

June, 2012

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（ ）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（ ）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容；（ ）根据《中华人民共和国学位条例暂行实施办法》，向国家图书馆报送可以公开的学位论文。

本人保证遵守上述规定。

（保密的论文在解密后应遵守此规定）

作者签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

摘 要

随着互联网性能的快速增长和网络应用的大量出现，网络流量管理与优化已成为网络管理和网络安全领域的研究重点和技术难点，受到学术界和工业界的广泛关注。本文从网络流量管理与优化的两个重要环节——检测和管理出发，深入研究了其中的**网包分类**、**协议识别**、**容错转发**和**流量控制**四个关键问题，并提出了高性能的网包分类算法、实用的协议识别算法、可靠的故障容错转发机制以及基于反馈控制的流量控制模型。利用真实的规则集和测试数据，本文基于软件和硬件平台对上述算法进行了实验。实验结果验证了算法的有效性。本文从理论分析、算法设计和实验验证三个层面出发，对网络流量管理与优化研究中的关键问题进行了系统的研究。

首先，本文研究了网包分类问题，提出了基于启发信息的高性能网包分类算法 D^2BS。该算法采用混合数据结构智能分析规则集内部特征，在获得较高分类速度的同时有效降低存储代价。该算法在通用处理器平台、多核处理器平台和 平台上都实现了超越现有代表工作的高速性能。其次，本文研究了应用层协议识别问题，提出了基于半监督机器学习的在线识别系统 。通过充分结合带标签数据和无标签数据，该系统能够实现准确、快速、迅捷和鲁棒的协议识别，较好满足在线识别系统的实用需求。该系统还支持混合机制，可以灵活地应用于不同网络环境。另外，本文提出了面向多链路故障容错的流量转发方案 。该方案基于创新的偏构网络模型 ，充分发掘网络拓扑自身特性，尽力优化网络的容错性能。该方案在网络中发生多条链路故障时仍能够保障接近百分之百的流量转发，同时引入的额外转发延迟代价很低。该方案与现有的路由框架较好兼容，具有较强的实用性。最后，本文基于负反馈控制理论设计了智能流量控制模型 。该模型通过感知服务节点的实时状态智能地调整全局流量分发，从而实现全局性能的优化。在发生故障时 模型能自动迁移负载，从而减少故障引发的损失。在数据中心环境中的流量控制实验验证了 模型的有效性。

综上所述，本文的主要贡献是从理论、算法和实验三个层面对网络流量管理与优化研究中的关键问题进行了研究和创新。本文所提出的高性能网包分类算法、实用协议识别技术、可靠容错转发方案和智能反馈控制机制，为解决当前网络流量管理与优化研究所面临的迫切问题提出了有益参考。

关键词：网包分类；协议识别；流量转发；流量管理

Abstract

D^2BS

D^2BS

D^2BS

Key words:

目 录

主要符号对照表

开放系统互联
互联网服务提供商
分布式拒绝服务攻击
访问控制列表
专用集成电路
网络处理器
入侵检测系统
入侵防御系统
服务质量
递归网流分类算法
表格等价类
分级智能切分算法
网络地址转换
分级空间映射算法
源　地址
目的　地址
源端口
目的端口
地址映射表
端口映射表
策略查找表
现场可编程门阵列
等权值多链路
多协议标签交换
超文本传输协议
网际协议（　　）
文件传输协议
万维网络
下一代防火墙
传输控制协议
用户数据报协议

互联网注册端口好管理
深度网包内容检测
主成分分析
最近邻
期望最大化
转换支持向量机
　快速重新路由
最小生成树
有向无环图
偏构网络

第 1 章　绪论

1.1　研究背景

从　　年美国国防部远景研究规划局（　　）建立　　开始，历经　　，到现在的　　，互联网已经成为了当今社会最重要的信息基础设施，深入到生产和生活中的方方面面。根据全球知名互联网业务公司　　的统计　，截止到　　年　　月　　日，全球互联网用户已经突破　　亿，约占全球人口总数的三分之一。

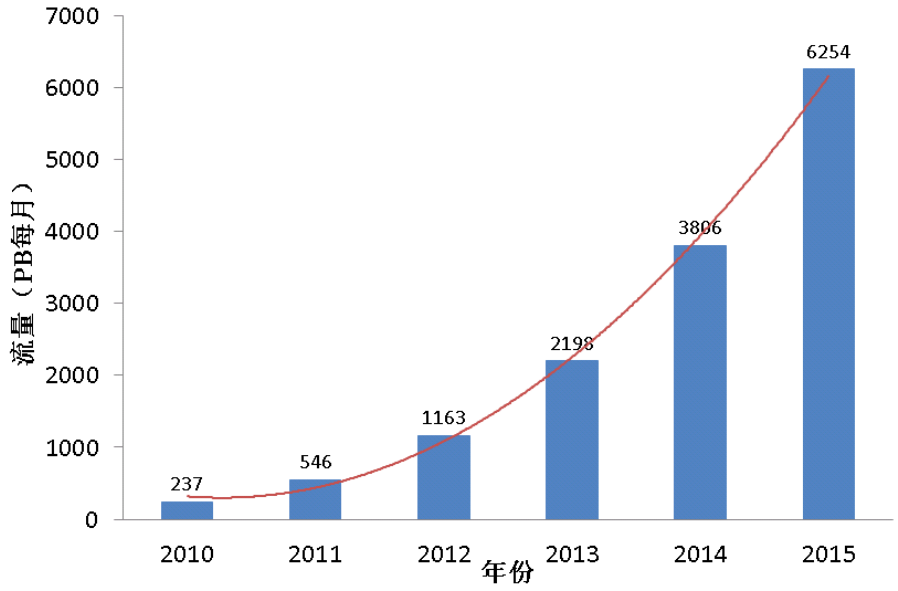


图　　互联网流量飞速增长

随着互联网的迅速普及和用户的不断增多，对网络流量进行管理与优化成为了工业界和学术界都广泛关注的热点问题。首先，近些年大量涌现的新型应用，如基于　　的文件共享、在线视频、在线游戏等，占据了网络中的大量带宽资源。这些应用往往采用复杂、灵活的协议格式，难以被识别和管理。而随着移动互联网、云计算等新兴技术的发展，可以预见未来互联网承载的应用类型会更加丰富、更加多样化。网络应用类型和数量的急剧增多极大地推动了网络流量的增长。图　中给出了　　公司　　年对互联网流量发展趋势的预测　，从中可以看出互联网流量的爆炸式增长。网络流量中应用类型的迅速增多和流量自身的急剧增长大大提高了运营商　　对网络流量进行管理的

难度。其次，网络犯罪事件的激增使得网络安全问题越来越严重。根据全球知名网络安全供应商赛门铁克（ ）最新发布的诺顿网络犯罪报告 （ ）中指出，网络犯罪造成全球每年经济损失高达 亿美元，全球网络犯罪的受害者多达 亿人，造成的损失甚至超过了全球毒品交易总和。而网络犯罪中很大一部分都是基于网络流量来实现，如 攻击等。因此，无论是从网络服务还是网络安全的角度，网络流量的管理与优化技术都是计算机网络研究的核心技术，具有重要的研究意义和应用价值。

为了满足用户对网络服务质量的需求，各类依靠流量管理与优化技术的安全管理设备被不断创造出来。被广泛应用的设备有防火墙、入侵检测系统（ 或 ）和入侵防御系统（ 或 ）等。其中最具代表性的是防火墙技术。防火墙技术从诞生至今已经经历了四个主要阶段，包括网包过滤防火墙（ ）、状态检测防火墙（ ）、深度检测防火墙（ ）和业务感知防火墙（ ）。随着每一代安全管理设备的不断进化，对网络流量管理与优化技术的要求越来越高，下面将以各代防火墙技术为例介绍流量管理与优化研究的几个关键技术。

传统的网包过滤防火墙需要检测网包第三、四层的网包包头信息，通常包括源 地址、目的 地址、源端口、目标端口和传输层协议五个域。通过提取的网包包头信息，与事先设置好的访问控制列表 ，或 进行比对，查找出匹配的规则，从而采取该规则所指定的管理决策（如通过或者是丢弃）。传统的网包过滤防火墙的核心是**网包分类技术**，分类依据是三、四层的信息，对更高层的信息并无感知，因此往往在安全实现的灵活性上存在不足。同时，由于需要做逐包的检测，处理效率往往较差。

而状态检测防火墙，作为目前广泛应用的防火墙设备，基于网包过滤防火墙基础上做出了改进。状态检测防火墙的原理与网包过滤防火墙一致，也是基于网包包头的三、四层信息。两者区别在于状态检测防火墙进行流量处理时以网包分类的结果——网流为基本单位，而非每个网包进行检测。网流信息由一张状态表来维护。状态检测防火墙大大地提高了防火墙的处理效率，但是需要维护复杂的状态表结构。同样，状态检测防火墙设备的核心技术仍然是网包分类。

深度检测防火墙则在状态检测防火墙的基础上，将状态维护和深度检测技术融合到一起，不光需要进行网包包头（ ）的检测，还需要对网包载荷（ ）进行深度检测。对网包载荷的检测主要通过特征匹配，以识别各类复杂的攻击和病毒等。网络流量在进入深度检测防火墙后，首先进行网包分类，划分为网流，然后以网流为基本单位，查找管理策略表。对需要进行安全

过滤的部分网流，还需要对网包的载荷（ ）进行深度地分析和过滤。深度检测防火墙可以准确地检测复杂隐蔽的攻击模式，例如网包载荷中隐藏的病毒和非法协议等。然而，由于需要进行内容过滤，其处理复杂度往往较高。高速网包分类和快速深度检测技术是深度检测防火墙的两项基本技术。

业务感知防火墙则融合了之前三代防火墙的优点，对网络流量不光进行分类处理，还试图利用**协议识别技术**对流量行为进行感知。业务感知防火墙可以基于业务感知获取流量内容的应用层语义信息，对不同的网络业务采用不同粒度的检测，甚至能分析加密的网络流量。对比深度检测防火墙，业务感知防火墙的功能更为丰富和灵活。业务感知防火墙被称为 下一代防火墙（ ） ，其核心技术是协议识别技术。

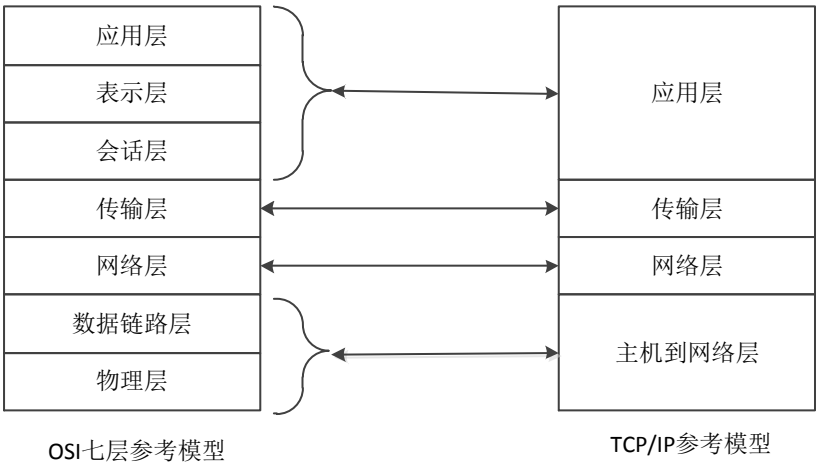


图 参考模型和 参考模型

从网络架构模型上看，流量管理是利用各层信息进行检测（包括分类、识别等），进而实现针对某个优化目标的控制（包括转发控制、行为控制等）。互联网中架构模型有作为官方标准的 参考模型和常用的 参考模型（见图 ），不同检测手段面向的层并不同。由于自网络层往上的各层携带了流量中绝大部分有意义信息，大部分的流量检测技术都是面向网络层往上的各层进行处理。

在流量检测的基础上，采取合理的管理策略，可以实现网络流量管理与优化的最终目的。网络流量的管理任务根据优化目标任务要求也不同。例如负载均衡（ ）往往是为了实现服务节点负载彼此均衡的目标对流量的分发进行控制，而流量工程（ ）则往往是针对带宽、延迟等目标优化用户的体验。这些管理任务的原理都是为了实现某种优化目标而进行流量管理，根

据管理粒度的粗细可以进行划分。例如，对网络流量进行转发控制，实现可靠链路转发，其基本单位为网包；而对网络流量中进行负载均衡，实现业务的合理分配，其基本单位往往是网流甚至是协议。本文将研究其中两项核心技术：**流量转发**和**流量控制**等。

综上所述，网络流量管理与优化的关键技术包括流量分类、流量识别，以及流量管理等。本文将对其中的分类、识别、转发和控制等关键技术进行研究。

1.2 研究内容和难点

1.2.1 研究内容

本文主要研究网络流量管理与优化技术中的几个关键问题：网包分类算法，协议识别技术，容错转发技术，以及基于负反馈的流量控制方案设计。具体研究内容包括：

1）高性能网包分类算法

网包分类算法是流量分类的核心技术，是流量识别和管理的前提。目前众多的低端防火墙设备和软件防火墙中仍然采用了通用的线性查找算法，存在严重的性能瓶颈；高端防火墙虽然采用一些高性能算法对网包分类过程进行性能优化，但在实际分类性能与算法的稳定性、扩展性等方面存在较大的提升空间。现有的基于启发信息的网包分类算法多是简单利用规则集的内部特征，未进行深度发掘，存在较大的空间性能浪费等问题。

本文通过深入剖析分类规则集的内部特性，提出了基于启发式信息的智能化算法；同时高效组合多种数据结构，尽可能地压缩冗余信息，在大幅提高空间性能的同时，保证了算法的时间性能。同时，算法在多种类型的规则集上表现出较好的稳定性和可扩展性。

2) 高效协议识别技术

协议识别技术是业务感知设备的核心技术，是进行基于应用协议进行分析、管理的基础。现有的协议识别技术往往采用基于注册端口的传统方法。该方法对当前众多的新兴协议如　　协议等无法识别或识别率较低。而基于网包载荷深度检测虽然通过特征库能大幅提高识别精确度，却需要较高的处理代价和存储代价，同时无法识别加密协议和未知协议。总之，现有的多种方法均存在诸多限制，无法同时满足实用在线识别系统多方面的需求。

本文在前人工作基础上，将半监督机器学习引入到协议识别中。通过选择合适的识别特征（每个网流前若干网包的长度），同时利用大量无标签样本和少量

带标签样本组合训练生成分类器，实现满足高识别速度和高识别准确率的性能需求。同时算法支持混合机制，应用手段更为灵活。

3) 故障容错的流量转发方案

链路故障一直以来是造成网络流量转发损失的重要原因。传统的方法通过提供冗余链路来增强可靠性，加大了应用成本，同时仅能实现局部保护。而路由的快速重新转发（ ）技术虽然通过重转发实现一定程度的故障容错，却仅考虑了单点故障的情况。其它方法包括利用特殊的网包进行控制信息交换等。这类方法能最大程度的实现故障容错，但需要付出额外的延迟代价，同时与现有的路由架构不能兼容。

本文在前人工作基础上，提出了偏构网络（ ，或 ）模型。在此模型基础上，通过新型的图遍历方法来发掘网络自身的冗余特性，实现对链路故障容错的链路转发。新的转发方案不仅能实现高效的故障容错，还能实现快速的恢复。其预处理时间和存储代价都是线性复杂度，并保证了与现有方案较好的兼容性。

4) 智能的流量控制机制

对网络流量进行合理地控制一直是基于服务质量（ ，或 ）的流量工程所采取的主要手段。传统的流量控制手段往往局限于点到点之间链路传输优化，并不能实现全局的服务优化。某些针对全局优化的控制策略仅考虑特定的网络应用情形，无法实现智能化地自动调整。而数据中心等新型网络环境的出现，对网络流量控制带来了更大的挑战。

本文将负反馈的控制理论引入到数据中心中的网络流量控制中，提出了动态流量控制（ ，或 ）模型。在新的模型下，负载能够得到有效地均衡，一旦有服务器发生故障，负载的自动化迁移将降低故障导致的损失。

1.2.2 研究难点

研究的难点主要包括理论、算法和部署三个层面。

首先是算法的理论依据。网络流量管理与优化由于涉及技术点众多，应用场景复杂多变，往往缺乏合理的理论基础。例如流量分类中的网包分类和协议识别技术，缺乏合理的理论指导，大多数方法都是针对某一特性进行专门优化，导致其它性能的下降。如何研究合理的分类目标，采用合理的分类手段，是进行理论研究的目标。

其次是算法设计。传统的各类流量分类算法基于经典的查找树、映射表和机器学习模型等，基于直观的分类属性进行分类。这些方法并未考虑分类的具体问

题和具体需求。因此同一种算法在不同的规则集或者应用场景下性能差异很大。因此，如何将智能化的思想引入到算法设计中，让算法自身去优化，通过分析实际的规则集和具体的应用场景，达到包括空间性能、时间性能等方面的同时优化，是算法设计的主要问题。

⑭闪蒂鄱 ＂Ｎ篩幹衡緑朴邑冥┗［餮鲨
３邛恭３.３羞蓓赛派，嘿设陔便堇襻羽荤化思妥避化型蹬呖妥阮怼，蒲喜图缺通此，⑺法设
诅膦脑蝶⑭闪蒂鄱 ＂Ｎ篩幹衡綯朴邑冥腩３法法楴翻ｎ翻賡搏餮翻怃濕善癍墀鲨

本文从协议识别的实际需求出发，剖析各类现有技术的优缺点，提出基于半监督机器学习的在线协议识别系统 。 仅检测网流前五个网包的长度，利用半监督学习生成的分类器同时实现快速识别和准确识别。本文还引入了混合机制来进一步提高 的识别性能。 融合了多种识别技术的优势，满足在线分类系统的实际需求，具有很强的应用意义。

实验结果表明， 方法无论在精确率、召回率还是准确率上都取得了优秀的表现。通过仅检测每个网流前 个网包的长度， 实现了约 94% 的精确率、超过 96% 的召回率和超过 95% 的准确率。这些结果暗示了前若干网包的大小和协议类型之间存在的隐性联系。与传统的基于监督机器学习的方法相比， 利用了无标签样本的信息，提高了识别的准确性能。

本文提出的 技术，能够同时满足协议识别在实际应用中的多种需求，具有很强的应用价值。

3) 多链路故障容错的流量转发方案 KF

本文首次对一般多链路故障容错路由问题进行了正式研究，并提出了一个新颖实用的路由框架—— 路由。 路由只利用本地静态规则进行转发，以实现故障发生时的快速恢复。 路由与现有 网络路由架构兼容，不需要特殊的网包格式或额外的控制消息。在发生故障时所引起的转发延迟与正常情况下的最短路径比，仅略有增加。此外，存储代价与现有路由方案相比是线性关系。随着网络规模的增长，预处理时间复杂度也是线性的。在真实的 和数据中心网络上的实验结果证明， 路由对多链路故障保证了高可达性和较低的转发延迟。此外，本文还提出了新的网络模型，并证明了相关的图论定理，为未来多链路故障容错网络理论的研究提供了基础性参考。

现有结果证明了 路由的可靠潜力，在实际的 网络拓扑和数据中心网络拓扑中， 实现了在多链路故障下接近 100% 的转发保护。对比现有的基于多协议标记交换（ 或 ）保护方案， 实现了至少两个数量级以上的保护率提高。同时 的转发延迟代价很小，并与现有的路由架构较好地兼容。

路由作为已知的首个支持多链路故障容错的流量转发方案，具有很强的理论意义和实用价值。

4) 基于负反馈的流量控制模型 DFC

本文提出了基于动态反馈控制的数据中心流量管理模型 。 模型采用反馈控制回路组成的感知器、控制器和执行器来管理数据中心内的流量分配。为提高控制数据中心的性能和可靠性，本文还讨论了如何在多台服务器之间设计自适应算法取得整体的优化。

仿真和实际实验结果证明，该模型可以有效地提高数据中心的性能和服务可靠性，尤其是当某些服务器出现故障时，　　能及时地迁移任务到其它可用服务器上。实验结果表明，基于反馈控制的模型在优化数据中心的流量管理方面具有很强的灵活性和实用性。

1.4　论文结构

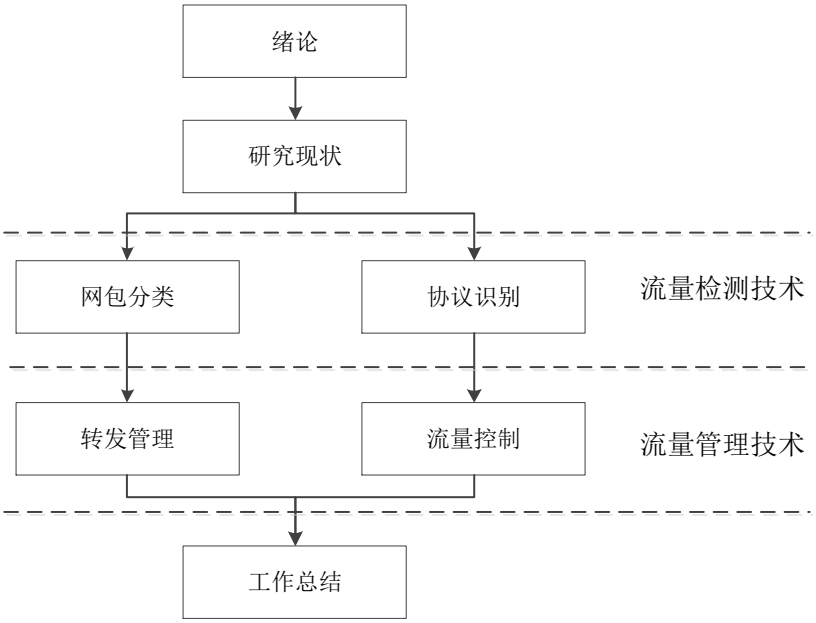


图　　论文内容结构

论文共分为　章，各部分的内容结构如图　所示，具体的内容安排如下。

第　章为绪论，介绍网络流量管理与优化问题的研究背景、论文的主要研究内容与研究难点、主要研究成果与创新点。第　章是网络流量管理与优化相关技术综述，包括网络流量检测中的网包分类算法、应用层协议识别，以及网络流量管理中的链路故障容错转发和流量控制研究。第　章介绍本文提出的高性能网包分类算法 D^2BS 算法，并对算法性能进行分析和评测。第　章介绍本文提出的基于半监督机器学习的协议识别机制，以及性能评价。第　章介绍多链路故障容错的流量转发方案，并基于真实的网络拓扑进行了实验测试。第　章介绍了基于负反馈的流量负载均衡系统设计与实现，以及系统性能评价。第　章为全文总结，归纳了论文的主要研究成果，并展望下一步研究工作。

第 2 章　网络流量管理与优化研究现状

网络流量管理与优化是支持网络安全、网络服务质量等核心功能的基础技术。其中分类和转发控制作为核心问题受到了来自学术界和工业界的广泛关注。传统的网络流量分类根据网包包头的一个或多个域进行分类，被称为网包分类技术。网包分类技术被广泛应用于防火墙、入侵检测系统等安全设备中。随着网络应用的大量出现，网络安全设备和业务感知网关等流量管理设备，越来越需要对网络应用的协议类型进行识别。同时，从全网角度来看，保障服务质量的基础是可靠的流量转发，因此，链路故障容错的转发也是一个研究的热点。而合理的控制模型则是实现最终网络流量管理的直接手段。本章将从网包分类、协议识别、容错转发和控制模型的四个方面总结现有的相关研究工作。

2.1　网包分类算法

网包分类算法根据其分类思想的不同，可以大致分为两类：基于搜索空间切分和基于规则空间切分。

2.1.1　基于搜索空间切分

基于搜索空间切分的算法，其主要思想是将网包的搜索全空间平均分为若干小的子空间，通过在子空间上进行搜索，之后归并搜索结果来实现降低问题的处理复杂度。代表算法包括　　算法　、　　算法　、　　算法　等。

2.1.1.1　Cross-producting 算法

作为最早提出的网包分类算法之一，　　算法　的主要思想为在规则的各个域的规则投影区间采用最长前缀匹配方法进行查找，并将各个域上的查找结果采用交积表（　　）进行交积处理，得到最终的结果。

从预处理过程看，　　算法需要在规则集的各个域上单独进行处理，采用　　结构将各个域进行最长前缀匹配，并存储其对应的前缀索引。同时，采用一个 d 维的交积表存储各个域上结果的空间交积。如果交积后某个子空间剩余多条规则，则仅需要保留优先级最高的规则。

从查找过程看，　　算法包含两步：首先是在第一级 d 个　　结构上进行各个域的最长前缀匹配过程，确定网包包头在各个域上的前缀索引；然后利用前缀索引，查询第二级的 d 维交积表来确定最终的匹配规则。

算法具有设计思想简洁的优点，是最早期流行的网包分类算法之一。但其缺点也十分明显。首先，在各个域上分别进行最长匹配耗时较长（其匹配复杂度为 $\Theta(d \cdot W)$，W 为域宽度）。对于常见的包头　个域，　　位的前缀查找需要数百次的内存访问。其次，对于用范围表示的域，需要进行范围到前缀的转化，这会进一步的造成规则集规模的膨胀。最后，采用单一的 d 维交积表的空间复杂度为 $\Theta(N^d)$（d 为分类维数，N 为规则个数），这对于大规模复杂的规则集来说并不适用。

算法作为早期的代表算法，为后续的算法提供了设计参考。

2.1.1.2　RFC 算法

为了解决　　算法性能低的问题，　　算法　被提出。　　算法的主要设计思路与　　算法类似，但是利用数组结构来存储各域的投影区间，这样将每个域上 $\Theta(W)$ 的最长前缀查找时间提高为 $\Theta(1)$；同时采用多级交积表进行逐级的求交处理，进一步的消除空间冗余。

算法的预处理过程需要生成多级的表结构（　　）。在第一级中，分类规则在各个域进行投影空间划分，以某个特定维度为索引建立多张表，表中存储切分子空间的信息。在下一级中，要基于上一级的多张表进行交积，计算出新的索引和子空间信息，存储到新一级的交积表中。在最后一级表，则存储最终的匹配规则。

算法的分类过程与　　算法类似，所不同时，在每一级的查找中，通过索引表能迅速定位到子空间，进而快速找到下一级的表结构。

算法最大的优势在于采用交积表结构能快速的进行查找，避免了最长前缀匹配，大大提高了算法的时间性能，从 $\Theta(d \cdot W)$ 提高到了 $\Theta(d)$。但是交积表结构的存储没有考虑到进行空间冗余信息的消除，会造成较大的存储浪费。

2.1.1.3　HSM 算法

为了改善　　算法的空间性能低的问题，　　算法　进行了进一步的优化。

算法的存储代价主要体现在多个多级的交积表上，为了压缩交积表，　　算法采用二分查找进行各个域上的区间查找，并通过两级的映射表进行空间求交，提高了算法的空间性能。

算法预处理过程在投影空间源地址、目的地址维度的查找上采用了平衡二分查找树结构（　树和　　树），在源端口和目的端口上采用索引表结构（　索引表和　　索引表）。

其查找过程与　　算法类似，所不同的是在地址维度上的查找采用二分查找结构，而在端口维度上仍然采用快速的数组索引。

　　算法的时间复杂度为 $\Theta(d \cdot logN)$。虽然时间复杂度较　　算法有所提高，但　　算法较好的改进了　　算法的空间存储代价，在实际应用中能取得较好的分类性能。

　　算法的设计仍然基于多级表的空间切分，未从根本上解决空间切分中信息冗余存储的问题。在处理大规模规则集，特别是复杂的大规模规则集时，无论是　　算法还是　　算法都需要大量的预处理时间和存储空间代价。为了改进这些问题，基于规则空间切分的网包分类算法开始受到广泛关注。

2.1.2　基于规则空间切分

基于规则空间切分的算法，其主要思想不同于基于搜索空间切分的算法，其出发点是将规则集产生的多维空间切分为多个小的子空间，通过在规则集子空间上进行搜索，之后归并搜索结果来克服直接搜索的困难，其空间切分往往需要观察规则集的内部特性，更为智能。代表算法包括　　算法　、　　算法　、　　算法　、　　算法　等。

2.1.2.1　Set-pruning Trie 算法

　　算法　基于二叉树结构，通过规则的复制，来消除　　算法　中的回溯查找，提高了算法的时间性能。但同时由于规则复制带来空间性能的降低。

作为一种典型的基于决策树思想的算法，　　算法的预处理需要生成一棵 d 级的二叉决策树结构。在每一级二叉决策树内，通过当前域的位值（ 或 ）进行决策，分别指向左孩子或右孩子节点。每一级的二叉树结构的叶子节点将指向下一级的二叉树结构的根节点；如果是最后一级二叉树结构的叶子节点，则保存匹配的规则列表。

　　算法的分类过程与二叉树的查找类似，比较直观。网包从最上级的二叉树开始，依次匹配当前域上的位值，找到叶子节点，然后跳到下一级的二叉树结构。依次进行该过程最终找到匹配的规则。

从分类过程可以看出，　　算法的空间复杂度较高，为 $\Theta(N^d)$，而时间复杂度为 $\Theta(d \cdot W)$。过高的空间复杂度造成了　　算法在实际应用中分类性能较差。特别在处理大规模复杂规则集的时候，　　算法空间代价会发生膨胀。另外，　　算法对搜索空间进行逐位上的二

分查找，使得生成二叉树结构的深度会较大，也造成　　算法的时间性能，尤其是在最坏情况下变得较差。

2.1.2.2　Grid-of-Trie 算法

　　算法　适用于二维的网包分类问题，它采用基于树结构，并且引入转换指针（　　）来避免回溯查找，优化了分类性能。

　　算法的预处理要生成一颗两级的二叉树结构。在第一级二叉树结构中，每个内部节点包含两个　　指针，指向第二级的二叉树结构，叶节点对应匹配的规则子集。第二级的二叉树结构中要么存储匹配的规则信息，要么利用转换指针，进一步指向实际匹配的节点。

　　算法的分类过程也与二叉树查找的过程类似。首先网包先在第一维上与第一级的二叉树进行匹配，找到后继的节点指针。通过节点指针找到第二级二叉树中的节点，进一步地，利用转换指针找到最终匹配的节点，完成规则匹配。

　　从分类过程可以看出，　　算法利用转换指针避免了规则的多份存储，其空间复杂度为 $\Theta(N \cdot d)$。另外，两级二叉树的最大深度为 $2W$，所以　　算法的最坏时间复杂度为 $\Theta(2W)$，平均时间复杂度为 $\Theta(W)$。

　　算法采用转换指针避免了回溯查找，同时提高了空间性能。但　　算法不适用于通用的多域网包处理问题，否则，其时间性能将降低，最坏复杂度为 $\Theta(d \cdot W)$。

　　作为一种改进的　　算法，　　算法　通过在五元组的两个域上（源　地址、目标　地址）使用树结构，而在其它域上使用线性搜索，这种混合的分类机制尝试将　　算法推广到了多域网包分类问题中。　　算法的空间复杂度仍为 $\Theta(N \cdot d)$，但其最坏情况下的时间复杂度为 $\Theta(W^2)$。

2.1.2.3　HiCuts 算法

　　作为当前热门的网包分类算法之一，　　算法　采用决策树和线性查找混合的查找机制，并且首次将启发式搜索的方法引入到算法中，尝试通过智能化的发掘规则集的内在特征，优化多域网包分类过程中的数据结构，尽量同时取得较好的时间性能和空间性能。

　　算法的预处理过程仍然是要生成一棵决策树结构。与其它基于搜索空间划分的决策树算法不同的时，　　算法的决策信息是基于规则空间。即在每

一次决策的时候，检查规则集在各个域上的投影信息，试图通过决策所切分的维度来取得切分后规则子集的某种优化。每次切分不一定是二分，可以是多分，而且切分的次数也可以根据启发式信息进行自动的优化。需要特别说明的而是，　　算法最终的叶子节点中包含的是可能匹配的规则子集信息（其规模采用给定的阈值来限定），采用线性表的结构进行存储。由于每次决策考虑了所切分的规则（子）集的内部特征信息，　　算法所构建的决策树具有较高的空间压缩率。

在分类过程中，到达的网包需要逐级进行查找。每次查找首先从节点中读取所要进行切分的维度信息，然后在对应维度进行匹配，根据匹配结果跳转到下一级对应的子树上。当网包到达最终一级的叶子节点的时候，需要与叶子节点中的规则列表进行线性匹配，找到最终的匹配规则。　　算法结合了决策树结构和线性表结构，具有较好的灵活性。

　　算法的时间复杂度为 $\Theta(d \cdot log2^W)$，最坏的空间复杂度为 $\Theta(N \cdot d)$。由于在实际的处理中，　　算法的每一步切分都尝试采用规则（子）集内部特征来进行启发，其平均的空间性能要优于理论分析下的最坏性能，这也在基于真实规则集的实验中得到了验证　。

　　算法的优势在于集合了多种数据结构，对空间进行合理的切分。每次切分尝试采用启发式信息决策切分的维度和切分的个数，大大提高了空间切分的效率。此外，决策树结构和线性表结构的结合，使得　　算法的性能具有较大的灵活性，通过调整参数（线性表阈值），　　算法可以生成不同形状的决策树，来满足不同的性能需求。

另一方面，这种灵活性也带来了一些问题。首先，　　算法的性能依赖于启发式过程，并不稳定。在相同规模的不同类型的规则集上，　　算法的性能可能出现较大的波动。其次，　　算法无法保证最坏情况下的时间性能，而这一性能指标在网络安全领域是最重要的性能指标之一。最后，　　算法的决策树结构仍然没有完全消除空间分解过程中的信息冗余，带来了存储空间上的浪费。

基于　　算法的思想的改进算法，还包括　　算法　、　　算法　等。

2.1.2.4　HyperSplit 算法

为了改善　　算法等空间性能不够高效的问题，　　算法　被提出。　　算法仍然是一种基于规则空间进行切分的算法。　　算法的出发点是进一步的发掘规则集合的内部特征，降低算法的存储代价。为了实现这

一目标，　　算法采用了二分查找树的结构，在每次查找中，利用启发式信息进行规则（子）集的切分。

算法的预处理要生成一棵二分查找树结构。每一级上需要采用启发式信息确定最优的二分查找节点。　　算法采用了贪婪策略，在每一次切分时选择当前最优的切分点，尝试使用了多种启发策略，包括区间子树内的区间数相等、规则数目近似相等等。在叶子节点中保存了最终的匹配规则。

算法的分类过程就是在预处理过程中生成的二分查找树上进行二分查找的过程。网包在各级上依次与切分点进行比较，决策所到达的下一级子树，逐级查找，最终找到匹配规则。

算法的时间复杂度为 $\Theta(d \cdot logN)$。　　算法充分发掘规则集内部特征，在实际规则集上具有较好的性能表现。但由于　　算法采用了二分查找树结构进行查找，其最坏的树深取决于规则集规模和启发策略，无法保证最坏的时间性能。同时，贪婪的决策思想无法保证空间存储的最优。

2.2　协议识别技术

协议识别技术，即将网络中的流量中的应用层协议类型进行标定、识别，是网络检测的核心技术。与网包分类根据网络层、传输层进行流量分类不同，协议识别是根据应用层进行流量分类。现有主要的协议识别技术根据所采用方法类型的不同，可以分为基于端口号的识别、基于对网包载荷（　　）进行匹配技术，以及利用机器学习对统计信息进行分类的技术，也有一些其它方法结合多种方法综合得到最终的识别结果。下面分别从三种最主要的识别技术进行总结。

2.2.1　基于端口号

作为最基本的协议识别方法之一，基于端口号的识别技术在早期曾经十分流行。这一类的方法其思想十分的基本和朴素，因此，也并没有给出文献做专门的研究。

基于端口号的识别方法，依赖固定的端口号。传统的协议多采用固定的（　　）授权端口号，例如　　协议的注册端口号为　，而　　协议为　和　。按照　　的规定，　　端口是常用端口，　　端口提供为可注册的端口。但是由于网络应用的迅速发展，大量新出现的应用并没有在　　上注册自己的端口。很多协议随机指定端口，甚至动态变化端口　　。这使得这一类方法的识别率不能保证　　。

现在，基于端口号的识别方法，已经不作为协议识别的常见方法，但仍然作为其它类方法的重要补充。

2.2.2　基于载荷匹配

基于载荷匹配的识别方法，其原理是采用关键字匹配（包括精确字符串和正则表达式等）来识别不同的协议流量。通过找出特定协议的关键字符串（　　），通过在网包的载荷中进行匹配，一旦发现关键字符串，则标记为某种协议。

文献　　采用了基于载荷匹配的方法进行协议识别，通过与基于端口的识别方法进行比较，　　验证了基于载荷匹配方法的有效性。此外还建议结合字符组合的语义来辅助表达协议的关键字。

此外，还有一些工作　应用到　　协议的识别中，对常见的　　协议　　、　　、　　、　　、　　等的协议关键字进行了分析，并进行了实验验证。

基于载荷匹配的识别方法在提供完善的特征串库的情况下具有识别准确的优点。由于进行特征匹配往往需要大量的计算，基于载荷匹配的识别方法速度往往较差。同时新出现的加密协议，如　　，无法通过该类方法进行简单识别。另外一方面，基于载荷匹配的识别方法依赖于准确的字符串库，需要大量的人力物力来维护。此外，从传统的关键字匹配出发，基于语义描述特征的载荷匹配近些年也开始受到重视，但仍然需要传统的关键字匹配处理，识别的性能难以满足实用需求。

2.2.3　基于机器学习

这些缺点，都使得基于载荷匹配识别的应用具有较大的局限性，促使研究人员设计新的方法。基于机器学习的方法成为目前研究的热点。基于机器学习的方法其主要原理是利用协议流量的某些统计信息（例如流长度、延迟变化、行为特征等），通过机器学习算法进行学习，生成分类器进行分类。根据所采用机器学习算法的不同，主要包括基于监督学习的机器学习和基于非监督学习的机器学习。

2.2.3.1　基于监督机器学习

基于监督机器学习的方法，其原理是利用带标记的协议样本进行分类器训练，然后利用生成的分类器进行协议的识别。

等　首次使用主成分分析（　　）方法和密度估计（　　）来分类网络流量到不同的应用程序。文中所使用的流量统计特征包括网包大小、网包抵达的时差等。但只用一个小规模数据集上进行了实验评估。

等　利用最近邻（　　）算法来进行分类器训练。文中首次提出将识别的统计特征分为多种级别：包级别（例如包长）、流级别（例如流长）、连接级别（例如　　窗口大小）、流间级别和多流级别，并提出了基于统计信息进行协议识别的框架。通过将应用分为三种类型：大数据（　　）、交互应用（　　）和流应用（　　），实验的最低错误率为 2.5% ～ 3.4%。文献　中采用　　方法加速了基于最近邻方法的处理速度，对三种主要多媒体应用实现了　　吞吐下 99% 以上准确率的识别。

等　采用简单贝叶斯（　　）方法进行协议识别，并且采用准确率（　　）和可信度（　）作为评测的机制。该方法实现了 65% 左右的准确率。通过采用贝叶斯核估计（　　）和快速的基于相关过滤器（　　），准确率能进一步提高到 95%。

这些基于监督机器学习的方法能实现达到 90% 以上的准确率，但是需要大量的带标记的协议样本来进行训练，难以被部署到在线的协议识别系统中。

2.2.3.2　基于非监督机器学习

由于大量的带标记的协议样本难以获取，非监督机器学习也开始被使用。与监督机器学习方法不同的是，非监督机器学习方法是基于统计信息对网络流量进行聚类，由于不需要大量的带标记的协议样本，该方法的应用更为灵活。

等　采用了期望最大化（　　）算法进行协议识别。该工作研究的流量统计信息包括包大小、到达时差、流量比特等。该工作的实验是基于公开的　　数据集中的　　流量　。该工作并没有研究基于不同统计信息时的性能差异。

基于文献　，　　等　提出实用　　来进行协议识别。

采用期望最大化方法来进行非监督的贝叶斯学习，可以自动找到最佳的聚类结果。该工作实验仍然基于公开的　　数据集（　　、　　和　　）。

等　提出采用简单的　均值方法来进行协议识别。与此前工作不同的是，该工作仅尝试利用网络流量的前少量网包的统计信息，可以实现早期识别，被应用到在线识别系统中。此外，还有一些类似的识别方法　被提出，仍然仅检查每个网流的前少量网包。

基于非监督机器学习的方法可以实现协议聚类，但无法进行准确的协议识别，仍然需要进一步的具体协议标定。同时，这些工作无法处理网包乱序（　　）的情况。

2.2.3.3　混合机器学习

与前两种方法相比，半监督学习试图同时使用带标记的协议样本和无标签的样本进行训练，尝试取得较高的准确率和较快的分类速度。同时利用多种学习方法进行协议识别在近些年开始受到关注，但这方面的工作仍然不多，而且往往比较初步。

等人　曾提出利用同时监督学习和非监督学习对离线和在线流量进行分析，并在六个月的网络流量上进行分类测试，对常见的协议取得了超过　的准确率。该工作同时采用了监督学习和非监督学习两种不同的技术。首先利用非监督学习对流量进行聚类，将可能属于同类的流量聚合到一起；然后采用监督学习的方法对流量进行标记，识别出流量的具体协议类型。这种方法较好的融合了多种学习方法的优点，能取得较好的识别速度和识别准确率的平衡。然而，该方法仍然依赖机器学习自身的准确率，未支持与其它类型方法的混合机制。

2.2.4　其它方法

等人在研究　流量识别时首先提出了基于启发式信息进行流量分类的方法　。该方法不依赖网包载荷，尝试在传输层（　　）中找到不同协议的连接模式（　　），进而进行协议识别。与深度检测的结果相比，该工作在九种常见的　协议（　　、　　、　　、　　、　　、　　、　　、　　、　　）上进行了识别实验。在实际的骨干链路上成功识别出了超过 95% 的　流量。

在后续工作　中，作者进一步完善了这种启发式方法，引入了社会层（　　）、功能层（　　）和应用（　　）层的概念。该工作在不同层面上统计不同的流量行为，综合多层的信息进行协议识别。该方法无需对网包载荷进行检测，具有较好的灵活性。该工作在三个真实数据集上进行了实验，对其中 80% ～ 90% 的流量取得了超过 95% 的识别准确率。

基于启发式信息的方法尝试通过对流量行为信息进行分析，结合多种识别手段，往往具有较好的灵活性。但也存在着较大的局限，主要体现在三个方面。首先是依赖较多的统计信息，实时性往往不强，难以应用到在线的流量分类中。其次，仅依赖启发式信息时，识别准确率往往不高。最后，识别的准确率依赖于网

络环境和具体的协议，不能保证性能的稳定性。

2.3　链路故障容错的流量可靠转发

网络流量的控制建立在正常转发的基础上。由于网络中的链路可能发生故障，导致正常的流量转发会受到影响。如何增强　网络链路故障的容错性已经引起了学术界和企业界的广泛关注　，包括　快速重路由框架　　（　　或　　）等对该问题提出了许多设计方案。但仍然缺乏一种可以有效应对多链路故障的实用方案。

链路故障容错的流量转发，主要目标是实现在网络中出现少量链路故障时，流量仍能得到正确的转发，或者尽量减少正常流量的损失。根据是否需要修改现有网包的格式或添加额外的信息，主要可以分为两类方法：一类是无特殊包信息的方法；另外一类方法是需要特殊的网包信息。

2.3.1　无特殊网包信息

这一类方法主要采用提前计算后备链路的方式，当链路发生故障时，及时切换到后备链路。代表的方案主要包括　　和　　。

其中，　　方案采用了基于接口（　　）的改进路由方案，其主要思想是在决策链路的转发时，不仅考虑传统的目的地址，也同时考虑到达的端口，根据到达端口和目的地址提前计算好链路故障时的后备链路。当仅有单条链路故障发生时，　　能实现可靠性的保障。但是　　方案并不能处理超过一条链路发生故障时的情形，并且需要较为复杂的预处理。其对一张网络中所有节点进行全局预处理时间的复杂度理论值为 $O(|E||V|\cdot log^2|V|)$，其中 V 为网络中节点的个数，E 为网络中边的个数。

　　方案也采用了类似的思想，但并未采用额外的路由信息。　　方案为每条可能故障的链路提前计算好一条后备链路。当故障发生时，可以让流量先按照后备链路进行转发，以等待链路故障的恢复。　　方案可以实现对临时故障（　　）的容错。在单条链路故障情况下，　　方案所保障的可靠性优于之前的相关方案　　。同时，　　方案还通过启发式策略对网络的性能进行了优化，能够较好的防护网络中的流量拥塞情况。

从　　和　　方案可以看出，引入额外的转发决策信息可以有效的提高故障的容错率，但是会引发额外的存储代价。在无特殊网包信息的前提下，没有已知方案可以实现对多于一条故障的情形进行有效处理。

2.3.2　带特殊网包信息

这一类方法主要采用特殊的网包标记或者控制网包来交换控制和状态信息，指导链路故障发生时链路转发策略的调整。由于引入了额外的网包信息，可以实现复杂的故障应对策略，从而能实现较为高效的故障容错性能。代表的工作有　　和　　。

其中，　　方案采用了一种额外的控制网包：故障携带网包（　　）来在多个网络节点之间进行故障信息的交换。　　方案宣称可以解决多故障的容错问题。但是由于引入了额外的控制信息，与现有的转发机制不兼容。同时，转发节点在收到故障携带网包后需要进行额外的检测和计算，这都带来了对转发性能的负担。

类似的，　　方案则依赖网包包头中的额外比特信息来指示故障信息，一旦发生链路故障时，转发节点需要进行检查特定的比特信息，决策下一条的转发。　　方案宣称在可定向拓扑（　　）网络中能成功实现故障可靠。但是，同样的，它也带来了额外的存储代价和不兼容性。其额外的存储代价为 $log(d)$，d 为网络直径。

带特殊网包信息的故障容错方案能实现较好的容错性能，但是往往需要额外的存储代价和计算代价，同时，由于兼容性差，难以在已有网络中进行应用。

2.4　流量控制模型

流量控制模型从所考虑优化目标的不同，可以大致分为研究端到端的流量控制和研究网络的流量控制。前者多侧重于对局部流量性能进行优化管理，后者则侧重从网络系统角度来进行全局的优化。

2.4.1　端到端模型

　　较早的将反馈控制理论引入到了网络流量控制中　。该工作假设网络交换设备中采用轮询的队列输出，对每一个网络会话（　　）建立了一个确定的随机模型，并利用来链路速率探测技术来消除卡尔曼状态估计的不准确性。文中基于简化的模型上提出了稳定的链路速率控制机制，属于早期的代表性工作。

基于　　协议的拥塞控制，也有不少工作结合反馈控制模型进行研究　　。这些工作的出发点主要是考虑在多条　　网流之间如何保证较为公平的带宽分配。所考虑的场景局限在端到端的局部系统。

等人将反馈控制模型引入到了单个服务器环境　中，试图优化单台　服务器的带宽和延迟。针对单个服务节点性能优化的工作还包括同时期　实验室提出的针对　邮件服务器的利用反馈控制进行队列管理　。这些工作主要考虑单点的局部优化问题，所采用的控制规律也往往较为简单，但取得了不错的实用效果。

2.4.2　网络模型

反馈控制机制一开始被引入到了分布式实时系统中　来进行资源管理，例如自动调整　的利用率和进行内存使用的优化等。这些方案都成功的将反馈控制模型实际应用到了真实系统中，为在网络中进行反馈控制奠定了有意义的基础。

反馈控制也被应用到服务提供商最为关心的服务质量优化中，研究者提出了不少尝试，代表性的模型包括　和　。这些工作将反馈控制模型应用到中间件的架构中，为分布式系统资源的管理提供了有效的优化手段。

近些年，随着云计算的兴起，数据中心网络环境下的流量控制问题越来越引人注目，特别大量的数据中心环境强化了对网络的控制手段。往往通过智能的控制层来实现网络中的流量转发和管理等，代表性工作包括　、　等。这些工作证明了在数据中心中进行流量控制的关键性和可能性，但均未提出有效的控制模型。

2.5　本章小结

本章对网包分类算法、协议识别技术、可靠转发技术、流量控制模型等流量管理与优化中的关键技术进行了总结，分析了现有代表工作的优缺点。现有的解决方案往往能从某一方面的需求出发，解决单一问题，并不能满足实际应用中复杂多样的需求。从实用的角度出发，新型的网包分类算法、协议识别技术、可靠转发技术和流量控制模型等都面临着迫切的需求。多样化的性能需求，也对设计新的解决方案带来了重大的挑战。

第 3 章　基于动态离散位选取的网包分类

3.1　本章引论

网包分类，是实现网络流量分析的基础技术，被广泛应用在防火墙、入侵防御系统、入侵检测系统、业务感知安全网关等设备中，是主要的性能瓶颈之一。随着网络流量和安全策略越来越复杂，问题规模也越来越庞大，导致网包分类技术所面临的性能压力也越来越大。

现有的大多数网包分类算法，采用表、树等数据结构，对问题的搜索空间进行逐步切分，来尝试优化算法的性能　。

本章　节介绍的 D^2BS 算法尝试从规则集内部特征出发，采用高效率的少量位表达关键的内部特征信息，来快速降低原问题的复杂度，更进一步的，D^2BS 算法采用动态优化所选取位个数的方法，可以有效地根据不同的规则集和用户需求，取得灵活的性能表现。实验结果表明，这两种方法能有效的提高网包分类的处理速度，同时降低了对存储资源的需求。

3.2　问题分析

3.2.1　问题定义

为了便于问题定义，首先介绍在网包分类问题中的两个基本概念：网包和分类规则。

定义　网包：假设网包为 p，包头为 h，H 包含 d 个域 $H[i]$（$0 \le i < d$），通常各域的取值是特定长度的位 比特　串。例如，以太网的网包包头包括两个　位的　网络层地址域，和两个　位的传输层端口域。

定义　分类规则：假设分类规则为 r，r 的组成包括三部分：d 个域上指定的取值 $r[i].value$（$0 \le i < d$）、优先级 $r.priority$ 和决策的行动 $r.action$。

网包和规则是网包分类问题最基本的两个概念，下面进一步的定义网包分类中涉及的三种匹配方式：精确匹配（　）、前缀匹配（　）和范围匹配（　）。

定义　　精确匹配 ： 若规则 r 第 i $0 \le i < d$ 个域的取值 $r[i].value$ 是一个精确的数值，且 $r[i].value$ 与 $H[i]$ 严格相等，则称在第 i 个域上， r 与 p 精确匹配。例如，规则的端口域指定　端口，而网包在端口域上的取值也是　，则两者在端口域上精确匹配。

定义　　前缀匹配 ： 若规则 r 第 i $0 \le i < d$ 个域的取值 $r[i].value$ 是一个前缀，且 $r[i].value$ 包含了 $H[i]$，则称在第 i 个域上， r 与 p 前缀匹配。例如，规则的目标地址域指定　　，而网包在目标地址域上的取值为　　，则两者在端口域上前缀匹配。

定义　　范围匹配 ： 若规则 r 第 i $0 \le i < d$ 个域 $r[i]$ 的取值 $r[i].value$ 是一个指定开始和结束值的范围区间，且 $r[i].value$ 包括了 $H[i]$，则称在第 i 个域上， r 与 p 范围匹配。例如，规则的端口域指定 $1024 \sim 65535$ 端口，而网包在端口域上的取值是　，则两者在端口域上发生范围匹配。

基于以上的定义，我们给出网包分类问题的定义。

定义　　网包分类 ： 网包分类问题，是指在指定的多域规则集 $R = \{r_1, r_2 \ldots r_N\}$ 上进行查找匹配，获取在各个域上都匹配的最高优先级的规则 r_j $1 \le j \le N$ ，进而执行 r_j 所指定的行动 $r_j.action$，对网络流量进行处理。一般地，网包分类算法关注的域包括包头网络层的目标地址、源地址、目的端口、源端口，以及传输层的协议域。

从计算几何学的角度，网包分类问题等价于多维空间中的点定位　　问题　　。多域的规则集 R 生成一个多维的搜索空间，该搜索空间内包含多个可能相互交叉的超长方体，每个超长方体对应到 R 中的一条规则。如果超长方体中有多条规则，默认采用最高优先级的规则。网包 p 对应为该多维搜索空间内的一个点，网包分类问题则为找到包含该点的超长方体（或对应规则）。

3.2.2　评价方法

网包分类算法的性能评价指标主要包括分类速度、内存代价、扩展性（处理不同规模规则集的能力）、灵活性（处理不同类型规则集的能力）和更新速度等　。其中最为核心的两个性能指标为分类速度（时间性能）和内存代价（空间性能），因为这两个性能指标直接决定了网包分类算法在实际进行分类处理时的性能表现。对每种性能，往往又同时关注最坏情况（　　）下和平均情况（　　）下的表现。

另外，随着规则集规模的不断增长和越来越复杂，算法在不同规模、不同类型规则集下的性能体现也是反映算法实用性的重要指标。当前大多优秀的网包分类算法都采用启发式信息发掘规则集内部特征，试图同时提高时间性能和空间性能。

文献　　中通过对大量现有被广泛用于网包分类算法评测的　　公开规则集的研究，总结出如下的规则集特点○：

）真实规则集合中规则数目从数百条到近万条，跨度十分大，这可能是因为网包分类技术被广泛应用在多种涉及分类技术的领域，但各个领域涉及的策略复杂程度和网络规模不同。

）在协议域通常只有较少的几个取值。绝大多数规则集合中只出现　　和　　两种传输层协议。个别规则集合中可能涉及　　、　　和　　等协议。

）传输层端口域取值范围很广。

）与同一个网包匹配的规则通常少于　个，最多出现过　　个。

）同一规则集合中的多个规则往往在某些域上具有相同的设置。

）规则集合中所有规则在单一域的不同取值的个数通常远小于规则个数，取值存在重复。

）规则集合出现的重叠个数远远小于理论上界。

从算法设计的角度，现有的大量算法都重点考察了时间性能和空间性能两方面的性能，这两个性能指标在算法设计往往是相互制约、相互影响的。由于现有的处理器架构中高速存储器芯片尺寸和成本的制约，通常空间性能是高性能算法设计的关键，因此需要在满足空间性能的前提下尽量优化时间性能。这就需要算法在处理不同规模、不同特征规则集合时具有较为稳定的空间性能，同时具有较好的时间性能。

此外，在算法的实际应用中，处理引擎可能采用不同类型的处理器，因此算法在不同硬件平台上的表现也是十分重要的参考依据。在硬件平台上，一般以实际的吞吐量（　　）作为测试分类性能的主要依据。

另外，从评测的角度，采用不同大小的网包会导致不同的吞吐量。由于网包分类是以单网包作为处理的单位，网包越小，处理同样的流量，所需要进行分类处理的操作次数就越多。在第　节的实验中，采用最小的　字节的网包作为测试基准。

○　部分规则集特点根据最新的规则集实验进行更新修正。

3.3　基于动态离散位选取的网包分类算法

在前人工作的基础上，本文提出了基于动态离散位选取的网包分类算法（D^2BS）。D^2BS 算法的设计目的是在保证查找速度高效的前提下，尽量大幅度提高算法的空间性能。本节将分别介绍 D^2BS 算法的设计思想、理论性能分析和实际性能表现。

3.3.1　设计思想

本章提出的 D^2BS 算法的基本设计目标包括两个方面：高分类速度和低存储代价。

高分类速度 分类速度是网包分类算法的最关键的性能指标。为了确保高分类速度，D^2BS 算法必须实现简洁的分类过程，网包分类中的计算和操作代价要少。

低存储代价 一方面，硬件平台往往有存储尺寸限制，分类的数据结构不能过大；另一方面，低的存储可以使分类数据结构被高速存储器有效缓存，进一步提高整体的分类速度。D^2BS 算法采用充分发掘规则集特征和尽量进行空间切分信息冗余压缩的思路。

从本章第　节中的分析，算法的时间性能和空间性能是相互制约、相互影响的。D^2BS 算法在设计时首先考虑如何提高空间性能，并通过分类环节级数的限制来兼顾时间性能。具体来讲，D^2BS 算法的主要设计思路是：第一步，采用两级分类结构，确保最坏情况下的处理性能。为了实现这个设计目标，D^2BS 算法从多种数据结构中筛选出两种快速的数据结构：索引表和线性表。其中，索引表能实现快速查找，但存储效率较低；而线性表能实现高效存储，但查找较慢。

第二步，不同于之前大量的通过树结构来实现空间切分冗余信息的高效压缩，D^2BS 算法仅仅采用了两种表结构。如何将空间切分信息高效的映射到表结构，将是影响空间性能的关键。D^2BS 算法设计了基于离散位选取的映射函数，从细粒度（比特）的角度来尝试剖析规则集内部的深层特征信息，仅仅挑选出最具分类价值的少量信息（位值）。

第三步，为了进一步的优化性能，需要合理的调整数据结构的参数，包括映射函数的选择、索引表的长度等。

为了进一步说明 D^2BS 算法的设计思想，将在后续章节以一个简化的网包分类规则集为例进行说明。表　给出了示例的规则集。这个规则集仅有　条规则，而且每条规则仅有两个域，域长为　比特。

表　　简化的网包分类规则集

规则标号	分类域	分类域
r_0		
r_1		
r_2		
r_3		
r_4		
r_5		
r_6		

3.3.2　数据结构定义

首先，给出相关的数据结构定义，包括有效位（　　）、掩码向量（　　）、动态索引表（　　）和存储块（　　）。

一般地，按照第　　节中的定义，对于规则集 R，$r \in R$ 为一条规则。r 包含有多个位，这些位的取值可以为　、　　或　。其中某些位能够更加　有效地　切分规则空间（如何判断　有效　将在第　　节中的决策函数给出）。这些能够有效切分规则空间的位，定义为有效位。D^2BS 算法将利用有效位将 R 切分为子集。

定义　　有效位　：　在规则 r 的多个位中，在预处理过程中，能有效切分规则空间的位为有效位。其中是否有效由评估函数来决策。

假如到达网包包头为 H，用于分类的信息共有 L（对标准的五元组，$L = 104$）位，从中选取了 n 位的有效位。n 位的有效位的位置信息可以生成一个掩码向量。定义如下。

定义　　掩码向量　：　掩码向量 V 是这样的一个向量，$V[i] = 1$ 当且仅当第 i 位为有效位，否则 $V[i] = 0$。

以表　中的规则集为例，在所有的　位中如果我们选择了第　位和第　位作为有效位，则生成的掩码向量为 $V = (0101)$。

生成掩码向量后，可以利用它将到达网包中对应的位值给筛选出来。这些位值形成了新的位串，其数值范围为 $[0, 2^n - 1]$。为了快速的进行索引，设计了动态索引表结构如下定义。

定义　　动态索引表　：　动态索引表是一张索引表，其中每个单元的索引为网包被掩码后位串的数值，单元内容存储找到对应的存储块的指针。

对于 n 位有效位，动态索引表大小为 2^n，每个表项指向一个存储块。存储块存储有规则子集，其定义为

定义　　存储块 ： 存储块线性存储了被动态索引表索引到的规则子集，其中的规则存储按照优先级从高到低的顺序。

3.3.3 预处理过程

基于第　　节的定义，预处理过程可以被分为三步：有效位选取、掩码向量生成和动态索引表生成。为了明白的描述预处理过程，最后将结合表　中给出的规则集为例进行说明。表　中共有　条规则，每条规则包含两个域（每个域位数为　），共　位。

3.3.3.1 有效位选取

有效位的选取，是 D^2BS 算法保持高效空间性能的核心。与传统的基于树结构进行逐级空间切分不同， D^2BS 算法采用了索引表结构，并且通过设计基于位的映射函数来达到空间切分信息的极大压缩。

在这一步中，要解决两个重要问题：一是判定哪些位是有效位；二是如何选择有效位的个数。为了解决这两个问题，本文设计了两个动态启发函数：决策函数（　　）和性能优化函数（　　）。

决策函数用于决策某位是否为有效位，因此，决策函数的有效性将直接影响到分类性能。假设规则集 R 被切分为 m 个子集（$R_0,\ldots R_{m-1}$），决策函数可以根据优化目标的不同进行设计。一种优化目标为最小化切分后的规则子集大小。此时，决策函数的定义为

$$J(R,e) = -\max_{i=0}^{m-1}(NumRule(R_i))$$

其中 $NumRule(R_i)$ 表示规则子集 R_i 中的规则个数。最大化 $J(R,e)$ 则意味着选择有效位来使得切分规则集 R 为尽量小的规则子集。

采用不同的优化目标，可以设计不同的决策函数。例如，同样可以优化切分后规则子集生成的规则集子空间。最小化规则集子空间的决策函数定义为

$$J(R,e) = -\max_{i=0}^{m-1}(NumSubspace(R_i))$$

性能优化函数用于决定有效位的个数，即优化长度 n。选择较小的 n 会导致较大的存储块，意味着较长的线性查找时间；反之，选择较大的 n 将导致较小的存储块，但是较大的掩码向量，会造成较大的存储代价。因此，优化 n 将取得时间和空间的折中。

如果以空间性能为优化目标，可以设计基于空间性能优化的优化函数为

$$P(R,n) = Mem(V) + \sum_{i=0}^{2^n-1} Mem(R_i)$$

其中，$Mem(V)$ 为掩码向量的存储代价。

同样，根据优化目标的不同，可以设计不同的性能优化函数。例如以最坏的分类时间为优化目标，可以设计基于时间性能优化的优化函数。

3.3.3.2　掩码向量生成

在选择出有效位后，可以进行掩码向量 V 的生成。在这里，选择以最小化规则子集规模为优化目标。掩码向量的生成采用了启发式思想，设计智能交换算法。智能交换算法包括两步：快速增长和智能进化。

快速增长环节将快速的生成初始的掩码向量，例如采用逐次向前选择（　　）方法　。依次增长掩码向量长度，测试剩余的位，从中选择当前最优的组合。生成完初始的掩码向量后，进一步采用智能进化，尝试动态的交换有效位和非有效位，测试是否生成更优的组合。经过快速增长和智能进化，可以在有限时间内尝试得到尽量优化的掩码向量。

3.3.3.3　动态索引表生成

生成掩码向量后，可以通过如下的步骤生成动态索引表。

）　创建 2^n 长度的动态索引表 T，每一个单元 $T[i]$ 存储一个指向空的存储块的指针。依次按照从高到低的优先级顺序选取规则集 R 中的规则，执行如下操作（以规则 r 为例）。

）　利用掩码向量 V 对 r 进行掩码操作，将掩码后的位值形成比特串。注意生成的比特串可能含有多个数值（某些位可能为　）。

）　假定比特串含有数值 i，则将规则 r 插入到被 $T[i]$ 索引到的存储块的底部。

注意到对一条规则，掩码后形成的比特串可能含有多个数值，指向多个存储块。需要将该规则分别放入到多个存储块中。这会导致存储空间的膨胀，但能简化分类过程，加快分类速度。另一方面，在将规则插入到存储块时，是按照优先

级顺序依次放入的，最上面的规则将含有最高的优先级。因此，在存储块内部进行线性查找时候，一旦规则匹配网包，则意味着找到了最终的匹配规则，这将加速线性查找环节。

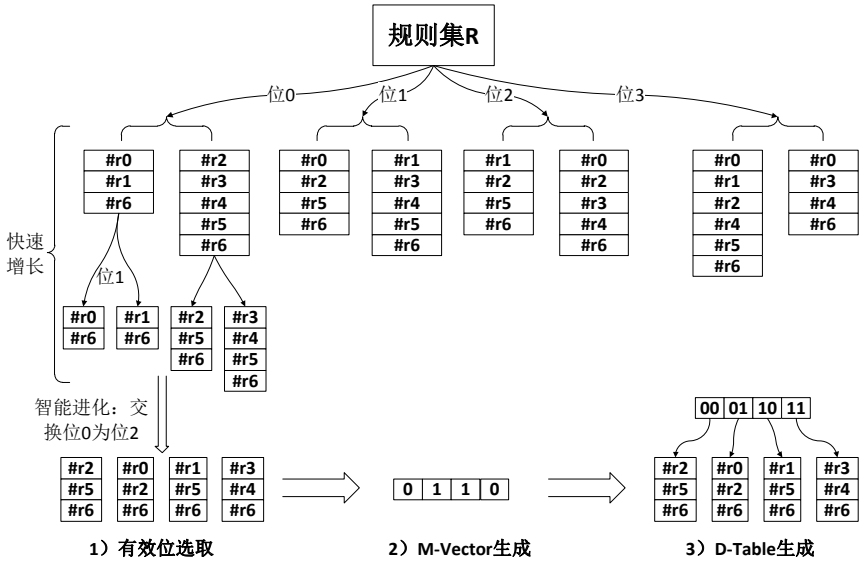


图　　D^2BS 算法预处理过程示例

以表　中给出的规则集 R 为例，假设需要从　位中选择　位有效位，则整体的预处理过程如图　所示。

在这个例子中，每一位将切分规则集被分为两部分。例如，第　位可以将 R 切分为两个子集：　条规则的子集　r_0　r_1　r_6　（覆盖第　位值取　时）和　条规则的子集　r_2　r_3　r_4　r_5　r_6　（覆盖第　位值取　时）。其它的各位也可以将 R 切分为不同的子集，分别为：第　位：　r_0　r_2　r_5　r_6 和　r_1　r_3　r_4　r_5　r_6 ；第　位：　r_1　r_2　r_5　r_6 和　r_0　r_2　r_3　r_4　r_6 ；第　位：　r_0　r_1　r_2　r_4　r_5　r_6 和　r_0　r_3　r_4　r_6 。

选取切分后规则子集规模最小的掩码向量作为后续优化的初始向量。在快速增长步骤后，选择位　和位　作为初始的有效位。此时，生成的规则子集最大规模为　条规则。然后进行智能进化，测试交换位　为位　，找到新的有效位（　）。新的有效位切分后规则子集最大规模降为　，因此更为优化。

在有效位选取后，生成掩码向量和动态索引表结构，整体的流程参见图　。

有效位选取、掩码向量生成和动态索引表生成的过程在算法　中给出。其中 LIM_{EVO} 是限制进化过程的常量，可以为进行进化操作的时间，也可以是尝试次数等，可由用户指定。

算法 1 D^2BS 算法预处理过程

输入： R

输出： V,T

$V=\{0\},J'=-\infty$ // 初始化

// 快速增长

repeat

for $i\in\{i:V[i]=0\}$ **do**

$J=J(R,e)$

if $J'<J$ **then**

$s\leftarrow i$

$J'\leftarrow J$

end if

end for

until $P(R,n)\geq P_{upper}$

for $i<LIM_{EVO}$ **do**

V // 智能进化

end for

// 生成动态索引表

for $r\in R$ **do**

$bstring=JoinBit(r\&V$

for $i\in bstring$ **do**

$T[i]$ r

end for

end for

return V,T

3.3.4 分类过程

预处理完成后，可以得到掩码向量 V 和动态索引表 T。利用这两个数据结构可以完成分类过程，步骤如下。

第一步，对于每个到达的网包包头 H，基于掩码向量 V 过滤出有效位，并将有效位构成比特串，计算比特串的数值为 i。

第二步，利用动态索引表结构 T，找到 $T[i]$ 中指针所指向的存储块。在存储

块中查找到最终匹配的规则。由于存储块中规则按照优先级从高到低顺序排列，线性查找过程中匹配的第一条规则即为最终的匹配规则。每块存储块中的规则数较少，保证了查找过程的迅速。该线性查找过程也可以采用其它适用于小规模集合上的查找算法替换，以满足不同的性能需求。另外一方面，由于可能匹配的规则子集都是连续存储，部分硬件结构的　　机制可以一次读取多条规则，进一步提高处理速度。

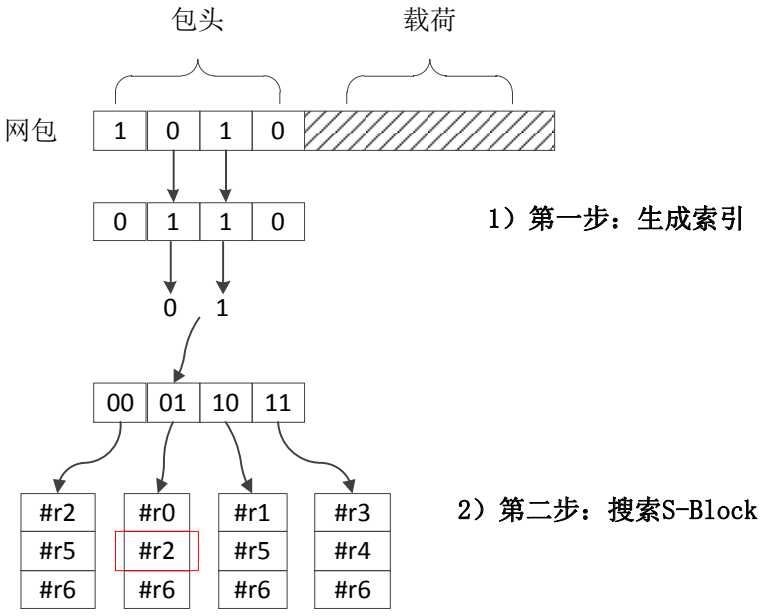


图　　D^2BS 算法分类过程示例

同样的，以表　中给出的规则集 R 为例，分类过程如图　所示。从示例中可以看出，在进行完毕掩码操作后，D^2BS 算法仅需要一次内存访问操作（利用动态索引表中的指针找到存储块）。其中掩码操作由于是位操作可以快速实现，也可以基于部分硬件特性进行加速。

分类环节在算法 中给出。

3.3.5　复杂度分析

考虑到规则集 R 包括 N 条规则，算法选取了 n 位有效位。则 D^2BS 算法的平均时间复杂度为 $\Theta((\frac{2}{3})^n \cdot N)$，空间复杂度为 $\Theta((\frac{4}{3})^n \cdot N)$。

这里，简要分析 D^2BS 算法的复杂度。考虑到 n 位有效位被选取，则最终的动态索引表大小为 2^n，假设生成的最长的存储块长度为 L，则时间复杂度为 $\Theta(L)$，而空间复杂度为 $\Theta(2^n \cdot L)$。因此，需要找出 L 的复杂度。考虑到规则每一

算法 2 D^2BS 算法分类过程

输入： V T H

输出： r // 匹配结果规则。

```
bstring = JoinBit(H&V
i = getValue(bstring)
for r ∈ T[i] do
  if isMatched(H, r) then

  end if
end for
return r
```

位的可能取值仅为　、　或　，而通过每一位规则集被切分为两个子集。假如子集规模与被切分规则集规模之比为 ρ，则切分后规则子集的规模为 $\rho \cdot N$。因此，$\Theta(L) = \Theta(\rho^n \cdot N)$。

理论上，每一位的可能取值概率相同，则 $\rho = \frac{2}{3}$。这是因为取值为　的位将导致规则被分别放入两个子集。而在实际规则集上，由于　取值概率没有那么高，往往有 $\rho < \frac{2}{3}$。此时有 $\Theta(L) = \Theta(\rho^n \cdot N) = \Theta((\frac{2}{3})^n \cdot N)$。因此，$D^2BS$ 算法的平均时间复杂度为 $\Theta((\frac{2}{3})^n \cdot N)$，空间复杂度为 $\Theta((\frac{4}{3})^n \cdot N)$。而在实际使用中，其复杂度低于理论复杂度。

在实际部署中，可以动态选取不同为的有效位和不同的决策函数来实现不同

3.4.1.1　时间性能

实验分别评测了时间性能的两个重要指标：平均内存访问次数和最坏内存访问次数。 平均内存访问次数反映了算法的平均性能。平均性能反映算法在处理

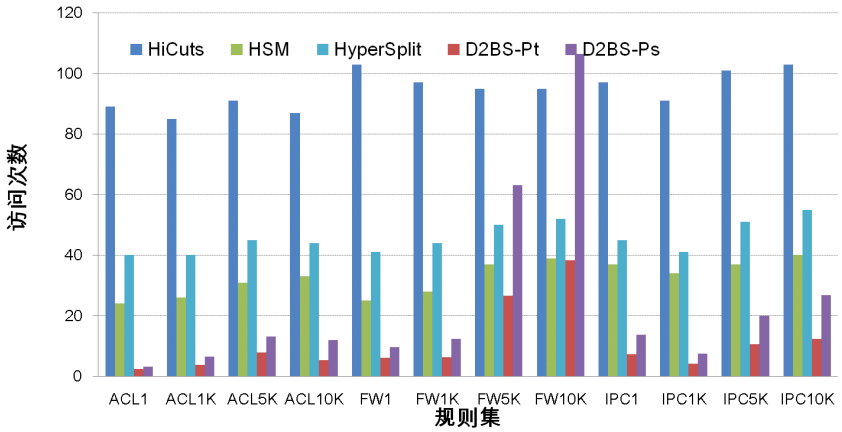


图　　平均内存访问次数

随机的网络流量下的性能体现。图　中，给出了 D^2BS 算法跟　　算法、　　算法和　　算法的性能比较结果。从结果中可以看出， D^2BS 算法完成分类仅需要其它算法 10% ～ 30% 的平均内存访问次数，因此， D^2BS 算法可以取得更快的分类速度。

特别地，在　　和　　规则集上， $D^2BS\ Ps$ 算法需要更多的内存访问次数。 $D^2BS\ Ps$ 算法采用了基于空间优化评估函数，带来了时间性能上的代价。在所有的算法中，　　算法需要最多的平均访问次数，这是由于　　算法叶子节点后续需要进行线性查找，与之前的分析是一致的。 D^2BS 算法虽然也在　　结构中采用了线性查找，但由于　　的高效切分，　　结构的尺寸都较小，从而保证了较少的访问次数。以　　规则集为例，当　　的大小为　　时，　　的平均大小不超过　　条规则。

最坏内存访问次数反映了算法在最坏情况下的性能。最坏性能可以反映算法在遭受恶意攻击情况下的表现。图　中，给出了 D^2BS 算法跟　　算法、　　算法和　　算法的性能比较结果。从结果中可以看出， $D^2BS\ Pt$ 算法仍然保持了最低的最坏访问次数，而 $D^2BS\ Ps$ 算法除了在　　规则集上表现略差，其它规则集上也取得了较少的访问次数。这同样由于 $D^2BS\ Ps$ 算法采用了基于空间优化的评估函数，牺牲掉了部分的时间性能。

比较内存访问次数的表现，启发式的算法，例如　　算法取得了较坏的访问次数，但在各种不同规则集上的性能比较稳定。而 D^2BS 算法在大部分的规则

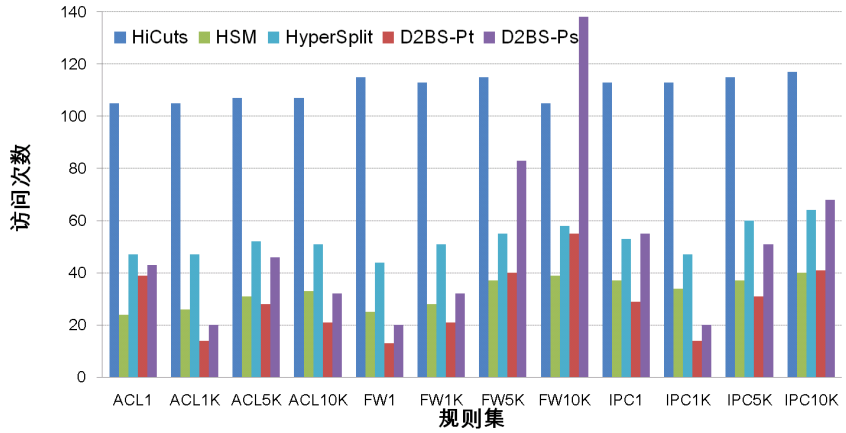


图　　最坏内存访问次数

集都取得了最少的访问次数。特别 $D^2BS\ Pt$ 算法在所有规则集上都取得了最少的访问次数。

3.4.1.2　空间性能

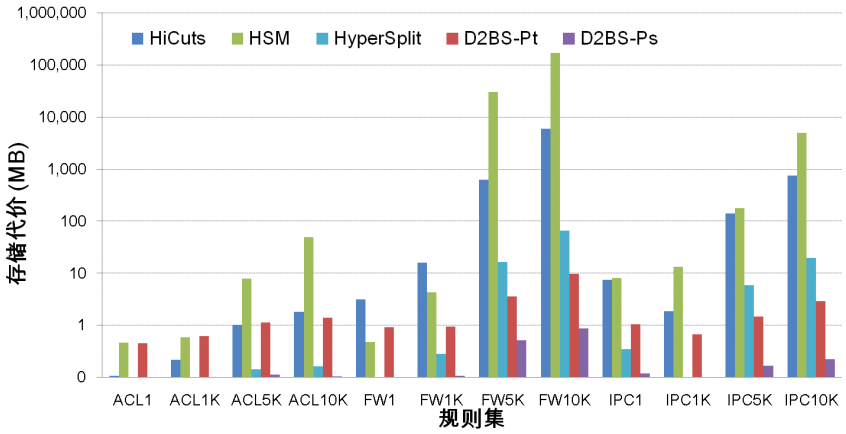


图　　空间性能

空间性能也是衡量算法性能的重要指标。当今大多数平台中的快速存储器尺寸都有限，算法数据结构所占用的内存对算法性能有着不可忽视的影响。特别是某些特殊平台，例如　　平台存在严格的存储限制。

图　给出了算法的空间性能比较结果。注意到纵坐标为对数坐标轴。从图　中我们可以看出，$D^2BS\ Ps$ 算法表现出了最佳的空间性能，在大多数的规则集上仅需要其它算法一个数量级以下的存储，特别是较大规模的规则集（包括　、　、　等）。在实验中，　　算法在大规模的复杂规则集　和　上无法成功生成分类的数据结构，给出其理论值作为对比的依据。以最复杂的规则集　为例，　、　和　算法都需要超

过　　字节的存储，而 $D^2BS\ Ps$ 算法仅仅需要不到　　字节的存储。即使是基于时间性能优化的 $D^2BS\ Pt$ 算法也表现出了优异的空间性能，特别是在大规模的复杂规则集上。

在比较空间性能实验中，注意到各类算法在　　类型的规则集上所需要的存储多于其余两种类型（　　和　　）。这暗示着　　类型的规则集具有更为复杂的内部结构。即使是同样规模的规则集，　　类型的规则集比　　和　　类型的规则集将产生更多的切分子空间。因此，作为对算法进行空间性能压力测试的客观基准，　　规则集可以较好的测试算法在处理复杂规则集时的性能体现。而即使在最大的　　规则集（　　）上，无论是 $D^2BS\ Ps$ 算法还是 $D^2BS\ Pt$ 算法都取得了远优于其它算法的空间性能。

3.4.1.3　扩展性能

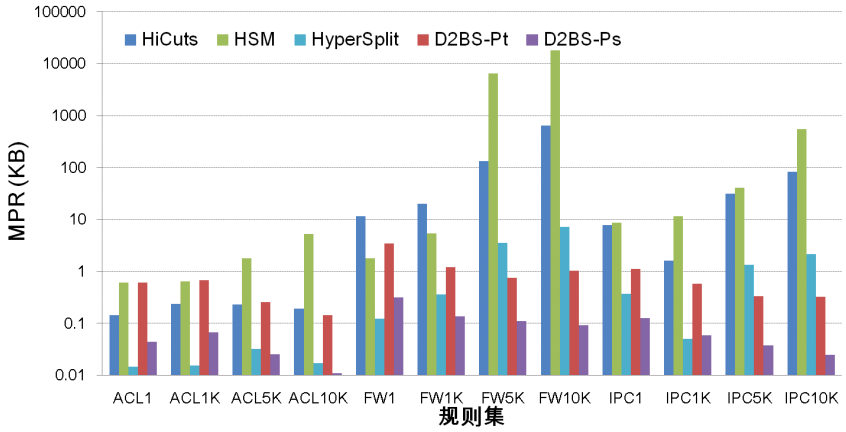


图　　扩展性能

算法的扩展性能，是算法的另一个重要指标。它衡量了算法在规则集快速增长的情况下算法的性能下降趋势。本文采用单位规则存储代价（　　）指标作为评测的指标，　　定义为算法生成的数据结构所占的存储代价除以规则集的规模。一般地，如果　　在不同的规则集上抖动的比较厉害，说明算法性能很不稳定，在新出现的复杂大规模规则集上将可能取得较坏的性能。反之，如果　　很稳定，则表明算法的性能随着规则集规模增长线性增长，具有较好的扩展性能。

图　给出了扩展性能的对比结果。图　中的结果表明 $D^2BS\ Ps$ 算法和 $D^2BS\ Pt$ 算法都表现出了比其它算法好的扩展性能。在所有的　个实验规则集上，　　的　　性能抖动变化超过　　倍，　　为　　倍，　　为　　倍。然而对于 $D^2BS\ Ps$ 算法和 $D^2BS\ Pt$ 算法，　　性能抖动约为　倍。这个

稳定的扩展性能保证了 D^2BS 算法在未来更大规模更复杂的规则集上也将表现出优异的性能。

同样的，比较基于搜索空间切分的代表算法　　和基于规则空间切分的代表算法　　、　　，从图　中的结果可以看出，基于搜索空间切分的算法具有较差的扩展性能。这是因为，基于搜索空间切分的算法在生成数据结构的时候并没有考虑规则集内部特征。即使是同等规模的规则集，内部特征不同，所造成的切分子空间差异可能较大。总之，基于搜索空间切分的算法对于规则集内部特征更加敏感。基于这一观察，为进行规则集内部特性的评测提出了新的思路，即对相同规模但不同类型的规则集，采用基于搜索空间切分的算法进行处理，所需要　　较大的则说明该对应规则集具有较复杂的内部特征。以　　的三个规则集为例，图　中的结果说明　　规则集具有最为复杂的内部特征，因此最难处理。

另外，比较　　在不同规模（从小规模到大规模）规则集上的变化趋势，只有 D^2BS 算法表现出了明显的下降。这一结果说明 D^2BS 算法在　　性能上具有亚线性特点。在处理大规模复杂规则集上， D^2BS 算法将有更好的实际表现。

3.4.2　硬件性能评测

为了测试算法在实际硬件平台上的吞吐性能，分别在多核网络处理器（　　）平台　　和现场可编程门阵列（　　）　　上进行了实验。

3.4.2.1　多核网络处理器平台

为了客观的验证算法性能，采用了最大的规则集之一（　　）作为实验规则集，并将算法在流行的多核处理器　　上实现。多核处理器运行在简单执行（　　）模式，以发掘处理器最大的性能潜力。其中，　　多核处理器的系统结构如图　所示。

　　多核处理器具有　个　　核，每个核的处理速度为　，可以根据使用需要启用不同数目的核。测试所采用的网包为　字节，测试在最小网包下的吞吐。 图　和图　分别给出了算法在多核处理器平台上启用不同数目核时的平均吞吐量和最坏吞吐量的性能。从结果中可以看出，从　个核到　个核， $D^2BS\ Ps$ 算法和 $D^2BS\ Pt$ 算法相对于其它算法都表现出了更高的吞吐性能（约为其它算法的　倍）。同时，在启用　个核的情况下，对于　字节小包流量，只有 D^2BS 算法可以实现线速的　　的处理速度。这些结果都表

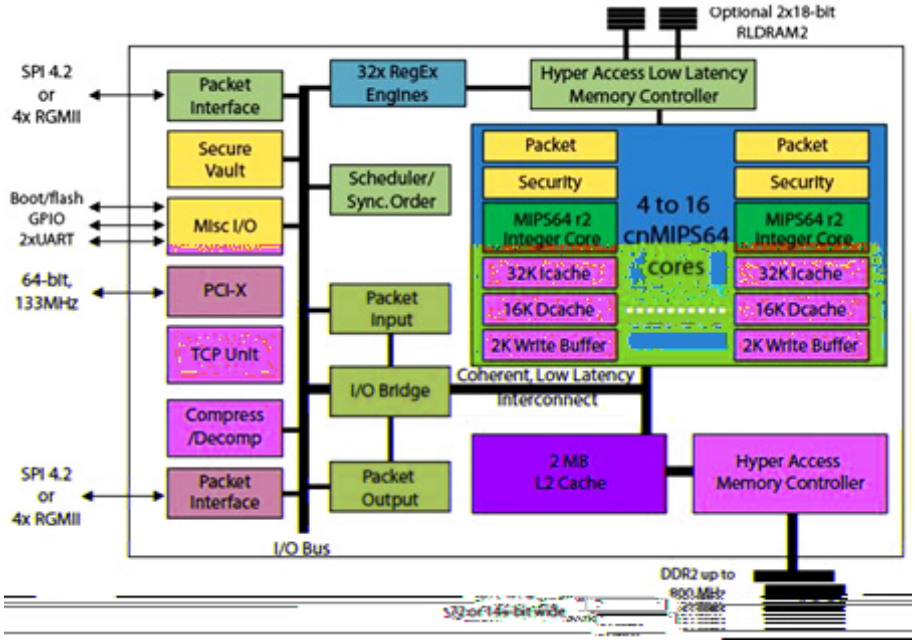


图　　多核网络处理器　　系统结构

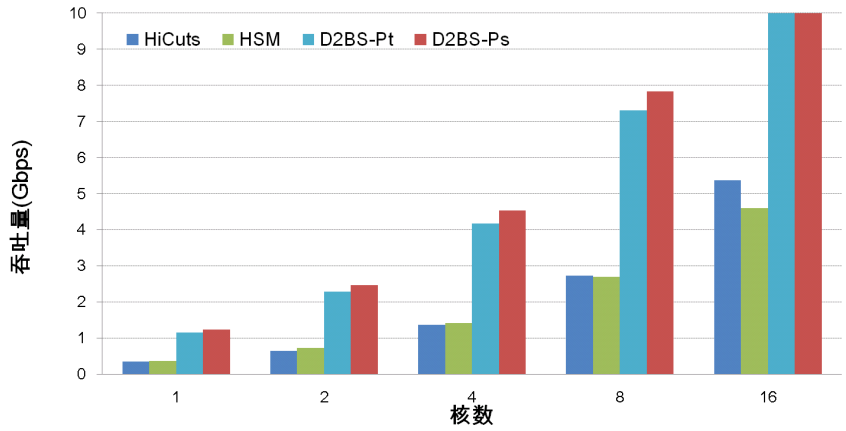


图　　多核网络处理器平台平均吞吐性能

明 D^2BS 算法在多核处理器平台上优于现有其它算法。同时，比较 $D^2BS\ Ps$ 算法和 $D^2BS\ Pt$ 算法，可以看出，$D^2BS\ Ps$ 算法比 $D^2BS\ Pt$ 算法取得了更高的吞吐量。这暗示着，在多核处理器平台上对算法进行空间优化往往能取得比进行时间优化更高的综合性能。

3.4.3　FPGA 平台

为了评测算法在　　平台上的性能，采用了流行的　　平台（模型为　　）。该平台共有　　的块存储器（　　），系统架构如图　　所示。考虑到硬件平台存储空间的限制，进行了 $D^2BS\ Ps$ 算法和

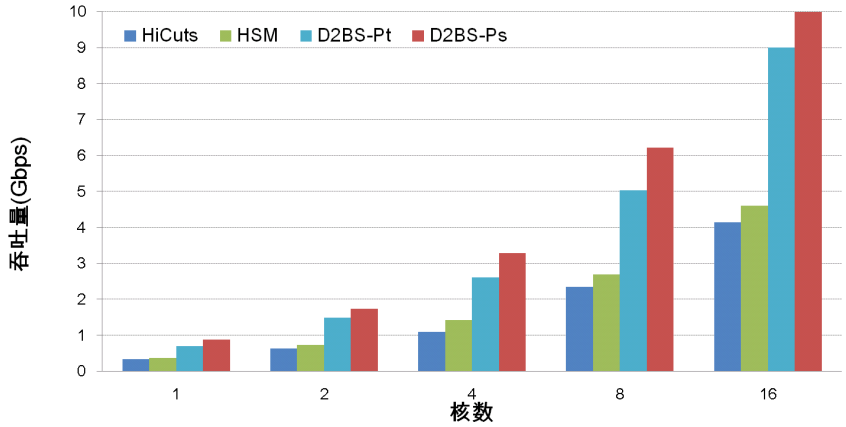


图　　多核网络处理器平台最坏吞吐性能

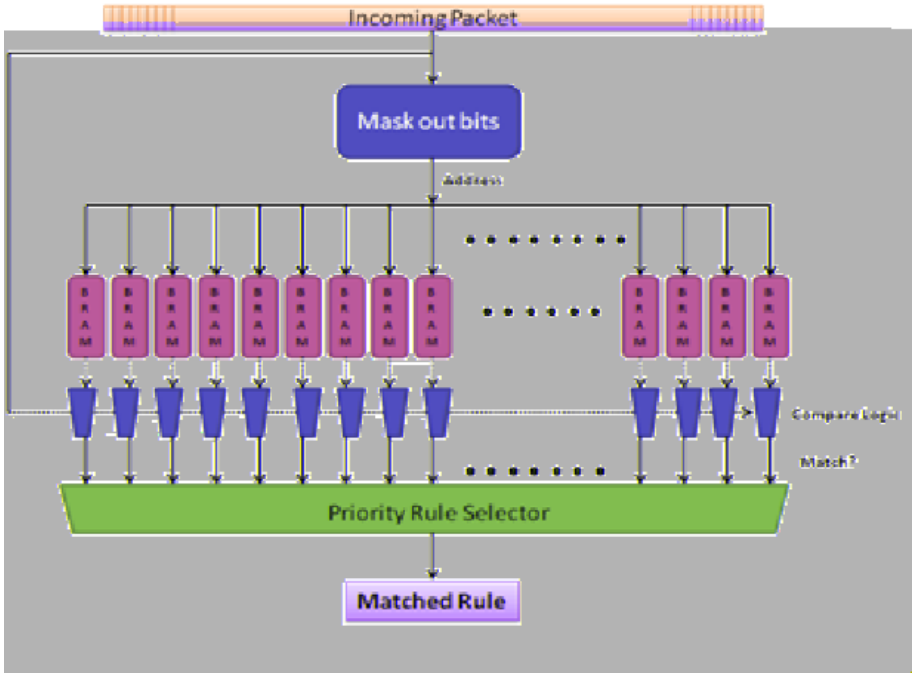


图　　平台系统架构

算法的性能比较。所采用的规则集包括　　类型的规则集，测试网包仍然为　　字节。

为了尽量发掘算法性能，将 $D^2BS\ Ps$ 算法进行了流水线（　　）设计。流水线过程包括两个环节。第一个环节将到达网包的包头中的特定位进行掩码处理，并利用掩码后的向量值所引导对应存储在　　中的　　。第二个环节是在索引到的　　内查找出匹配的最高优先级的规则。流水线结构确保每一次网包分类过程都仅仅需要一次流水线过程，保证了算法的最坏性能。

为了利用　　平台的　　并行查找的特点，将　　进行垂直存储，使所有的　　可以被同时读取和匹配。每块　　中存储有一个　　，

这样，在一个时钟周期内可以完成　　内的匹配。

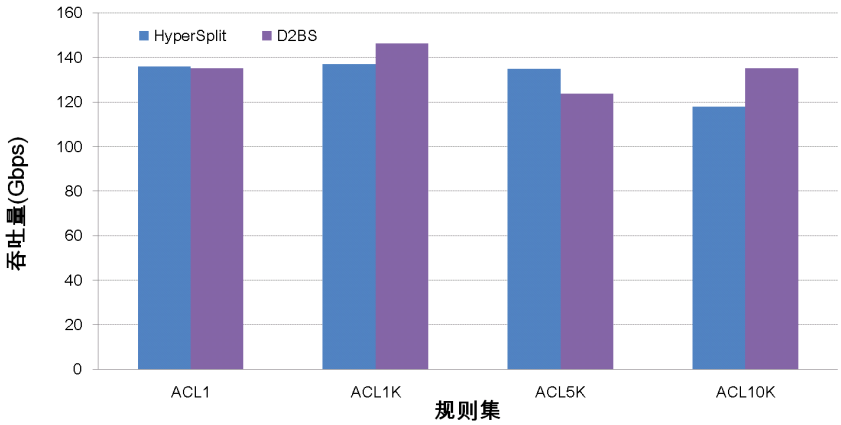


图　　平台吞吐性能

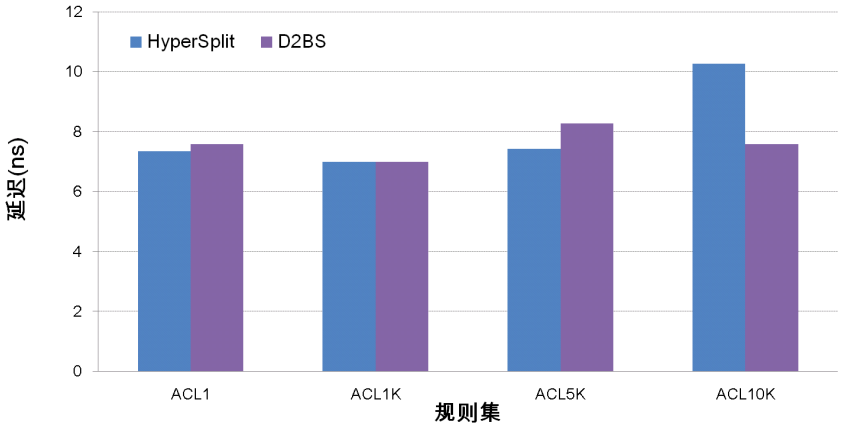


图　　平台延迟性能

图　　给出了　　平台上算法的吞吐量。在最大的　　规则集上，　　平台成功运行在　　频率上，等价于实际　　的吞吐量。这比已知最好的结果　　高了 14%。

另外，还评测了分类算法在处理网包时的延迟性能。图　　给出了算法在处理时引入的网包延迟。从结果中可以看出，D^2BS 算法始终保持了　　以下的处理延迟，这要优于现有工作　　。低的延迟也保证了 D^2BS 算法的实用性。

3.5　本章小结

本章主要研究了网包分类算法，并基于现有算法的分析，提出了一种创新的算法 D^2BS 算法。D^2BS 算法采用了启发式思想来充分发掘规则集内部特性，基于

位选取的映射函数高效地解决了空间切分中的信息冗余问题，大大提高了算法的空间性能。同时，通过两级分类结构，保证了时间性能。

实验结果揭示了 D^2BS 算法的实用性。无论在时间性能、空间性能还是扩展性上，无论是软件平台还是硬件平台， D^2BS 算法都大幅优于或接近已知的最佳算法。 D^2BS 算法在多核平台上，开启　个核时，在一万条规则的规则集上唯一实现了　　的线速处理；在　　平台上， D^2BS 算法实现了　　的吞吐性能。值得注意地是， D^2BS 算法的理论性能并不突出，但由于其充分发掘了规则集的内部特性，对切分信息进行尽力压缩，大幅提升了空间性能，同时保证了高的时间性能。

综上所述，本章研究了网包分类问题，并提出创新的 D^2BS 算法，无论是从分类速度、内存使用还是扩展性和实用性等方面均优于　　、　　、　　等高性能代表算法。

第 4 章　基于半监督机器学习的协议识别

4.1　本章引论

应用层协议识别技术是网络流量识别的基础技术之一，它将网络中的流量分为对应的协议类型，因此又被称为流量分类技术。与网包分类技术相比，协议识别技术进行的是应用层上的分类处理。相对于网包分类技术基于静态规则的分类，协议识别技术所采用的算法往往更加灵活、多样化。

传统的网络服务供应商和管理员采用基于传输层端口号的方法来进行简单协议识别。这种方法在各种流媒体、　　等新兴协议出现后，已经越来越不实用。新兴的网络协议往往采用非固定的端口号或动态协商端口（如　　、　　等），甚至能进行端口伪装（如　　、　　等），冒用其它常见协议（例如　　协议等）端口。这些新出现的协议给应用层协议识别技术带来了重大的挑战。

正如第　章中的总结，现有的协议识别技术，从基于端口、基于深度内容检测到基于机器学习的方法，都存在着固有的问题。基于端口的识别技术在复杂网流环境中无法保证较高的识别准确率。基于内容检测的方法处理复杂度很高，无法实现高速的流量处理，而且依赖于提前生成的特征集。同时，基于内容检测的方法无法处理带有加密内容的流量。而传统的机器学习方法要么采用监督机器学习，需要提前标定大量的样本；要么采用非监督机器学习，无法实现精确到类的协议识别。这些方法难以满足当前协议识别的实用需求。

实用的协议识别技术首先要能满足在线识别，需要实现早期识别（　　），即在网流发起的前几个包内即作出判断。同时要实现快速的流量处理和可扩展，因而不能依赖深度内容检测。而基于机器学习的方法相对则更为灵活和高效。因此，基于合适的机器学习的方法，同时融合其它方法的混合机制，将更为有效。

本章研究根据协议识别技术自身的特点和当前的需求出发，设计出实用的创新协议识别方法　　。首先，分析协议识别技术的具体需求；其次，从需求出发采用了基于半监督机器学习的协议识别技术，所提出的识别方法仅需要检测每个网流前　个网包的长度信息，而无需进行载荷上的深度检测；最后，利用　　对常见的六种协议（　　、　　、　　、　　、

、　　）进行了在线识别实验。实验结果表明，　　能满足实用的在线识别需求，在获得较高准确率的同时，保持了高的识别速度。

为了进一步提高新方法的灵活性，本章还引入可靠度系数，来对识别结果进行评估，将可靠程度较低的识别结果让其它识别器（例如基于深度内容检测）进行识别。这种混合机制可以提高整体的识别准确率和应用灵活性。

4.2　问题分析

随着新兴协议的不断出现，流量协议识别所面临的挑战越来越多，一些在传统环境中有效的方法如基于传输层端口的识别技术，在现有环境中已经被证明不能实用　　，而仍在使用的基于深度内容检测和基于机器学习的方法都存在自身难以克服的问题。下面将分别分析目前协议识别技术面临的挑战和评测方法。

4.2.1　难点分析

现有的协议识别技术主要面临下面四个方面的挑战：准确识别、高处理速度、早期识别和灵活扩展。

准确识别是协议识别技术的基本需求。高的识别准确率是进行后期流量处理的先决条件。目前最准确的方法是基于内容检测的方法，其理论识别准确率可以达到最高（100%）。而基于机器学习的方法，其识别准确率从 70% 到超过 90% 不等。一般地，协议识别方法的识别准确率要在 90% 以上才被认为是实用，在此前提下，越高越好。随着网络流量流量中协议类型的增多，给识别的准确率带来越来越大的挑战。

高处理速度也是进行实用的协议识别的重要需求。当前的主干网络带宽已经从百　发展到上　，对高速的网络流量进行实时处理，需要很高的处理速度。传统的基于内容检测的方法最主要问题也在于处理速度过低。为了实现在线识别，需要大量昂贵的专用硬件处理设备来进行加速，这影响了基于内容检测方法的大规模部署和应用。因此，高处理速度是实用协议识别技术的重要指标，也是目前亟待解决的重要问题。

早期识别是在线协议识别的需求。新的网流在建立之后，通过流量处理设备在通信双方进行交互。如果等整个网络的生命周期完成才识别出最终的结果，那么针对协议所采取的行动则无法执行。例如，网络管理员希望禁止掉占带宽巨大的　　协议。某　　网流在发起后，传输完毕数据，然后终止连接。越早成功识别，则越早可以采取中断操作，进一步地，网流造成的带宽消耗则越小。现有的

诸多协议识别方法，往往依赖较多的流特征（如流长度等），因此无法实现早期识别。

灵活扩展则意味着协议识别技术要适用于不同的网络流量环境和不同的用户需求。例如，在主干网络和数据中心网络中，网络中各类协议所占的比例往往不同，针对某类协议识别准确率高的方法，往往无法适用于不同的环境。另外，企业网络用户往往对于识别的准确率十分看重，而个人用户则可能更看重识别的速度。而现有的诸多协议识别方法往往局限于某种特定技术，难以有效地组合多种方法的优势，取得较好的灵活性。

4.2.2　评测方法

对协议识别的方法进行评测主要包括准确性能、处理速度、预处理速度等方面，这些性能综合反应了协议识别技术的应用特性。

准确性能包括对识别方法的精确率、召回率和准确率等性能参数，分别体现了在识别的敏感性、可靠性和正确性上的体现。一般将基于内容检测的方法的识别结果作为准确性能的参考标准。

处理速度指单位时间内识别的网络流量多少，或识别速率，多采用吞吐量（每秒钟成功识别的网流个数）作为衡量的标准。处理速度是一个很重要的指标，直接影响系统对网络流量的实际处理性能。

预处理速度包括当系统被迁移到一个新的环境中，需要多久的预备处理以完成相关数据结构的生成。预处理速度对于在线的协议识别系统十分关键。因为大部分系统在进行分类器更新时都需要根据新的参数，重新生成数据结构。预处理时间过长，会导致系统中断过久。

此外，在进行评测的数据集上，由于大多数公开的数据集（例如　　）基于信息安全和用户隐私的考虑，并不包含任何应用协议的标记信息，无法直接作为评测使用。因此，需要利用其它的手段采集数据集并进行协议类型标定。

4.3　半监督机器学习

4.3.1　定义和假设

半监督学习　　，可以同时利用标记（　　）和未标记（　　）的样本来培训分类器。该方法已经被越来越多的研究者关注。传统的机器学习依赖大量带标记的训练样本，而收集和标定大量的样本数据十分困难和耗时。

在半监督学习中，训练数据集是由两部分组成：带标记的样本（每个样品由特征对表示） $X^L = \{(x_i, y_i)\}_{i=1}^{l}$ 和不带标记的样本 $X^U = \{x_j\}_{j=l+1}^{l+u}$。对于二类分

类，$y_i \in \{-1,1\}$ 对于多类分类，$y_i \in \{1,2,\ldots,N\}$，其中 N 是测试类的个数。半监督学习的最终目的是生成一个判别函数 $f: X \to Y$，可以根据样本 x 来计算出正确的结果 y。一般地，在预先知道密度函数的前提下，半监督学习会在监督学习的基础上取得更准确的效果。

在真实的网络流量分类中，大量的标记样本获得代价昂贵，因此，传统的监督学习方法，如　　由于缺乏足够多的标记样本而表现较差。而另外一方面，无标签样本则是很容易收集的。例如，采集无标签的网络流量对于众多的流量采集工具来说十分简便，如　　。因此，基于半监督学习方法分类网络流量同时利用标记和未标记的训练数据，将可能取得更佳的分类效果。

采用半监督学习之前，必须首先确定一些假设。文献　　中建议，同一类的样本应该以较大的概率聚合到一起，另外，不同类的样本之间相隔应该较远。

基于这个假设，可以找到一个自然的方式来利用未标记的数据，生成密度函数 $p(x)$ 和 $p(y|x)$，其中 $p(x)$ 是样本的概率密度函数，$p(y|x)$ 是标签的条件概率密度函数。相反，传统的鉴别方法在估计 $p(y|x)$ 时候没有考虑样本的概率密度，并不适用于半监督学习。

这个假设并不适用于所有的分类情况，但在流量分类问题上可以较好的满足。由于为类似网络应用程序设计的协议往往具有类似的网包长度。例如，　　协议，在经过几个网包的协商后将试图快速进行大数据的传输。在　　中，采用了现有的转换支持向量机（　　，或　　）方法　来解决分类器设计问题，这也符合之前的假设分析。

4.3.2　转换支持向量机

　　是一个标准的支持向量机（　　）在同时利用标记和无标记样本情况下的扩展。假如有 l 个带标记的样本 $\{x_i, y_i\}_{i=1}^{l}$ 和 u 个无标记的样本 $\{x_j^*\}_{j=1}^{u}$，其中 $x_i, x_j^* \in R^d$，$y_i \in \{-1,+1\}$。最终利用无标记的样本训练生成一个线性的分类器 $sign(w^T x + b)$。图　中给出了　　的一个简单例子，表明无标签数据确实有助于提高分类效果。图　中，“ + ”和“ − ”分别表示不同类别的样本，而实心圆代表无标签样本。仅利用标记样本的　　生成的分类超平面为虚线，而利用额外的无标签数据的　　生成了更准确的分类超平面，如实线所示。

对于可线性分离的数据情况，半监督机器学习问题可以由式（　）给出。

$$\begin{aligned} \text{argmin} \quad & (y_1^*, \cdots, y_u^*, w, b): \\ & \frac{1}{2}||w||^2 \\ subject\ to: \quad & \forall_{i=1}^{n}: y_i[w \cdot x_i + b] \geq 1 \end{aligned}$$

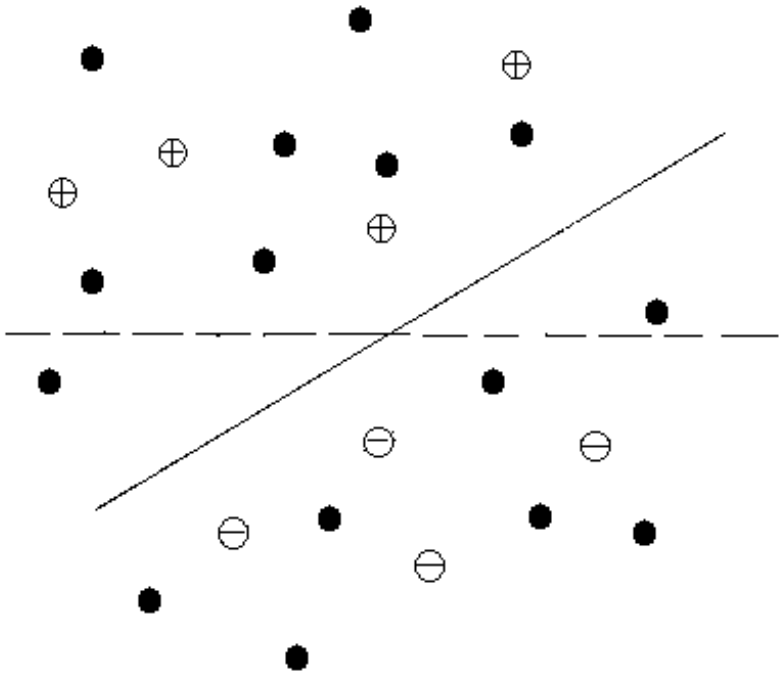


图　　在　　中，利用不带标签的样本可以增加分类准确性

$$\forall_{j=1}^{u} : y_j^*[w \cdot x_j^* + b] \geq 1$$

在式（　）中，y_j^* 是对无标记样本 x_j^* 的标记。在这种情况下，　　是对每个测试样本生成标记 $y_1^*, \cdots, y_u^*$，同时生成一个满足最大化概率的超平面 $< w, b >$ 分离原先标记的样本和新标记的数据。

对于非可分离的数据，可以使用标准　　中的松弛变量 ξ_i 来进行处理，参见式（　）。

$$\begin{aligned} \text{argmin} \quad & (y_1^*, \cdots, y_u^*, w, b, \xi_1, \cdots, \xi_n, \xi_1^*, \cdots, \xi_u^*) : \\ & \frac{1}{2}||w||^2 + T\sum_{i=0}^{n}\xi_i + T^*\sum_{j=0}^{u}\xi_j^* \\ subject\ to: \quad & \forall_{i=1}^{n} : y_i[w \cdot x_i + b] \geq 1 - \xi_i \\ & \forall_{j=1}^{u} : y_j^*[w \cdot x_j^* + b] \geq 1 - \xi_j^* \\ & \forall_{i=1}^{n} : \xi_i > 0 \\ & \forall_{j=1}^{u} : \xi_j^* > 0 \end{aligned}$$

其中，T 和 T^* 是可以调整的参数，用户可以通过调整参数，控制分类边距阈值。此外，被测样品的标签被分配为 $+1$ 的数量，也可以进行手动指定。因

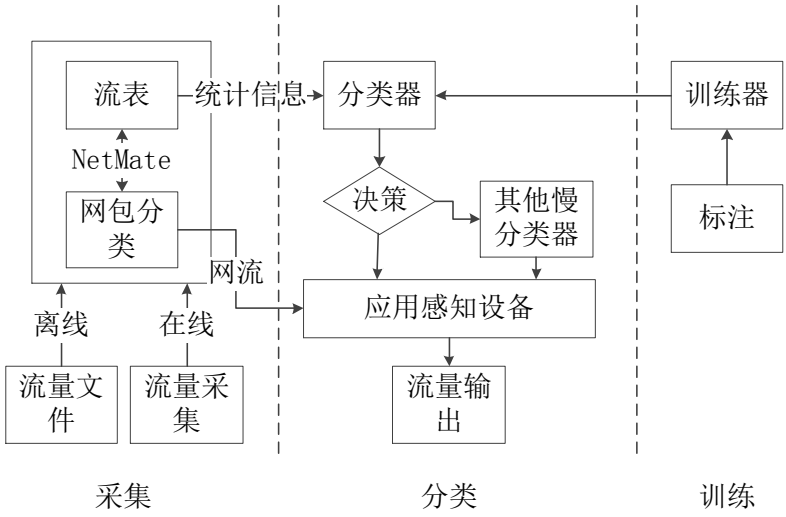


图　　系统框架

此，通过这两个参数可以满足不同的性能要求。

中对无标记样本的标记过程也需要优化，这其实是一个组合优化问题。理论上，计算一个精确的　　是　　，在本文中采用快速的解决方案以保证处理速度　　。

为了解决这个问题，　　设计了一个算法来找到一个近似解，保证在有限步骤收敛。　　等人　则设计了一种改进的快速方法来解决大型线性　　问题 L_2　　的有限牛顿法。　　等人　用这种方法作为　　的子问题求解方法。通常采取　　的方法来加速学习过程。

在流量分类中采用　　有两个主要优点。首先，在流量分类问题的情况下，存在大量未标记的流量和很少标记的流量，符合半监督学习条件。此外，分类阶段需要较低的计算复杂度，以保证较快的速度，这样，可以在保证良好的精度实现快速的网络流量分类。

4.4　基于半监督机器学习的协议识别方案

在本节中，首先介绍　　系统的框架，然后分别在二类和多类的情况下讨论具体的分类方法实现。之后，详细介绍混合方案的引入和在网包乱序情况下的解决方案。

4.4.1　系统框架

整个系统主要包括三个部分：**采集**，**训练**和**分类**。这三个部分一共包括六个子组件：采集器（　　）、决策器（　　）、其它慢分类器（　　，或　　）、训练器（　　）、标记器（　　）和分类器（　　）。图　给出了系统的框架示意图。下面将讨论每个组件的功能。

4.4.1.1　采集

该部分仅包括一个组件，采集器。在这一部分，需要通过采集器在线地采集网络流量或者导入离线的流量数据。在诸多现成采集器工具，选择了　　作为系统中的基本采集器组成部分，因为它是开源平台，而且支持灵活地插入新的自定义模块。

原始　　支持将传入的网包分类为网流（根据网包包头的五元组信息）。在　　基础上，设计一个流量管理模块来记录分类后网络的统计信息，包括每个交互网流的前几个包的大小（不包括　　握手信息）。值得注意地是，网包大小的统计结果和网包的序号将被顺序采集，而无需等待整个流结束，这种流水线结构有利于后续分类环节的处理。

4.4.1.2　训练

训练包括两个组件，包括训练器和标记器。训练器组件会根据给定的训练数据产生分类器。通常情况下，未标记的流量可以由　　收集，而带标记的流量可以被其它识别工具进行标记，例如　　。标记器还提供了一个灵活的用户标记界面，支持标记流量上标签的手动检查。用户参与的检查步骤，将提高对流量进行标记的准确性。

4.4.1.3　分类

分类部分包括三个组件：分类器，决策器和　　。主要的半监督学习算法实现在分类器组件中。对于基本分类，分类器将在产生分类结果的同时生成一个可信度参数 C（定义参见第　　节）。

如果启用了混合机制，会多出一个额外的检查步骤。决策器组件将比较可靠度系数 C 和一个给定的阈值 τ，如果 $C \leq \tau$（即分类结果不够可靠），决策器将网流发送给慢分类器，并返回它们的分类结果，因此，不同部分的流量可能会由不同的组件分别进行分类。　　提供了对不同慢分类器的支持，可以优化调整分类精度和分类速度之间的权衡。实验中采取基于深度检测的慢分类器，来实现

准确的流量识别，但处理速度相对较慢。改变 τ 值可以实现性能的调整。一般来说，一个较小的 τ 会导致更高的分类准确率和较低的速度，反之亦然。

4.4.2　分类器设计

在实际网络的流量中，同时存在不同类型的应用协议。因此，　　需要同时处理多类的情况。为方便描述，在这里，首先讨论二类分类（　　）的情况。二类分类中 $y_i \in \{+1,-1\}$。后续将该方法扩展到多类分类（　　）的情况。

4.4.2.1　二类分类

在二类分类的情况下，对采集的流量根据二类分类器给出标定的结果。训练样本根据携带标签与否可分为两部分。对于带标签样本，表示为 $\{x_i,y_i\}_{i=1}^{L}$，其中 x_i 是第 i 个样本，y_i 表示对应的协议类型，L 是样本集的大小。同时，未标记的训练集表示为 $\{x_l^*\}$，大小为 U。然后，设计一个基于　　的分类器来预测测试流量的标签信息。

标准　　的处理过程可以由式（　）给出。

$$\begin{aligned}\min_{w,\{y^*\}_{j=1}^{U}} \quad & \frac{\lambda}{2}||w||^2+\frac{1}{2L}\sum_{i=1}^{L} l(y_i w^T x_i) \\ & +\frac{\lambda^*}{2U}\sum_{j=1}^{U} l(y_j^* w^T x_j^*) \\ subject\ to: \quad & \frac{1}{U}\sum_{j=1}^{U}\max[0,sign(w^T x_j^*)]=r\end{aligned}$$

其中 λ^* 是由用户来控制未标注样本对训练过程影响力的参数，将通过交叉验证来最终确定。r 一般是样本中带标签　　的比例，而 l 通常设置为损失函数，定义为 $l(x)=max(0,1-|x|)^2$。

算法 给出了在二类分类情况下训练　　分类器 $<w,b>$ 的过程。该分类将对未知数据计算 $R=wx_{unseen}+b$，并进行标记，计算过程在式（　）给出。

$$y=\begin{cases}+1 & R>0\\ -1 & R<0\end{cases}$$

在算法 中，函数　　是为了解决问题（　）。

$$\min_w \frac{\lambda}{2}||w||^2+\frac{1}{2L}\sum_{i=1}^{L} l_2(y_i w^T x_i)+\frac{\lambda'}{2U}\sum_{j=1}^{U} l_2(y_j^* w^T x_j^*)$$

算法 3 二类情况下　　训练过程

输入：

带标签的训练数据 $(x_1,y_1),\cdots,(x_n,y_n)$

不带标签的训练数据 $x_1^*,\cdots,x_n^*$

参数 λ,λ^*，见式（　）

输出：

分类的超平面 $<w,b>$

(w,b)　　$(x_1,y_1),\cdots,(x_n,y_n),\lambda$

$\lambda' := 10^{-5}$ // 给定的小数值

while $\lambda' < \lambda^*$ **do**

do

$(w,b,\vec{y}^*)$　　$(x_1,y_1),\cdots,(x_L,y_L)$

$(x_1^*,y_1^*),\cdots,(x_U^*,y_U^*),\lambda,\lambda'$ // 采用修改的有限　　方法 l_2　　解决该问题

标签交换过程

while　　>0

$\lambda' = 1.5\times\lambda'$

end while

w,b

式（　）中是一个监督的　　问题，而 λ' 和 $y_1^*,\cdots,y_U^*$ 都是已知的参数。

4.4.2.2　多类分类

真实的网络流量需要同时对多种类型的协议进行分类。传统的适用于二类分类情况的　　方法，将无法处理这种情况。为了扩展到多类分类情况，下面将基于二类分类的过程来实现多个类的分类。问题的关键是如何设计这些二类分类，同时在二类分类基础上生成多类分类的结果。一般来说，有两种类型的组合技术可以采用：一**对一**方法或一**对多**方法。

一对一　对于每一对协议，训练一个二类分类器。如果有 N 种协议，使用此方法将一共产生 $\frac{N(N-1)}{2}$ 个分类器。进行分类时，各分类器将对结果进行表决，获得最终的结果。

一对多　对于每种协议，整个数据集分为两部分：匹配子集和非匹配子集。基于这两部分训练两类分类器。对于 N 类，将产生 N 个分类器。如果预测

结果在不同的分类器之间发生冲突，则以具有最大 $\frac{R}{||w||}$ 值的分类器结果为准。

实验中同时测试了这两种方法，结果发现一对一方法的性能要优于一对多方法。分析一对多方法，每次在进行分类器训练时候，目标协议考虑为一类，非目标的所有协议考虑为另一类，这通常会导致不平衡的训练数据规模。此外，虽然 非目标 类别的样本可以聚集成几个协议类型的集群，在使用二类分类情况下，在原有的分类空间分类可以生成一个超平面。然而，这对于一个多簇的分类空间，分为两类将会导致准确率的下降。此外，最终分类超平面将受到每一个二类分类器的强烈影响，而在一对一方法中，分类只需要处理一对协议，这使得它更容易获得正确的结果。

此外，从直观上也可以解释两种方法的优劣。具有 d 个特征的样本可以被视为 d 维超空间中的点。一对一方法将最终产生 $\frac{N(N-1)}{2}$ 个超平面作为最终的分类面，而一对多方法仅产生 N 个分类面。因此一对一方法可以获得更精确的结果。

基于上述分析和实验结果，我们选择了一对一方法。训练和分类过程在算法 中给出。

4.4.3　混合机制

图　　流量的分类面

混合机制的核心思想是运用可靠度系数（　　，或 C）来帮助确定流量的哪一部分应该转移到　　。本文在对　　分类结果的观察基础

算法 4 多类分类情况下的处理过程

输入：

标记样本 $(x_1,y_1),\cdots,(x_L,y_L)(y_i\in\{1,\cdots,N\})$

无标记样本 $x_1^*,\cdots,x_U^*$

参数 λ,λ^*，见式（　）

测试数据 x_{unseen}

输出：

对每一个未知样本 x_{unseen} 进行预测 y_{unseen}

for $i,j\in\{1,\cdots,N\},i<j$ **do**

$<w_{ij},b_{ij}>$

end for

$y=[0,\ldots,0]\in R^N$

for $i,j\in\{1,\cdots,N\},i<j$ **do**

if $w_{ij}x_{unseen}+b_{ij}>0$ **then**

$y(i)=y(i)+1$

else

$y(j)=y(j)+1$

end if

end for

$y_{unseen}=arg\max_i y(i)$

上，提出了基于可靠度系数的混合机制。

对测试数据 x，　　将生成一组超平面 $<w_{ij},b_{ij}>$。每个超平面对测试数据 x 计算 $R=wx+b$。R 可以被认为是从测试数据到分类超平面的距离。如果 $R\geq 0$，x 将被视为某类，反之则为另外一类。当 $R=0$ 时，说明测试数据刚好位于超平面上，可以被识别为两类中的任意一类。一般地，较小的 R 往往意味着可能不可靠的分类结果。实验结果支持了这一猜测。图　给出了一个一个　　流量的超平面。在结果中，大多数　　流量在超平面的上方（被正确分类），而只有少数的流量在超平面下方。然而，即使是被分错的　　流量，仍然以较大的概率位于超平面附近。

在图　中，对　　个　　流量进行了分类测试，其中测试数据显示为环，而超平面以虚线表示。如果分类结果为正（超平面以上），则该数据被正确识别。

图　显示部分测试数据被误判为负（超平面以下），其中大部分误判数据都位于分类超平面附近。

一般认为，如果离超平面的距离值较大，预测的结果将更加可靠。本文中根据距离值定义一个可靠度系数 C。可靠度系数的范围为 $[0.5, 1.0]$，并且应该与距离 $\frac{|R|}{||w||}$ 成正相关关系。当 $\frac{|R|}{||w||}$ 接近 ∞，C 应该接近　。当 $\frac{|R|}{||w||}=0$，测试样本位于超平面，这表明一个分为两类的概率应该相等，此时 C 应该为　。凡是满足上述关系的函数都可以被认为是一种描述可信度的参数。

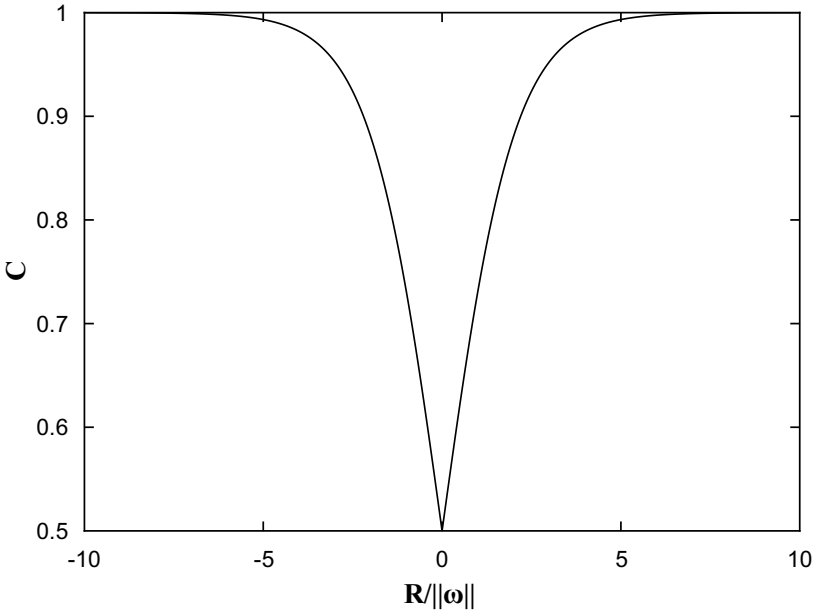


图　　一种可能的可靠度系数定义

基于上面的观察，在式（　）中给出了可靠度系数的一种可能定义。

$$C = \frac{1}{1 + e^{-\frac{|R|}{||w||}}}$$

为了给出直观的印象，图　中给出了 C 的函数曲线。图　表明，式（　）定义的可靠度系数在 $\frac{|R|}{||w||}$ 从　到 $+\infty$ 时，取值从 0.5 到 1.0，符合要求。

在实验中，最终的分类面不是一个单一的超平面，而是不同的超平面的组合。因此，对于每个测试样本，每一个超平面都会给出距离值 $\frac{|R|}{||w||}$，这将导致在不同的 C 值。一般地，可以采用 C 值的不同计算方法，如最小值、最大值、平均值等作为计算最终可靠度系数的依据。从理论上讲，利用最终的一组分类面分类样本时，只有一个正确的，因此数据仅接近少数分类面，而其它的超平面将与数据保持较远距离。因此最大值 C_{max} 和平均值 C_{avg} 通常较大，无法精确地体现分类状况。因此采用最小值 C_{min} 往往是比较有效的方案。其计算方法为

$\frac{|R|}{||w||} = \min\{\frac{|R_{ij}|}{||w_{ij}||} | i,j \in \{1,2,\ldots N\}, i < j\}$。

4.4.4　网包乱序

在实际网络中，一些网包可能由于多路径传输导致乱序。目前，大多数解决方案依赖于网包重组（　　）技术。这种技术可能会导致一些新的问题，例如，如果分类技术需要网流的前 k 个网包，则必须进行等待，直到所有的前 k 个网包都到达后再进行重组。更坏的情况是，一些网包在传输中可能会发生丢失，将导致重传等待。这些情况都会造成依赖网包重组方案较大的处理延迟和存储代价。

为了解决这个问题，设计了能兼容缺失特征的分类器。所有的有效特征（网包长度），都不小于　，将缺失的特征值缺省设置为　。因为 $R = wx + b$，其处理过程并不做改变。这意味着，当丢失了一些特征时，使用一个近似的超平面来进行分类，而不是确切的准确分类。新的超平面平行于缺失的特征在特征空间的维数。通常，这种修改可能会带来一定的分类精度损失。

4.5　性能评测

4.5.1　实验环境

表　　数据集的全局统计

数据集	采集日期	流数	网包	容量（字节）
$trace_A$				
$trace_B$				

最新公开的数据集（例如　　），由于信息安全和隐私的原因，并不包含任何应用协议的信息，无法直接作为实验使用。本文中实验基于在一个大的包含数百个各种服务器的研究机构收集的真实网络流量数据集。该数据集主要由两部分组成：$trace_A$（在　　年收集）和 $trace_B$（在　　年收集）。一个可靠和准确标记的流量数据集，将对后续研究提供很大的帮助，

出于采集时间的考虑，这些数据集并不是十分庞大，但这些数据集是在不同时间点采样收集的，以尽量客观反映网络流量的平均情况。为了验证　　的分类结果，首先采用内容检测的方法来标记数据集中的流量协议。

基本的统计信息和应用信息在表　和表　中给出。在表　中，发现　　和　　协议的流数（　　）和容量（　　）都占了较大比例。比较两个数据集文

表　　数据集的协议统计

协议	$trace_A$ **流数**	$trace_A$ **容量**	$trace_B$ **流数**	$trace_B$ **容量**

件，观察到一些有趣的特性。以　　协议为例，从　年到　年，容量百分比增长超过 400%，而流数的百分比却发生了下降，这主要是由于通过　　协议的视频应用在近些年急剧增加。此外，　　协议的比例也增加了很多，这意味着用户越来越关注自身的安全和隐私，开始重视使用加密内容的协议。后面的实验将集中在　种代表性应用，包括　　（非视频的　　），　　（基于　　），　　，　　，　　和　　。

图　　不同端口的流数和容量的累积分布函数

为了显示不同协议端口号详细分布，采用了累积分布函数（　　）对 $trace_B$ 进行了描述。图　　采用了对数刻度　轴，根据端口号进行了统计（对于双向连接的流量，以其中较小的一个端口号作为记录的端口）。从图　　中，可以看到，

端口范围为 和 的应用占据了大部分的流数，而端口范围在 的应用则占据了大量的容量。这意味着，虽然使用 短流仍然保持流数的很大一部分，各种动态下载应用的流量占据了大部分的容量。

实验平台集成了 ，作为流量抓取的组件。实验平台运行在一台 上，系统为 操作系统（内核 ），硬件为英特尔 （主频为 ，缓存 ）和 的 内存。所有的程序都由 语言实现。

4.5.2 测试指标

为了全面地评测 的性能，采用了多种性能指标。首先给出几个前提定义如下。以应用协议 A 为例。

- **真阳性** *(True Positive，或 TP)* 协议 A 的所有流量中被正确识别为协议 A，这代表了分类的准确性。
- **真阴性** *(True Negative，或 TN)* 非协议 A 的流量被成功识别为非协议 A，这也代表了正确的结果。
- **假阳性** *(False Positive，或 FP)* 非协议 A 的流量被误判为协议 A。例如正常的 流量被识别为 流量。假阳性会导致误报（ ）情况。
- **假阴性** *(True Negative，或 FN)* 协议 A 的流量被误判为非协议 A。这将导致漏报，从而导致准确度的下降。

基于上面的定义，进一步给出性能指标的定义，包括**精确率**，**召回率**和**准确率**。定义如下：

- **精确率** 精确率（ ）定义为被识别为 A 的流量中，真实为 A 的比例。其计算公式为 $\frac{Traffic_{TP}}{Traffic_{TP}+Traffic_{FP}}$。精确率可以对某个特定协议或者多种协议进行评测。
- **召回率** 召回率（ ）被定义为协议 A 的流量中，被正确识别为协议 A 的比例。其计算公式为 $\frac{Traffic_{TP}}{Traffic_{TP}+Traffic_{FN}}$。召回率可以对某个特定协议或者多种协议进行评测。
- **准确率** 准确率（ ）为流量中所有被正确识别的流量所占的比例，其计算公式为 $\frac{Traffic_{TP}+Traffic_{TN}}{Traffic_{TP}+Traffic_{TN}+Traffic_{FP}+Traffic_{FN}}$。注意准确率是对流量整体进行评测的参数。

除了以上的参数，还采用了误判矩阵（ ）来辅助描述分类结果的细节。

误判矩阵是一个 $N \times N$ 的矩阵，其中，元素 (i, j) 是第 i 类协议被误识别为第 j 类协议的比例。误判矩阵可以给出错误分类结果的细节，用于发现不同协议的

内在联系特性。

4.5.3　不同网包个数结果

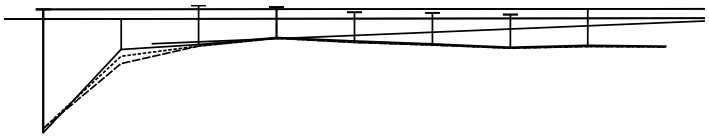


图　　不同网包个数下精确率、召回率和准确率的结果

尝试仅利用每个网流前 k 个网包的长度（不带　　控制报文）来进行协议识别。从理论上讲，增大 k 值将可能提高分类结果的准确性，但增加处理延迟。因此，需要选择一个合适的 k 值，以实现及时和准确的识别。

为了找出一个合理的 k，对不同取值的 k 进行分类实验。每次实验随机从训练集中抽取　　个标记流量样本和　　个无标记流量样本来训练分类器。每个实验重复　次，分别对 k 从　到　　进行实验，计算平均值作为实验的最终结果。各实验的精确率、召回率和准确率结果在图　中给出。

精确率 图　中，当网包从　增加到　时，平均精确率曲线从约 80% 增大到超过 94%。对于超过　个网包，平均精确率保持在 93% 以上，其中最好的结果是 95.2%（当　个网包时）。此外，当网包从　增加到　时，不同协议之间，精确率的差值从约 40% 降低到 20%。当超过　个网包时，差值变化不大。这一结果意味着，当 $k \leq 4$ 时，精确率随着包数量的增加而提高，当 $k > 4$ 后保持稳定。

此外，还检查了在最小的网包个数时，最坏和最好的精确率所出现的协议，分别是视频（基于　　）和　　协议。这个结果说明，基于网包长度来识别流行的基于　　协议的视频应用是十分有挑战的（约 60% 的精度），而　　协议则相对比较容易基于网包长度被识别（近　　%）。

召回率 如图　所示，平均召回率曲线与平均精确率曲线类似。当网包个数从　增长到　时，召回率从约 80% 增长到 94%。当网包的数量从　提高到　，平均召回率曲线在 93.61%（　包）到 95.09%（　包）之间波动。这意味着，超过　个网包就可以取得良好的效果。此外，当之有　个网包时，召回率的方差是 60% 以上，而超过　包时，则大幅减小。例如，基于　个网包的长度，方差只有 11.7%。进一步检查了　个网包的实验结果。最差的应用是视频（基于　　），这表明，视频识别（基于　　）是当今流量分类研究一个很有挑战的问题。

准确率 图　给出了所有应用协议上的分类结果。结果表明，当网包个数从　增长到　时，准确率从约 81.6% 增长到 93.9%。当超过　个网包时，准确率保持在 93.6% 以上。令人惊讶地是，最好的准确率也在　个网包时达到，为 95.1%，这个结果与现有工作　是类似的，文中采用　个网包长度取得了 90% 左右的准确率。

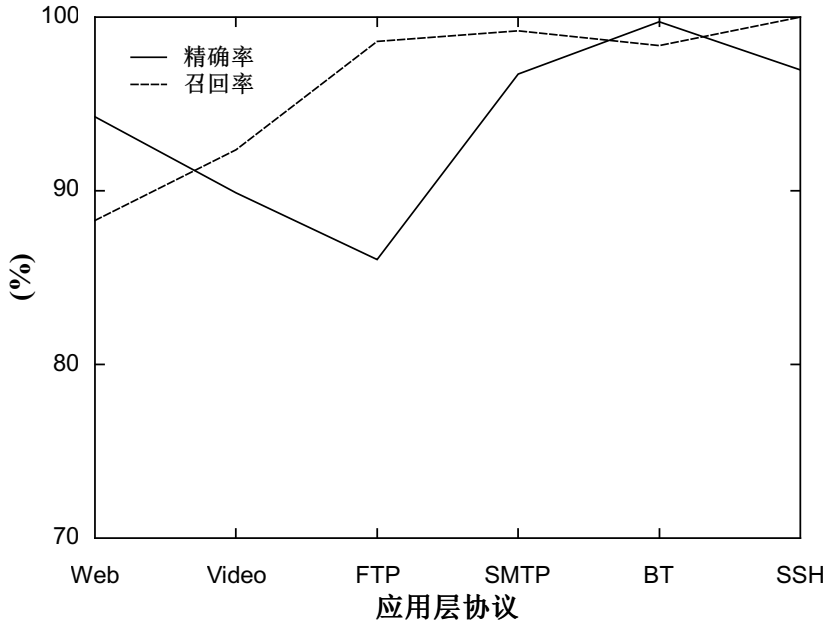


图　　个网包下的协议识别结果

最佳网包个数选取 基于图　中的结果，可以得到一个基本结论。当网包的数量较小时，所有的性能指标都较差，不同协议时间的性能方差都很大。部分原因是因为只有两个分类特征，没有足够的信息来代表网流的识别特点。随着网包个数的增加，性能迅速提高，然后继续保持在一个较高的水平。最佳的性能结果发生　个网包时。

基于每个网流的前　个网包，图　给出了所有测试协议的精度和召回结果，包括　，　，　，　，　和　协议。图　中给出的结果，精确率

表　　误判矩阵

比率	

反
范围从 86.04% 到 99.73%，而召回率范围从 88.29% 到 100%。所有这些结果表明，

在所有应用协议的识别上，同时取得了较高的精确率和召回率。

误判矩阵 在表　中，给出了基于前　个网包的误判矩阵。从这个矩阵，可以看看到所有测试的应用协议的详细分类结果。例如，在第一列中， 6.31% 的　　流量被误识别为　　（基于　　），0.80% 的　　流量被误识别为　　，0.85% 的　　流量被误识别为　　。大部分误判的比率都较小，最大的误判发生在　　和　　之间，这意味着，　　和　　协议之间的前　个网包大小的分布是相似的。一个可能的原因是，它们都使用　　协议，从而其初始化过程是类似的。

总之，最好的识别结果发生在　个网包时。因此，在后续的实验中，基于前　个网包的大小来进行协议识别实验。

4.5.4　与基于监督学习方法的对比

为了验证　　的有效性，与基于监督学习的分类方法，支持向量机（　　）进行了性能比较。在每次实验中，共有　个带标记的网流样本，不同数在鳙

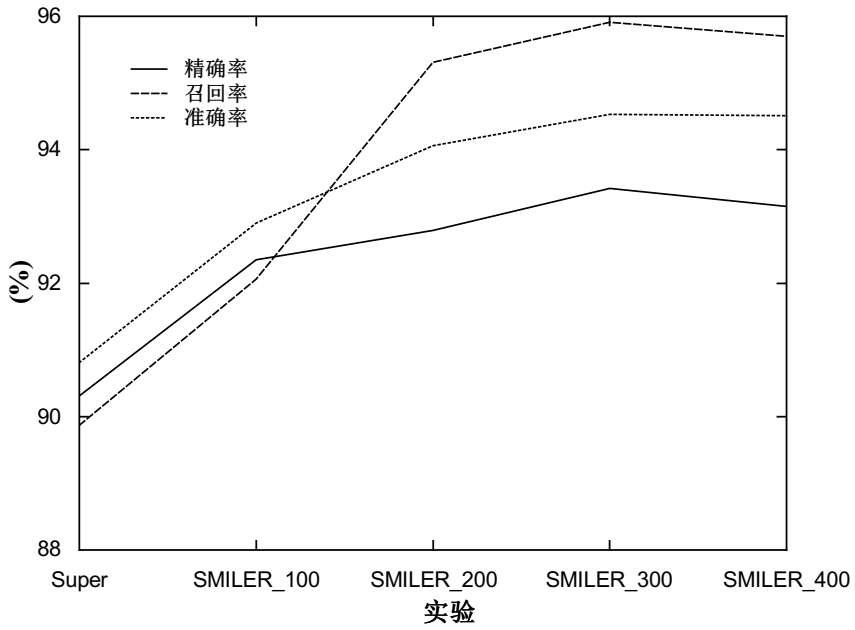


图　　和监督学习方法的准确率结果比较

并没有增加，反而降低了一些。这可能是因为样本选择的随机性和过度学习造成饱和。在这一实验结果的基础上，建议标记数据和未标记的个数保持在　　左右的比例为宜。

4.5.5　乱序网包处理

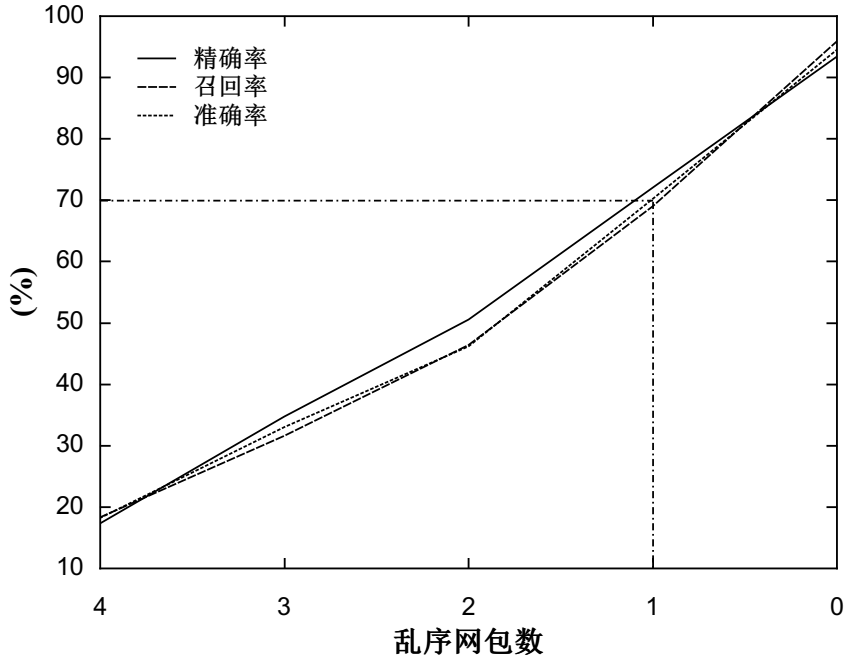


图　　不同数目乱序网包情况下的识别性能

乱序网包（　　）是大多数以机器学习为基础的流量识别方法难以处理的情形。因为无序的网包会导致分类特征的混乱。为了评估　　在处理乱序网包下的性能，进行了不同数量乱序网包情况的实验。由于乱序网包彼此不同，为了给出一个公平的比较，本文中认为乱序的网包将发生最坏情况（在传输中丢失）。一个实用的网络流量协议方法应满足即使信息不完整，也要尽量实现早期识别。实验中，随机丢掉前　个包中的若干个包，在精确率、召回率和准确率方面进行统计。

图　给出了乱序网包情况下的实验结果，从中可以看到，乱序网包带来了三个性能指标的下降。但　　仍然成功的保证了一定的识别率。很显然，乱序网包越少，识别的性能越高。当发生一个乱序网包的情况，　　仍然保持了约　　的平均准确率。另一方面，随着乱序网包的减少，　　性能几乎呈线性增加。这些结果表明在网包乱序的情况下，　　的性能仍然比较稳定。

4.5.6　混合机制

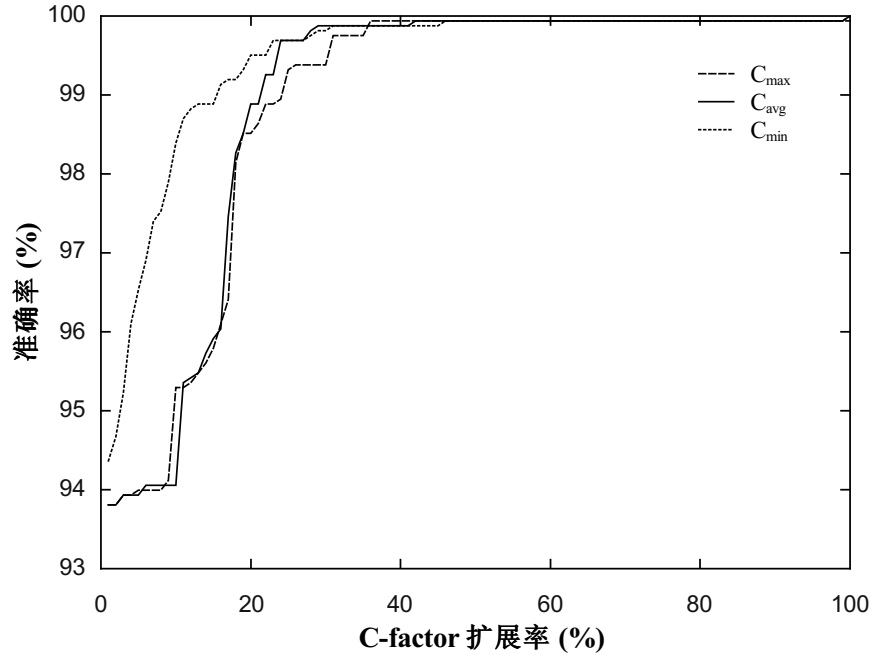


图　　不同可靠度系数下，混合机制带来的准确率

支持混合机制是　　另一个灵活性的优势。为了验证引入混合机制后分类准确率的提高，基于不同的可靠度系数的值进行了实验，实验中其它的慢识别器（　　）采用了基于内容检测技术。在 图　　中给出了实验结果。实验结果分别采用多种可靠度系数的定义，选取所有单独分类器中最小、平均和最大的距离作为可信度计算依据。横轴表明可靠度系数的放宽上限，例如，放宽　　意味着

阈值计算公式为 $\min(C)+25\%\times(\max(C)-\min(C))$。从图　　中，可以看到，引入混合机制确实提高了识别的准确率。当阈值为放宽　　时，几乎所有流量都被准确识别。

图　　中的结果也表明，基于最小的距离计算可靠度系数（C_{min}）情况下性能比平均距离和最大距离情况下都高，这与第　　节中的分析是一致的。所有的结果表明，混合机制有效地提高协议识别的准确率。

4.5.7　分类速度和更新速度

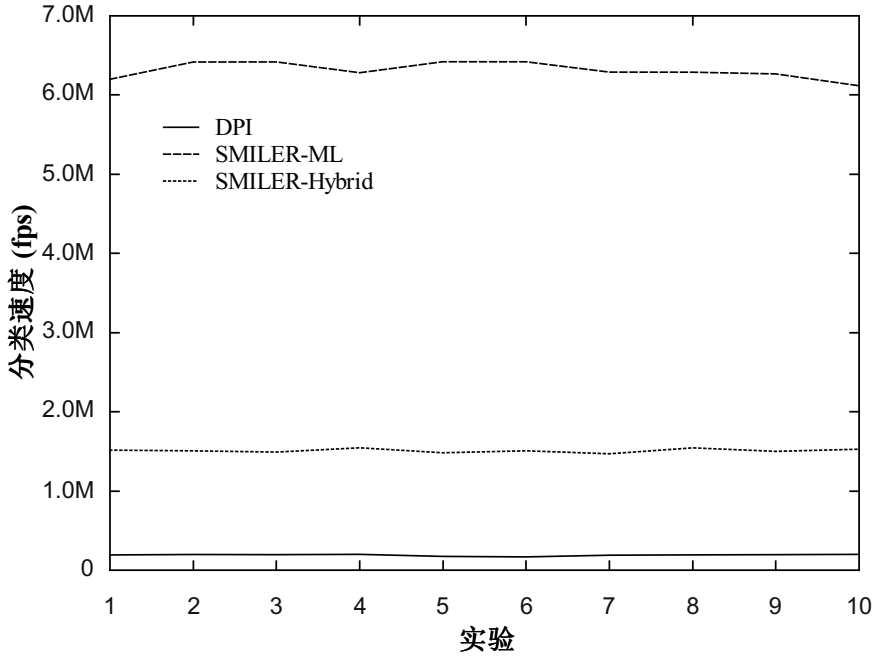


图　　分类速度比较

实验比较了传统的基于深度检测的识别技术、　　（纯机器学习）和　　（混合机制，　　的流量由　　进行处理）。为了估计典型的基于深度检测的分类器的处理速度，假设每个网包需要检测　个字节的数据，而每个流仅需要检测前　个网包，假设进行深度检测的数据结构只需要　　字节存储。这些假设对于当前基于深度检测的识别技术来说都是有利的假设。一共进行了　次实验，每一个实验运行了　秒的在线识别。采用每秒处理流数作为速度单位（　　）。

比较结果在图　中给出。图　说明　　（超过　　）和　　（超过　　）的识别速度都远远高于基于深度检测的识别技术（只有约　　）。这是因为　　并没有进行网包载荷的检测，这样的检测通常十分复杂，需要大量的时间和存储代价。图　中的结果说明

在识别速度上十分迅速。

同时，在实验中，　　预处理速度也非常快。例如，对　　个流的数据集进行训练（包括　　带标签和　　个无标签的），只需要小于　　秒。这比基于深度检测的方法（通常需要几十秒钟，大规则集甚至更多）要快得多。快速的预处理保障了　　在线（　　）更新的实用性。

4.6　本章小结

本章研究了网络流量管理与优化中的又一关键分类问题——应用层协议识别。在分析现有协议识别工作的基础上，本章剖析了实用的在线协议分类所需要满足的特性，提出基于半监督机器学习的在线协议识别系统　　。
仅检测网流的前五个网包的长度，利用半监督学习生成的分类器同时实现快速识别和准确识别。　　引入了混合机制来进一步提高　　的识别性能。

第

第 5 章　多链路故障容错的流量转发

5.1　本章引论

随着网络技术的飞速发展，互联网已经成为各类信息化应用的基础设施。部署在网络上的服务越来越多，对服务质量的要求也越来越高。互联网　尽力而为的特性试图利用不可靠的网络来实现网包的可靠转发。然而，只要短暂的链路故障，如几百毫秒，就可能会导致服务质量严重下降。现有被广泛部署的路由方案，如　　和　　，都需要全局的信息交换，链路故障恢复时间往往较长，难以满足实用需求　。

近些年，在集中式路由解决方案　　中，所有路由节点由一个集中式的控制器统一管理，一旦发生故障，由集中式的控制器重新进行路由计算。然而，路由器和控制器之间的信息交互，往往导致不可避免的恢复延迟。因此，非集中式的本地故障容错转发的方法更为实用。它可以提供在无集中式控制器情况下的快速故障恢复，也无需其它路由节点的辅助，可以满足严格的可靠性需求。现有的本地故障容错转发技术主要分为两类，需要特殊网包信息和无需特殊网包信息(参见第　章)。

需要特殊网包信息的机制，一般需要带有额外信息的网包，如转发的路径，并往往需要改写或删除这些信息。虽然这类方法可能提供良好的容错结果，网包格式的改变和修改网包信息意味着高昂的实际部署成本。因此，该类方法虽然理论上能够取得较好的保障性能，但在实际中往往会遇到部署上的诸多困难。

无特殊网包的机制，仅利用已知的信息，如目标　地址和网包到达端口等。这类方法无需额外信息，与传统的　路由模式最为兼容。然而，现有的方法只能处理单点故障，而在真实的网络环境中，在同一时间可能发生多个故障　。因此，该类方法面临的最大挑战是如何提高容错的可靠性能。

本章基于图理论，创新性的提出了偏构网络（　）模型和　回路。基于理论模型，提出了　　（　）转发方案，该方案通过发掘网络自身的容错特性增强转发的可靠性；另一方面，该方案不需要特殊的网包，较好地满足了实用需求。

5.2　问题分析

链路故障容错转发，是在网络中构建转发机制，使得当局部发生故障（例如链路中断）时能采纳备用的转发链路，将网络流量通过新的路径转发到目标节点。该项技术从实用的角度，主要面临下面四个方面的挑战。

- **可靠性高**。可靠性高意味着无论是单条链路故障还是多条链路故障，都要能实现容错，而且引入较小的额外代价。在单条链路故障的情形下，传统的方案往往采用提前计算的方法，针对每一种故障情况计算出备选的路径，并将备选路径存储到路由器接点的转发表中。这种方法虽然能较快的实现故障切换，但无法成功处理超过单条的链路故障。
- **恢复速度快**。恢复速度快意味着当发生故障时，新的转发行动需要及时生效，尽量减少网包的丢弃。采用特殊网包信息或者控制信息的方法在恢复速度上，由于受到需要额外的信息交互和计算过程的影响，往往会导致较大的恢复延迟，无法满足实用高速恢复的需求。
- **存储代价低**。存储代价是由于现有的路由器往往存在存储器件上的限制，大的存储器件往往需要昂贵的生产代价。由于存储的网络状态越多，往往意味着为容错带来更多的可选方案，所以存储限制也为可靠的链路容错方案设计带来了不小的挑战。
- **兼容性好**。现有的基于最长前缀匹配的路由方案被广泛部署，因此新的方案要能与现有方案兼容，满足实用需求。现有的路由转发方案是基于目标地址的转发。网包在从路由器入口到达后，需要在路由器内进行查表操作，查找出最长前缀匹配的表项，然后从对应出口转发出去。在整个路由转发过程中，路由表是静态的，没有额外的控制或交互信息。因此，基于传统路由过程，容错方案的设计也会面临兼容性的挑战。

为了同时解决以上四个方面的难点，本章认为基于无特殊网包机制进行设计更能符合实际需求的观点，并进一步的提出了创新的解决思路。在现有工作基础上，通过利用创新的网络模型——偏构网络来充分发掘网络拓扑自身的内部特性，通过巧妙设计静态转发规则来实现高可靠性、快速恢复，以及存储和兼容性上的平衡。

5.3　偏构网络

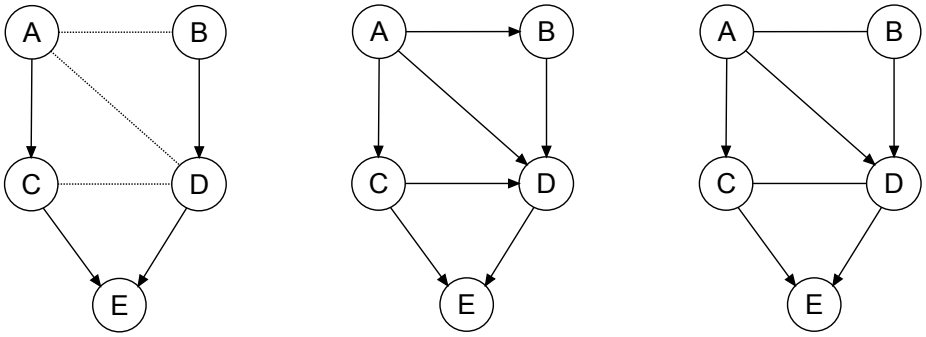
在本节中，首先介绍基本假设和偏构网络（　　）模型。之后，介绍一种具有较好的容错潜力的图遍历方法，　　遍历。最后，提出了多链路故障容错的路由问题和相关的定理。本节的目的是分析多链路故障容错问题，并且试图找到最大

化容错性能的方法。同时，为后续方案的设计提供理论基础。

5.3.1　基本假设

对于一个给定的网络 W，将网络看作为一个图模型 $G(V,E)$，其中 V,E 分别是指网络中的节点（或顶点）和边。对于 $e \in E$，若无特殊说明， e 是双向边。此外，假设转发过程中没有任何额外的标记或封装，所有流量都是基于目的　的路由转发，即标准的　转发。

5.3.2　模型建立



例子　　例子　　例子

图　　比较　　、　　和　　，其中　　是目的节点。

现有的路由过程基于不同的模型，如最小生成树模型（
，或　　）和有向无环图模型（　　，或　　）。对于每个目标，　　的基本思想是只利用部分的链路（位于最短路径上的），并设置为确定的指向目标节点的方向。而　　的基本思想是用到网络中所有的链路，并均设置为确定方向。本文中，在　　的基础上，提出了偏构网络模型（
，或　　）。　　也利用到了所有的链路，但只设置部分的链路为确定方向。下面给出偏构网络的定义。

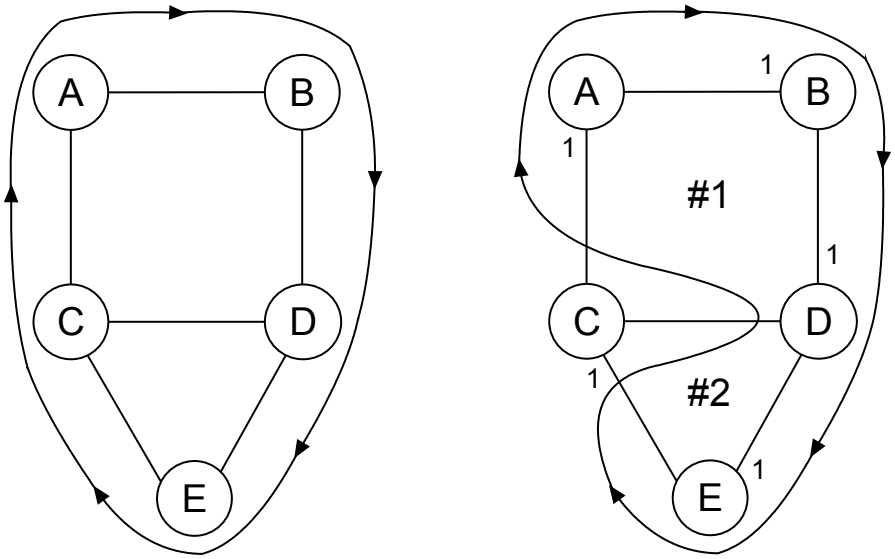
定义　　偏构网络 ： 对一个给定的网络，对每个转发目标，根据目标，偏构网络对网络中的部分链路选择一个确定的方向。

注意　　模型是根据每个目的地址生成的，因此，即使是对于相同的网络拓扑结构，对于不同目的地址，仍然会生成不同的　　。图　中给出了在同一个拓扑结构分别生成的　　、　　和　　之间的区别上。在图中，节点　　为目标节点。

以图　　中的　　为例，每个节点只采用了最短路径（以跳数为距离）上的链路。在　　中，由于每个节点只有一个出边，一旦发生失败，则该节点将丢弃转发的网包。在　　（图　　）的情况下，某些节点包括多条出边，这样一旦其中某条发生失败还可以迅速切换到剩余的出边上，从而更好的保护了网络转发的可靠性。例如，如果　　失败了，　　可以很容易地启用其它的链路，如　　从而快速恢复转发。但是，　　中仍然存在一些不足，某些节点只有一条出边。以节点　　为例，如果　　失败，　　将没有可转发的出边。为了达到更好的容错性能，　　的将某些边设置为无固定方向，或双向可用。在图　　中，　　和　　都是这种类型的边。当　　发生故障时，　　仍然是可用的。因此，与　　和　　相比，　　的故障保护具有更好的灵活性。当然，按照　　的路径进行转发，在链路出现故障后新的路径未必是最短的。然而，通过合理的构造　　，可以试图让这一路径近似最优。

实际上，将某些边双向使用并非新的想法，但仍存在尚未解决的问题，比如哪些边是无向的、对于有向边如何确定方向等。在第　　节中我们将详细讨论　　的构造过程，展示如何建立一个高效的　　来实现路由良好的容错性能，同时保持近似最优的转发延迟。

5.3.3　KF 遍历



遍历　　完全　　遍历

图　　遍历一个简单的例子

在这里，给出　　遍历的定义和相关定理。　　遍历显示了良好的故障恢复能

力，将被采用在后面的路由设计中。

5.3.3.1　定义

定义　　遍历　：对于一个无向图 $G(V,E)$，对任一 $v \in V$ 和 $e \in E$（注意：e 可以在两个方向上使用），　遍历访问 v 至少一次，访问 e 在任一方向上至多一次○。

对于一张任意图，有可能存在一个以上的　遍历。例如，图　显示了一个　节点图上两种不同的　遍历。图　中的　遍历生成一个的节点访问顺序　；而在 图　中产生的节点访问顺序为　。很自然的一个问题是，在任意一个图中是否始终存在　遍历。下面给出了一个存在定理。

定理　存在性定理　：对每个无向图 $G(V,E)$，如果 G 是连通的，则至少存在一个　遍历。

在这里，将其　遍历产生的访问顺序作为　环路。定理　揭示了　遍历的一般存在性。其实在一般情况下，对同一张图往往存在一个以上的　遍历。在所有的　遍历中，如果能访问到每个 e，则定义其为完全　遍历。注意，完全　遍历必须访问每个 e，而普通的　遍历则不需要。然而，无论是完全　遍历还是普通　遍历，都不能在同一个方向上访问同一个 e 超过一次。

定义　完全　遍历　：对于一个无向图 $G(V,E)$，一个完全　遍历是这样的一个　遍历，满足访问到所有的 $e \in E$。

以图　中例子为例，图　是一个完全　遍历，而图　则不是完全遍历。与普通　遍历的存在性定理类似，完全　遍历也有如下的存在性定理。

定理　存在性定理　：对每个无向图 $G(V,E)$，如果 G 是连通的，则至少存在一个完全　遍历。完全　遍历的访问路径构成完全　环路。

下面给出定理　和　的证明思路。

证明 对任意连通图 $G(V,E)$，假设 C_{el}、C_{kf}、C_{ckf} 分别表示欧拉环路、　环路和完全　环路。如果对于 $v \in V$，$degree(v)$ 是偶数，则对 G 必然至少存在一个 C_{el}　，让 C_{kf} 和 C_{ckf} 分别取欧拉环路，则定理成立。反之，如果对于某个 $v \in V$，

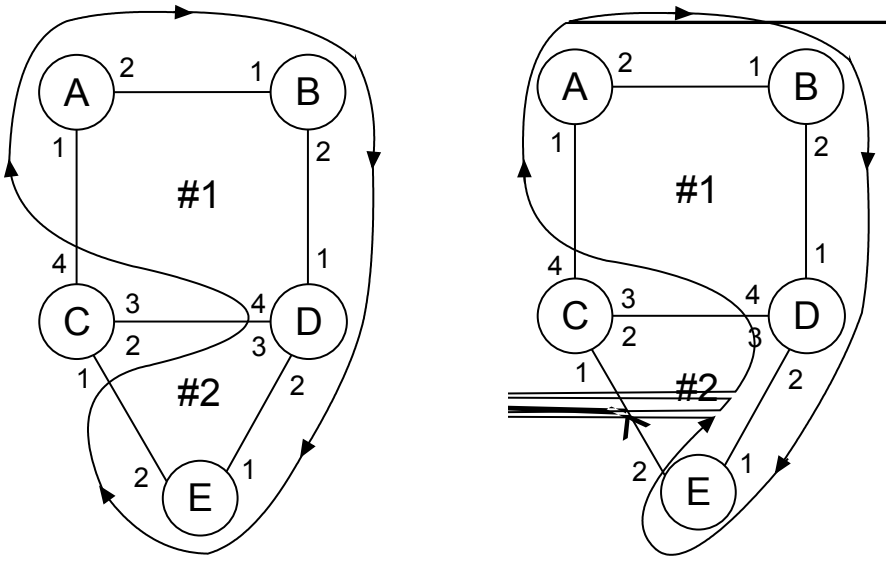
○　后文将　遍历和欧拉、哈密尔顿遍历进行比较。

$degree(v)$ 是奇数，让这些奇数度的节点组成集合 $V_{(odd)}$。那么集合 $V_{(odd)}$ 必然含有偶数个点，因为 $\sum_{v\in V} degree(v) = 2|E|$ 是偶数。因此，$v \in V_{(odd)}$ 可以两两组成对。则存在如下的引理成立。

引理　：对于 $V_{(odd)}$，必然存在一种对组合，使得对于对 i，构造以奇点对作为起点和终点的一条奇点路径 $p_i(i = 1,2\ldots,\frac{|V_{(odd)}|}{2})$ 可以被创建。满足对于 $i \neq j$，则 p_i 和 p_j 相互不覆盖。

实际上，引理　很容易得证。依次选取对，将路径上的点删除，使得剩余的图仍然连通，最终仅存在一对奇点即可。基于引理　，将每条奇点路径 p_i 上的边加倍，构造成为新图 G'，G' 必然存在新的欧拉环路，此环路是新的　完全环路。定理得证。 □

5.3.3.2　故障容错



完全　遍历　　故障情况下的完全　遍历

图　　完全　遍历显示了在故障情况下的容错性

由定理　，完全　遍历总是存在于连通图中。进一步的问题是对一个给定的图，如何找到一个完全　遍历。解决方法很简单。当图中没有环状嵌套结构时，利用深度优先搜索（　）产生遍历。否则，当遍历碰到新的环结构时，首先搜索新的环结构，完成后回溯继续原先的环遍历。

以图　为例。图中有两个环：#1 和 #2。假设从任意一个节点开始遍历，比如节点　，沿着顺时针方向遍历。当遍历沿着　→ →　的顺序到达节

点　　后，碰到了新的环（#2），这时尝试遍历新的环结构。在新环上的遍历（　→　→　→　）完成后，继续之前的环（#1）上的遍历。最后，所得到的完全　环路为　→　→　→　→　→　→　→　。这种基于　　思想生成的完全　　遍历被命名为基于　　的完全　　遍历。

可以看出，完全　　遍历显示了在故障发生时很好的容错特性。仍以图　　为例，按照每个节点的访问（到达或离开）顺序，每个节点的端口可以设置一个序号。以节点　　为例，访问的过程是访问　　和　　各一次，访问　　两次。分别将　　和　　的序号设置为　和　，而边　　的序号为　，　。这样每个节点可以获得其连接边的有序序列。例如，对节点　　是　　。通过循环这一有序序列，可以生成转发表，如表　所示。

表　　节点 D 的转发序列

到达	**号转发**	**号转发**	**号转发**

表　中给出了图　　中节点　　的转发表（以节点　　为目标节点）。第一列表示网包到达的链路，而后面各列表示按照优先级顺序的转发候选边。例如，第一行表项表示，　　是网包到达的链路，　　、　　和　　则是按照优先级排序的转发链路。

一个完全　　遍历会对其中每个节点产生这样一张的转发表。反过来，转发表也能生成原始的完全　　遍历。当一些链路如　　发生故障，遍历仍保持访问到每一个节点，如图　　所示，因此完全　　遍历有良好的故障恢复能力，可以有效保护节点。

作为总结，表　中比较了知名的汉密尔顿遍历　、欧拉遍历　和　　遍历以及完全　　遍历。表　中，第一列和第二列分别代表每个节点和边（两个方向）被访问的次数，而第三列是指在单一方向上的每个边被访问的次数。最后一列是遍历的存在性。从表　中可以看出，　　遍历更加灵活，它的存在性与拓扑无关，而汉密尔顿遍历和欧拉遍历的存在都依赖特定拓扑。

5.3.4　可达率

衡量路由方案的容错能力现有几种方法。例如保护节点的比例　。在本文中，选用了更精确的参数：可达率。假设 D, F_k 分别表示目标路由器和整个多链路

表　　遍历比较

遍历名	节点	边	单向	存在性
环路	$=1$			与拓扑有关
环路	$\geq$		分别是	与拓扑有关
环路	$\geq$			始终存在
完全　环路	$\geq$			始终存在

故障的故障空间，可达率（　　）$R_{W,k}$ 由等式（　）给出。

$$R_{W,k} = \frac{\sum_{F_k}\sum_{D} N_{delivered}}{\sum_{F_k}\sum_{D} N_{connected}}$$

其中 $N_{delivered}$ 意味着故障发生后仍能成功转发的路由节点数目，而 $N_{connected}$ 代表仍连通到目标节点的路由节点个数（目标本身是不包括在内）。同样的，不可达率（　　）表示为 $1 - R_{W,k}$。

很明显有 $0 \leq R \leq 1.0$。而较高的 R 意味着更好的故障容错能力。当 W 不连通时，没有路由框架可以保证这种情况下的可达性。因此，在本文中专注于网络保持连通的情况。

5.3.5　多链路容错路由问题

多链路容错路由问题的目标是在所有的故障情况下最大化 R。在这里，首先定义 完美容错性 和相应的定理。

定义　　完美容错性 ： 对任意的网络拓扑 $G(V,E)$，如果对所有的多链路故障情况，都能保证 $R = 1.0$，则对 $G(V,E)$，该路由机制提供了完美容错性。

因此，如果总是可以找到一个具有完美容错性的路由机制，总可以达到理想的 100% 的容错保证。然而，下面的定理意味着这是不可能的。以静态规则为基础的解决方案○ 并不总能保证完美容错性。

定理　　完美容错不可满足定理 ： 对任意的网络拓扑 $G(V,E)$，如果允许任意多条链路发生故障，则基于静态规则的转发无法一定保证完美容错性，即使 G 始终保持着连通性。

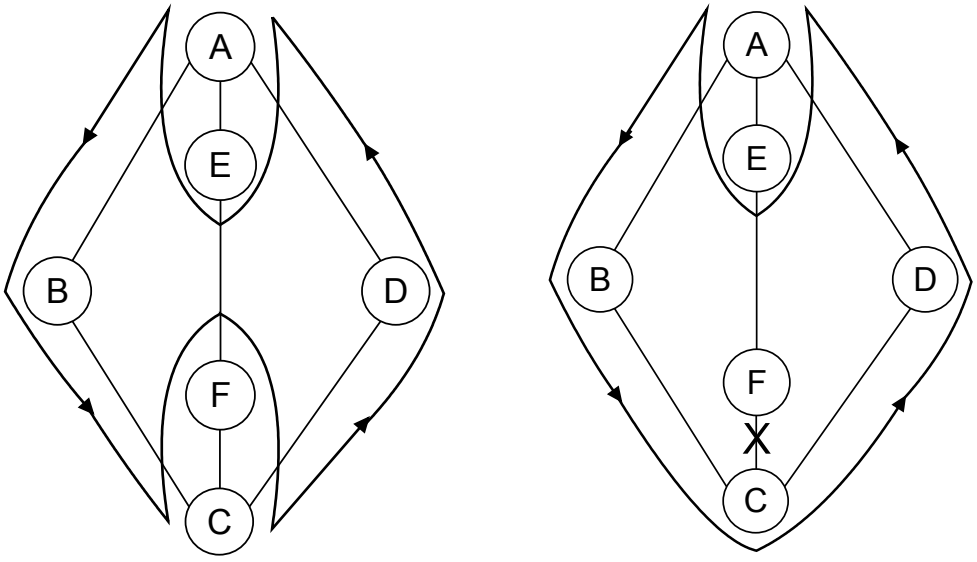
○　这意味着转发决定只由存储在路由表中的规则决定，预先计算后不会改变，除了静态规则外没有额外的控制信息。

定理　　说明对于某些网络，有可能存在一种不破坏连通性的故障情况，基于静态规则的路由方案不能总是保证多链路的故障下的容错性。定义这些拓扑结构为　非完美　的拓扑结构。

实际上，通过构造一个　非完美　的拓扑结构，则可以证明定理　　。很自然一个问题是具有什么特点的拓扑结构是　非完美　的。下面给出　非完美　定理，并进一步给出证明。

定理　　非完美　定理　：　对任意网络拓扑 $G(V,E)$，如果 G 中含有　非完美　的子结构，则不存在以静态转发为基础的路由，使得 G 在保持连通的多链路故障下总能保证完美容错性。

定理　　中证明了基于简单的本地路由规则无法保证多链路故障情况下的完美容错性。因此，在本文中，基于新的　　模型提出的　　路由，提供近似最优的容错性能，同时保证了兼容性的路由机制。



正常转发　　故障时转发

图　　非完美　定理证明用例

定理　　的证明思路如下给出。

首先给出引理　　。

引理　　：　基于静态规则的遍历无法保证在任意故障情况下都能满足全节点访问，全节点访问意味着到达节点至少一次。

为了证明该引理，图　　给出了一个示例。为了遍历到所有节点，该遍历必须从节点　　或者　　访问路径 $p=$　　，因此必须存在一个方向，使得 p 上

m 个节点被访问到，其中根据抽屉原理，$m \geq 2$（路径 p 包含　个节点）。假设遍历从　　访问了　个节点为　　，如图　　所示。则让边　　发生故障，则　永远无法被访问到，如图　　所示。同样的，假如从节点　　访问路径 p，节点　　将无法保证被访问到。一般地，当拓扑中包含有如图　中的环　环嵌套结构时，任意遍历都无法保证全节点上的访问。

证明　基于引理　，可以通过构造反例来证明定理　。考虑一张连通图 $G(V,E)$，其中 $|V| = N+1$，N 个节点组成一个辅助平面 L，L 含有环　环嵌套结构，而另外的节点 d 是目标节点。对于所有的 $v_i \in L(i = 1,2\ldots,N)$，边 $e(v_i,d)$ 连接 v_i 和 d。

假设让 $N-1$ 条边发生故障，例如让 $e(v_i,d)$ 故障（$1 \leq i \leq N-1$），注意到 G 仍然是连通的，因为 L 连通，且边 $e(N,d)$ 无故障。如果能保障　完美　容错，则每一个 $v_i \in L$ 都应该始终访问到 d，或者始终访问到 v_N，因为只有 v_N 连接到 d。

由于 v_N 是可以任意选取的，因此对于任意 $v_i, v_j \in L(i \neq j)$，v_i 必须能够访问到 v_j，或者说在任意故障后，还要保证 L 上的全节点访问，这与引理　冲突。定理得证。　□

5.4　基于偏构网络的容错转发方案

在本节中，首先描述基于入口（　　）的路由机制的基本思想，然后给出预处理和查找过程。之后通过剪枝优化提高性能。最后，分析了预处理阶段的时间复杂度。

5.4.1　基于入口的路由

传统的　路由的基本过程可描述如下：基于目标　地址，在路由表中查找，找到一个具有最高优先级（通常最长前缀）相匹配的条目。条目的行动域，通常是转发端口。

同样，这个过程可以表述为：目标地址 $\rightarrow$ 转发端口 ○。基于入口的路由，不仅利用目标地址，同时也查找入口端口。此外，行动域给出了所有的出口序列，并按照优先级顺序排列。这一过程可以表述为　目标地址　入口 $\rightarrow$ 排序后的出口序列　。

基于入口的路由在当前路由器架构下能取得更灵活的性能。大多数现有的路由器为了保持查找效率，在每个线卡中都存有一份相同的转发表　。与传统的路由机制唯一的区别是，基于入口的路由转发表表，在不同的线路中可能会有所不

○ 在　（　　）的情况下，超过　个转发端口可以被分配到一个目标地址，在本文中不涉及多径路由。

同。因此基于入口的路由所需的实际内存是传统路由机制的 d 倍（d 为网络中的节点度数）。此外，当网包来自一个入口时，只有相应的转发表将被查找，查找的域仍然只有目标地址。因此，查找的过程完全一致，复杂性并没有增加。考虑传统的　路由的状态数是 $O(N)$（N 是路由节点的个数），那么基于入口的路由的复杂度为 $O(d^2 \times N)$。在这里，没有采用基于源地址的路由机制，因为引入源地址信息后会导致在查找时候的额外代价。

5.4.2　预处理

基于入口的路由其预处理阶段包括三个步骤。首先，基于每个目标节点构造一个　　。然后在每一个　　，计算每条链路的优先级。最后，为每个路由器生成路由表。注意在计算中，每个目标节点彼此互不干扰，可以分开考虑，并分别进行预先计算。集中式的服务器或者分布式的控制协议都可以被采用以完成预处理环节　。

5.4.2.1　PSN 构造

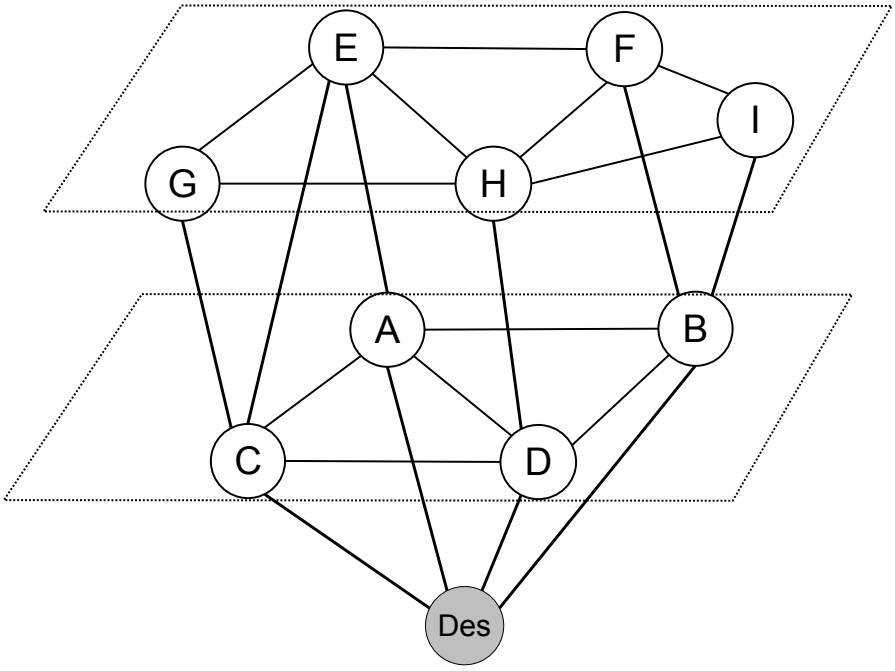


图　　构造　　示例

对于每一个目标，基于到它的距离（简单的跳数），每个路由器可以被标记一个距离值。根据这些值，在网络中的路由器，可以被划分为不同的层，这被命名为辅助层（

链路是主链接（　　）。那些从一个更高的辅助层指向较低的辅助层的主链路是下行连接（　　），而那些从较低的辅助层指向较高的辅助层的链路是上行连接（　　）。通过将每个主链路给定一个特定的方向，可以把每个路由节点作为目标建立一个　　。

图　中给出了一个简单的例子。在这个例子中，　　是目标节点。而节点　　和节点　　分别属于不同的辅助平面 #1 和 #2。链路　　是辅助连接而链路　　是主链路。每个主链路有两个方向。以链路　　为例，　→　　是下行连接，而　→　是上行连接。

在　　的构造中，每个目的地的路由，可以基本沿着主链路进行转发。当没有主链路时，采用辅助连接作为转发补充。

5.4.2.2　权值计算

由于路由器可能连接相同类型的几条连接（下行连接、上行连接或辅助连接），需要计算同一类型链路之间的优先级。优先级次序，可以用该链路到达目标节点的潜力来衡量。假设所有链路的转发质量是相同的，对连接到高潜力的邻居路由节点的链路应该有更高的优先级，因此首先需要计算每个邻居路由器的优先级。在这里，提出了三种基本的方法来完成优先级计算。

- **基于链路**。这种方法是比较直观的。认为具有更多的连接的路由器应该有更高的潜力才能到达目的地。因此，路由器的优先级，可以由它的所有链接的权值总和计算出。由于链路的类型可能会有所不同，包括主链接或者辅链路等，因此也需要对不同的链路类型设置不同的权重。例如，如果路由器　　拥有两个下行连接，而路由器　　只拥有一个下行链接，则路由器　　比路由器　　具有更高的优先级。
- **基于路径**。这种方法与基于链路的方法类似。其区别在于计算的依据是互不相交的路径权值，而不是链路权值。由于具有共同链路的路径可能会因为共同链路故障而同时失败，具有更多的不相交的路径的路由节点，显然拥有更高的容错潜力。
- **基于概率**。对于一个给定的网络 W，假设事先知道的每一个链路的故障发生概率，就可以计算出每个路由节点抵达目标节点的概率。这种方法基于概率论，比前两者更准确的反映了节点的容错性能，但是需要一个更复杂的计算过程。

5.4.2.3　路由表生成

路由表的生成与目标节点和入口相关。由于入口可以连接到高层、低层或同层的路由器。根据来源路由节点类型的不同，入口可以被分为上行入口（　　）、下行入口（　　）和同层入口（　　）。根据入口端口类型的不同，路由表的生成过程也有所不同。

上行入口。当网包从一个上行入口到来时，首先检查所有的下行连接。如果存在可用的下行链路，选择一个具有最高优先级的。如果没有可用的下行连接存在，检查辅助连接。如果没有可用的下行连接和辅助连接，则在所有的上行连接中采用智能选择算法进行选择。否则，从入口连接发回。

下行入口。当网包从一个下行入口到达时，首先利用一个在智能下行连接选择算法可用的下行连接中进行选择。如果不存在可用的下行连接，则需要挑选具有最高优先级可用的一个辅助链路。如果没有可用的辅助连接，检查上行连接。否则，原入口连接发回。

辅助入口。当网包从辅助入口到达时，首先尝试从可用的下行连接中按照优先级进行挑选。如果选择失败，则采用智能选择算法选择一条辅助连接。如果再次失败，则从上行链路中进行按照优先级进行挑选。否则，原入口连接发回。

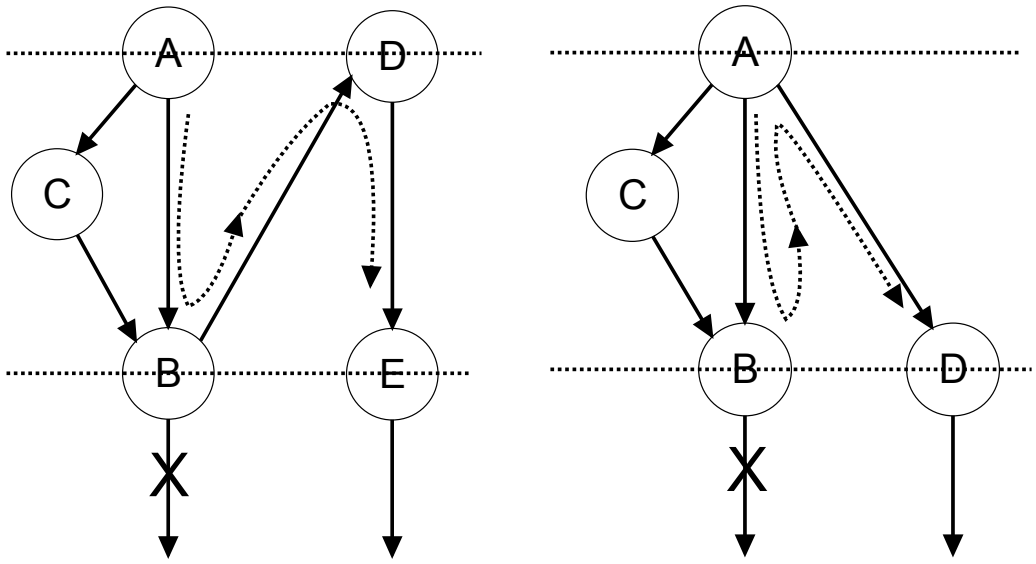
按照这些基本原则，可以为每个入口产生候选转发端口列表。对于本地的流量，在检查其目标地址后，从下行连接、辅助连接或上行连接中中挑选优先级最高的连接转发即可。下面将给出在下行连接、辅助连接或上行连接中进行智能选择的算法描述。

5.4.2.4　智能选择算法

在描述智能选择算法之前，我们首先给出关键路由器、关键连接和普通路由器、普通连接的定义。

定义　　关键路由器、关键连接和普通路由器、普通连接 ： 路由器是一个关键路由器，如果它有多于两条连接，或者它在　　中是目标节点，否则它是一个普通路由器。连接到关键路由器的连接是一个关键连接，否则它是一个普通连接。

普通路由器和普通连接可以作为转发中继，因此它们可以连接成一条链。通过一个关键连接或一条普通连接构成的链，最终可以找到一个关键路由器，这是该连接指向的真正终点。对于一个路由器上的所有连接，那些具有相同的真正终点的连接可以组合在一起，被认为是一组。根据这些定义，可以给出智能选择算法如下。



上行连接智能选择算法　　下行连接智能选择算法

图　　智能选择算法示例

- **智能上行连接选择**。当智能上行连接选择发生时，这意味着来自上行入口的网包，而且并不存在可用的下行连接或辅助连接，从而需要选择上行连接。其基本思想是试图转发网包到与入口连接不同的组中的连接，否则选择一个关键路由器。图　　中给出了一个例子。假设包从上行入口　　到达，并且　　没有可用的下行连接或辅助连接，从而　　试图选择连接　　，它属于不同的组。此外，如果　　也不可用，此时　　只有两个属于同一组的连接，然后　　将通过的一个关键连接转发网包，即连接　　。
- **智能下行连接选择**。当使用智能下行连接选择时，它意味着网包从下行连接到达，转发他们需要一个向下链接。如果下行入口连接的是一个关键路由器，这意味着没有通过这个关键路由器的路径，因此需要从不同的关键组中进行选择。图　　给出了这样一个例子。　　转发网包给　　，当　　的下行连接都失效，从而　　通过连接　　转发回去。如果　　从连接　　收到网包，　　即可知道　　没有路径到达目的地，因此，需要选择一个不同组的连接，或连接　　。
- **智能辅助连接选择**。不同于上行和下行连接选择，智能辅助连接选择需要依赖于完全　　遍历产生的转发矩阵。智能选择辅助连接发生时，意味着没有下行连接，从而在同一个层中应该尝试其它路由器，以检查是否某个路由节点有一个下行连接，以到达一个较低的层次。完全　　遍历提供了较好的容错性，以保证在辅助层的访问范围尽量大，从而实现较好的容错保护。

5.4.3　路由查找

基于生成的路由表，查找过程与传统的路由查找是相似的。假设网包从入口 i 到达，而在端口 i 上的路由表是 T_i。查找过程首先进行目标地址匹配，找到最好的匹配项，然后在行动域中的连接列表中检查，找到的第一个可用的连接进行转发。连接发生故障时，路由器就会从行动域中删除该转发连接。请注意，经过预处理后，所有的转发处理分布在每个路由器中，在本地没有任何控制机制，这与现有的路由器架构是兼容的。这种兼容性同时也保证了转发性能。查找过程在算法 中给出。

算法 5 路由查找过程

输入：

网包 *p*

入口 *ip*

输出：

出口 *op*

des　　*p*　获取目标地址

rTable　　*ip*

entry　*rTable*　　*des*

for *r*　*entry.result* **do**

　if not *r.isfailed* **then**

　　return *r*

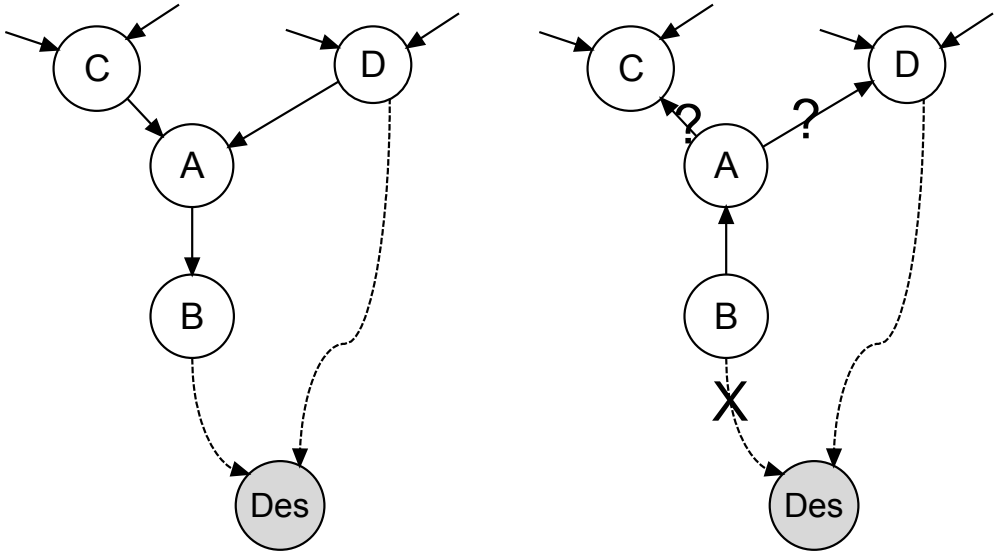
　end if

end for

5.4.4　上行连接剪枝优化

对上行连接可以进行特殊优化，实现更好的容错性能和更短的转发延迟。在这里，利用剪枝优化来消除一些故障情况下不必要的上行连接。

图　中给出了一个例子，其中　只有一个下行连接　和两条上行连接　和　。正常转发的情况显示在图　中。如果连接　→　出现故障，　将收到来自　的流量，然后　应该从它的两个上行连接（　和　）中进行选择。经过　和　，　连接到两个子网络。因为经过　的子网络并没有抵达目标节点的路径。因此，将连接　从入口连接　的转发连接列表中删除。剪枝优化后，转发过程中计算的复杂性和转发延



正常情形　　失败情形

图　　上行连接剪枝优化例子

迟都会降低。

5.4.5　路由更新

集中控制器可以轻松处理　　路由的更新问题。为防止网络的拓扑结构发生严重改变后导致原先的路由规则失效，可采取定期更新策略。特别地，当改变仅仅发生在某个辅助层时，只需要对同层和更高一层的节点进行路由更新。同时，为了避免可能出现的转发错误，应当采用低层优先的原则，在更新时首先更新较低层的路由器，之后依次更新更高层的路由节点。

然而，当网络的拓扑结构发生较大的变化，例如改变了多个辅助层时，可以通过重新生成　　结构来实现正确更新。下面将分析重新生成　　结构的预处理时间复杂度。

从　　的生成过程中可以看出，对于某个目标节点来说，网络中的每一条连接都需要进行常数时间的处理。而每个节点都可能作为目标节点以保障任意点可达的转发。因此整个预处理过程的复杂度，应该与网络规模成线性关系，为 $O(n)$，其中 $n = |V| \times |E|$ 是网络的规模。　　在预处理过程的线性复杂度，保证了该方案良好的可扩展性，即使在大型网络中也不会导致更新时间的爆炸，使得路由方案具有较强的实用价值。

5.5　性能评测

本节评测了路由的可达性能，转发延迟增加性能和预处理时间性能。首先给出评价的拓扑集和实验平台。

5.5.1　测试环境

表　　实验采用的拓扑

名称	拓扑名	节点数	边数	平均度

为了获得可靠的结果，利用从　　和数据中心中获取的真实拓扑结构进行了实验。每个拓扑结构的属性在表　中给出。

在所有的拓扑结构中，　　系列是从　　骨干网中获取的　，而数据中心拓扑，是企业网和数据中心中的常见拓扑　　。利用这些真实的网络拓扑结构，测试了网络中 k 条（$k=1,2,3$）连接故障时的性能。

评测平台是一个基于　　的　　服务器，运行的操作系统内核是　　，是　　，内存为　　。所有的代码由　　编写。

5.5.2　可达率

首先测试在一般的多链路故障的情况下，　　的容错可靠性。图　中给出了当 $k=1,2,3$ 时的可达率。可达率的计算参考式（　）。从结果中可以看出，对于所有的故障情况，　　路由始终保持一个很高的可达率，非常接近 100%。当发生单条链路故障时，在所有的数据中心的拓扑结构上实现了完美的保护，达到 100% 可达。在　　拓扑上，　个拓扑（　　）实现了 100% 的可达性，在其它拓扑上实现超过 99.8%（大多超过 99.9%）的可达率保证。

当 k 增大时，可达率仍然是近乎完美。当　个链路故障同时发生时，所有数据中心的拓扑结构上实现超过 99.99% 的可达率。在　　拓扑上的结果显示两个

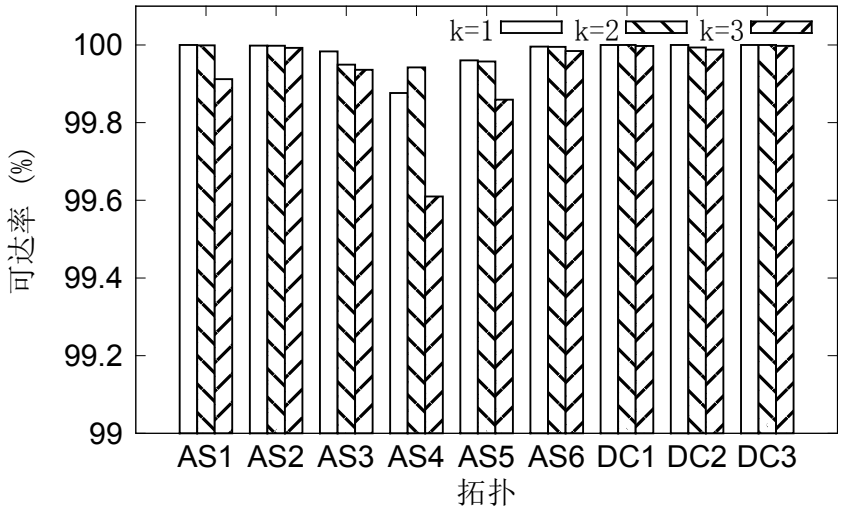


图　　k 故障情形下的可达率

有趣的特点：第一，即使在多条链路同时故障时，可达率保持接近 100%；其次，从单一故障到多故障时的结果是稳定的。这表明，　　路由能够处理复杂的故障情况。

5.5.3　转发延迟代价

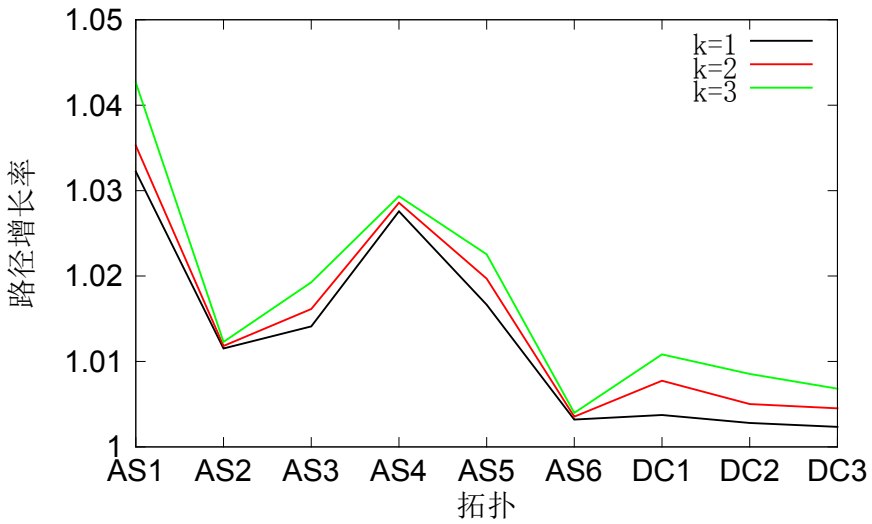


图　　平均转发延迟的增长

为了给出转发延迟的具体情况，比较了发生故障后的转发延迟和正常情况下最短路径转发的延迟比率。路径长度是网包从发出到达目的地址（本文中以跳数来近似计算）的距离。以路径延伸比作为评价指标，即　　路径和最短路径长度之间的比率。

图　给出了实验结果。在所有故障情况下，在所有拓扑结构上，　路由仅引入了微不足道的平均转发路径长度的增加（小于　）。这些结果表明，　路由实现了近似最优的转发路径长度。此外，随着 k 增加，路径拉伸比也增加。这是因为在更多的故障发生时，　路由将利用更多的后备连接，以保证成功转发，结果导致了转发的额外延迟代价。

5.5.4　预处理复杂度

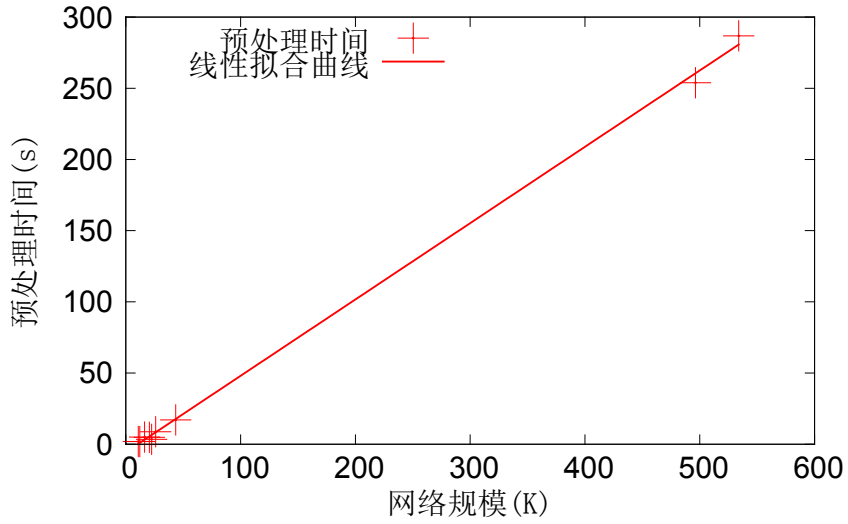


图　　预处理时间结果

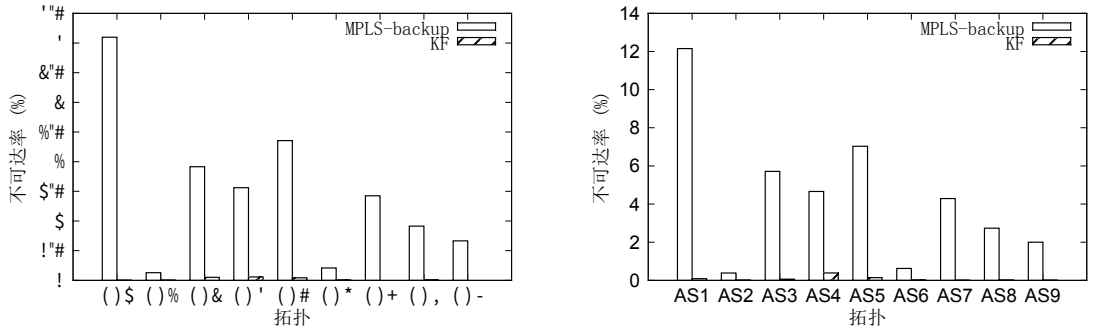
为了验证更新过程的有效性，评测了在所有拓扑结构上的预处理时间。特别地，对大多数的拓扑（　、　、　、　和所有数据中心的拓扑），预处理时间不到　秒。对非常巨大的拓扑结构（现有方法无法在可以接受的时间内完成计算），　的预处理时间代价也是可以接受的。在评测中，所有的计算没有任何并行处理加速优化。然而，由于不同的目的地址的路由计算自然是彼此不耦合的，简单地使用并行处理就可以提高性能。平均来看，对每个目的地的预处理，在不超过　秒的时间内就可以完成。在利用并行处理时，可以迅速完成全局的预处理。另一方面，预处理在真实场景中不经常发生，只有当网络拓扑发生严重变化，如大量的路由节点或连接故障时才需要预处理。然而，今天的骨干网络拓扑结构（包括　网络、数据中心网络和企业网络）在现实生活中往往不会经常有巨大变动。即使是大型　网络，一些拓扑变化频繁，但引发的路由更新并不迫切。例如，网络管理员可能需要临时改变链路状态进行维护操作等　。

图　中的结果表明，　路由的预处理展示了良好的线性复杂度。图中给出了基于网络规模（$|E| \times |V|$）的线性拟合曲线。校正拟合系数 $R^2 = 0.999$，这一结

果强力支持了预处理的时间复杂度是线性的。

5.5.5　与现有方案比较

5.5.5.1　与 MPLS 的比较结果



条连接故障　　条连接故障

图　　多链路故障时与　　的性能比较

首先与骨干网上常见的路由协议多协议标签交换（　　）　　的备份路径方案进行了比较。在　　的备份路径方案中，为每个单链路故障的情况下计算出一条备份路径。虽然多链路故障也可能发生，但对多故障的保护需要相当复杂的计算和海量存储，这对今天的路由器来说是不实用的。在发生链路故障的情况下，　　的备份方案，可以切换到相应的备份路径上，以保证容错性。

为了清楚地进行结果对比，以不可达率（　　）作为比较参数。少的不可达率说明更好的容错性得到了保证。比较结果显示在图　　和图　　中。所有的结果表明，　　的备份路由方案（空心）比　　路由（栅格）在不可达率上要高至少两个数量级。以拓扑　　为例，当两个故障同时发生时，对于　　备份路由方案，不可达率为 4.10%，而对于　　路由，只有不到 0.001%。

另外一个有趣的现象是，仅比较　　备份路由方案的结果，在　　和　　上取得了比其它拓扑结构更好的结果。从网络的度来看，平均度大的拓扑，比平均度小的拓扑（例如　　）的，结果要好。这意味着　　备份路由方案严重依赖网络拓扑自身的特性。而　　路由对于网络拓扑结构则不敏感。

5.5.5.2　与 SNH 比较

为了与近期最好的工作之一　　进行比较，也进行了保护节点率上的实验。保护节点意味着，从它发出的所有网包在所有的单点故障情况下可以达到目

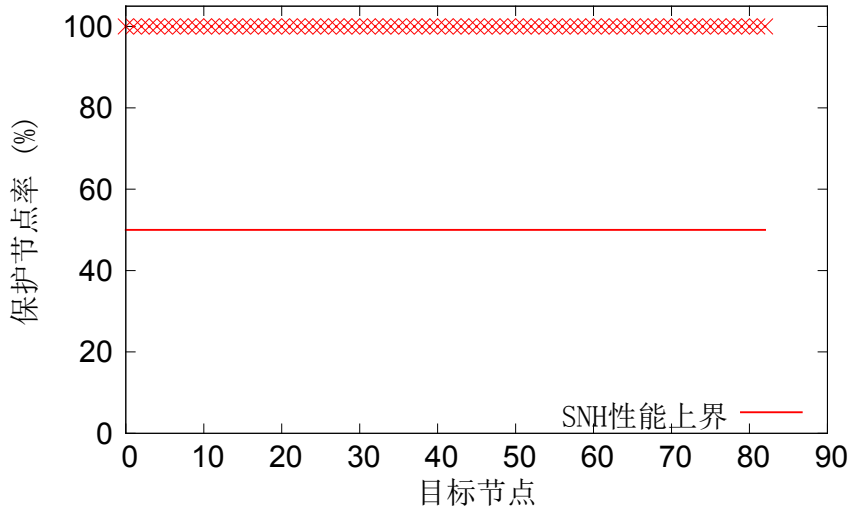


图　　在单连接故障时与　　在拓扑　　上的性能比较

的节点　　。图　　给出了比较结果。从图中可以看出，　　路由在每一个目的地址上对所有节点提供了 100% 的节点保护，而在基于　　的一个简化拓扑结构上（平均度为　　），　　方案节点保护的结果是小于 50%。很显然，　　也为所有节点提供了更好的节点保护。与　　方案相比，本文提出的方法只需要一个额外的信息——网包到达入口。另一方面，　　方案的预处理时间复杂度很高。例如，　　方案在基于　　的简化拓扑上需要近　个小时的计算　　（计算平台为　　处理器　　）。

5.5.6　结果讨论

可达性评价结果表明，　　路由是实用的多链路故障容错路由方案。即使在延迟敏感的情况下如现代数据中心，　　的较低的延迟代价也保证了它的实用性。此外，预处理过程的线性时间复杂度同样十分重要，尤其是在大型网络中应用时。

考虑图　　中的结果，由于测试拓扑结构的限制，有两个大规模的拓扑的结果远离其它七个较小的拓扑。这可能会影响最后的拟合结果。为了准确验证预处理性能，进行了补充实验，仅在七个较小的拓扑上进行了线性拟合。结果仍然十分接近线性（调整后的拟合系数，R^2 约等于　　）。未来在其它拓扑上的实验将提供更多的结果支持。

5.6　本章小结

链路故障容错的转发技术一直以来是学术界和工业界都积极关注的核心技术。由于实际应用需求在可靠性、恢复速度、存储代价、兼容性等方面的苛刻要

求，现有的方案难以满足真实的网络应用场景。

本文首次对一般多链路故障容错路由问题进行了专门研究，并提出了一个新颖实用的路由框架——　路由。　路由只利用本地静态规则进行转发，以实现故障发生时的快速恢复。　路由与现有的　网络路由架构兼容，不需要特殊的网包标签或额外的控制消息。在发生故障时所引起的转发延迟与正常的最短路径比，仅略有增加。此外，　与现有路由方案的相比，在存储代价上是线性关系。随着网络规模的增长，预处理的时间复杂度也是线性的。在真实的　和数据中心网络上的实验结果证明，　路由对多链路故障保证了高可达性和较低的转发延迟。此外，本文还提出了一个新的网络模型——偏构网络模型，并证明了相关的图论定理。

现有结果证明了　路由的可靠潜力，但仍然有一些有趣的问题值得讨论，其一是，什么类型的拓扑具有较高的　容错性？对这一问题的理论研究，将可以为网络拥有者设计具有良好容错性能的网络提供重要参考。另外一个有趣的话题是如何让　路由支持加权连接的网络，以及支持流量工程等。

第 6 章　基于负反馈的流量控制

6.1　本章引论

近些年，对网络流量的控制需求越来越强烈。特别随着云计算的兴起，大量的业务和服务都被迁移到数据中心中，对流量控制提出了更高的要求，如　　公司　　和亚马逊等　　。这些数据中心往往包括大量的服务器和网络交换设备。例如，在谷歌公司的数据中心中，超过　　台服务器一起合作，提供着全球搜索服务　　。一次使用谷歌的网页搜索请求可能需要访问数以千计的　　服务器（　　），并询问　　级的数据存储，大量的网络流量在数据中心内进行传输。过重的网络流量往往会导致不可避免的服务器故障。据观察，数据中心内的某些连接，往往会发生比其它链路更高的故障率　　。这些观察意味着数据中心中需要利用合适的流量管理技术来进行控制。

本章试图从分析流量管理的具体需求出发，基于控制理论，将反馈控制引入到流量管理模型中 在统一的模型指导下研究在复杂网络环境，特别是数据中心网络的有效管理机制。本章提出的动态反馈控制模型（　　，或　　），可以有效提高网络的整体性能，同时在保障了故障发生时的服务质量上也有较好的效果。

6.2　问题分析

对网络流量进行控制一直以来就是研究的热点问题。该问题主要指通过对特定的流量进行分发等管理手段，实现负载、业务等的调整，进而实现某种优化目标。该问题研究的基础是流量监测技术（包括分类、识别等），所面临的难点主要是如何设计合适的控制模型，采用恰当的控制手段。

传统的控制手段往往侧重于网络局部（如端到端）的性能优化，通过　　协议的参数调节，对网流发送速率进行调整或者对交换节点队列等进行优化　　，从而实现资源的分配优化等　　，这方面的工作主要面向对单个服务器进行服务质量的优化。也有部分工作面向整个网络服务质量引入了反馈控制的思想　　，尝试利用控制流量对分布式系统进行控制和管理。

然而，在数据中心中，传统的流量管理手段遇到了一些新的问题。首先，数据中心中的网络拓扑结构往往不同于普通的局域网　　，因而利用拓扑结

构的自身特性可能取得更好的控制效果。其次，数据中心对性能要求要远高于普通的网络环境，对性能的稳定性提出了很高的要求。例如，在高能物理和核物理（　　）社区数据中心，其带宽从　　一直到　　，海量的信息传输对稳定性要求很高。同时，因为数据中心的计算和存储设备在物理位置上往往聚集在一起进行集中管理，其系统模型比传统互联网更具有全局可控性，为采用集中式的控制模型提供了可行性。目前有一些工作　　已提出将管理层从交换设备中抽离进行集中控制。这些研究推动了流量控制研究的发展，比传统管理设备更易于部署，但智能的流量管理的仍然迫切需要合适的理论和模型。

6.3　基于反馈的控制理论

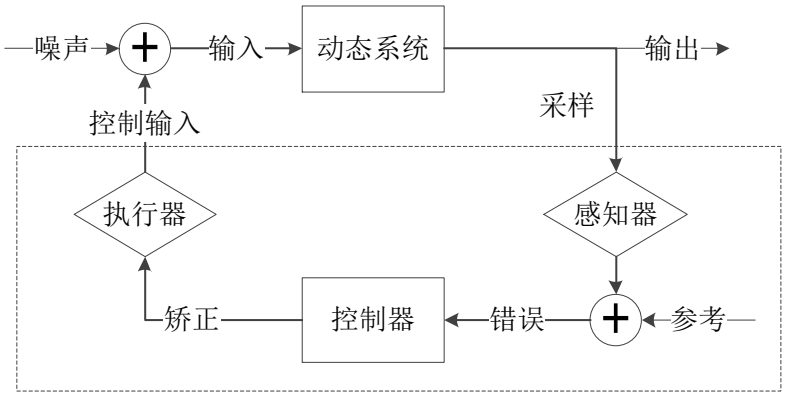


图　　反馈系统模型

基于反馈的控制理论　　往往适用于对动态系统的控制，如现代的数据中心环境。反馈一词意味着控制器在调整输入变量时候要考虑到输出信息，一般包括正反馈和负反馈。正反馈使得输出和输入相似，导致系统那个偏差不断偏大，可以起到放大控制作用；而负反馈则使得系统输出与目标误差减小，起到稳定系统的作用。本文采用负反馈，系统构成闭环机制。图　给出了反馈控制系统的基本模型，其中包含一个反馈控制环路和受控的动态系统（或控制对象）。反馈控制回路是反馈控制系统的主要组成部分，一般由三部分组成：**感知器**，**控制器**和**执行器**。受控的动态系统则是进行控制的对象，在本文中指数据中心。

感知器 感知器的主要任务是对控制系统的输出进行采样，并将采样结果传输到控制器。在数据中心环境中，可以使用一个计数器来统计各服务器队列的状态，反映每个服务器的流量处理的状态。

控制器 控制器将感知器的输出结果和参考输入进行比较，得到控制误差。之后，控制器采用预定义的控制算法，调整输入值并将调整策略传送到执行器。

执行器 执行器根据控制器的调整结果来产生信号，用以调整控制系统的输入。为了方便控制器管理数据中心内的网络流量，可以采用基于策略调整的交换设备作为执行器，如文献　　中提出的策略感知交换设备。

根据反馈控制回路的三个主要组成，反馈控制机制一般可描述为：控制器利用感知器检测控制系统的输出，并利用执行器来调整控制系统的输入。调整的策略需要根据系统的传递函数来进行设计。

传递函数可以利用拉普拉斯变换来进行描述　　。假设 $G_1(s)$ 代表原开环受控系统的传递函数，$G_2(s)$ 代表反馈回路的传递函数，则全局的闭环系统传递函数在式（　）中给出。

$$G(s)=\frac{y(s)}{v(s)}=\frac{G_1(s)}{1+G_1(s)G_2(s)}$$

式（　）中，$v(s)$ 和 $y(s)$ 分别代表控制系统的输入和输出。

从式（　）中可以看出，动态反馈回路在全局控制系统中相当关键，所以动态反馈控制的核心问题是确定动态反馈回路的各项参数。

6.4　动态反馈控制

在本节中，正式提出了基于反馈控制理论的　　模型。同时，为了解决多个服务器的动态控制问题，本文讨论并提出实用的自适应算法设计。

6.4.1　模型

不同于传统高性能计算中心采用昂贵和特殊服务器的做法，当前互联网数据中心，多利用通用服务器和交换机来获得更好的性价比　　。因此，数据中心内的流量管理所面临的两个首要要求是性能和可靠性。首先，从性能角度出发，一套实用的流量管理机制，可以动态分配数据流，进而获得更好的资源利用率和整体性能提升。另一方面，可靠性也是非常重要的，特别是利用不稳定的服务器来搭建稳定的数据中心，也需要智能的流量管理机制来完成流量迁移。这两点需求也正是　　模型的设计出发点。

目前，数据中心虽然可能会采取各种各样的拓扑结构，但往往都包含两个基本组成单元：服务器和互连设备（例如交换机），分别用来提供不同的功能。通用服务器用于处理数据和存储资源，而互连设备负责数据流的转发。由于大多数数

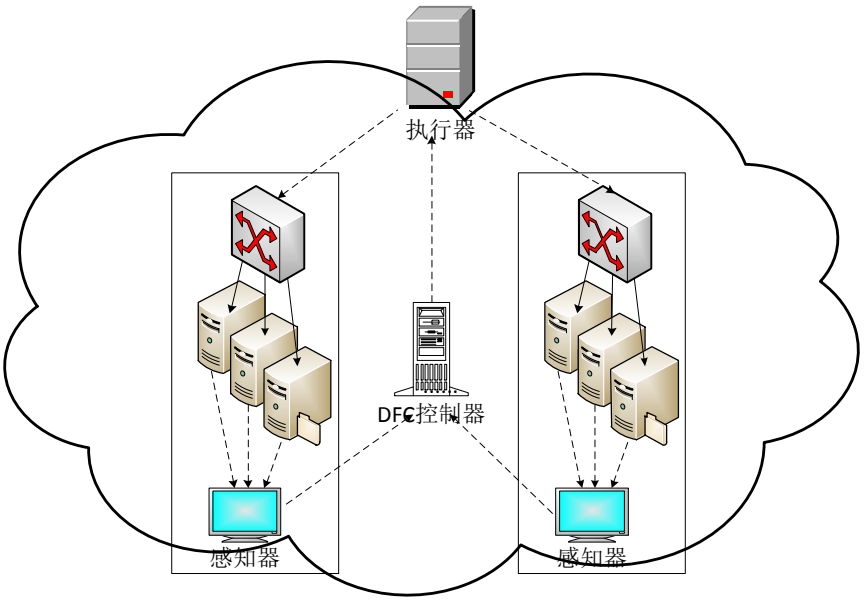


图　　基于反馈控制的流量管理模型

据中心都是分层构建，在最底层往往以功能块为基本单位。图　中给出了一个数据中心的　　模型。每一个功能单元中，利用一台感知器检测服务器的状态信息并发送到　　的控制器。　　的控制器则根据收集到的信息，智能化地使用自适应算法动态调整工作量。在这个模型中有两个信息通道：流量通道（实线）和控制通道（虚线）。下面将分别解释这两个通道的功能。

流量通道 流量通道包括互连设备和服务器之间的实际网络流量，它支持数据中心内的数据传输路径。流量通道在　　模型中是一个原始的开环控制系统，即受控对象。

控制通道 控制通道包括三部分：感知流量状态的感知器、智能控制器和策略的执行器。感知器部署在服务器上，以获得足够的机器状态信息，如　　的使用率、带宽和内存利用率等。策略执行器则根据从　　控制器获取的流量管理策略进行任务调整。

并没有直接控制各个服务器的状态。因为网络系统状态往往十分复杂，而且是实时动态变化，直接操作服务节点的状态将难以实现全局的优化和及时的调整。利用控制器可以仅仅对交换设备进行控制就实现全局化的优化目标，这种方式将更加灵活和方便。在现代数据中心中，交换设备的数量相对于大量的服务器来说要少得多。基于这个条件，反馈控制回路可以为数据中心提供更稳定的服务。特别地，当一台服务器发生故障时，　　的控制器可以将处理任务动态迁移到其它可用的服务器。

假如对功能单元中的一台服务器，输入数据流带宽是 $p(t)$，而处理能力为

$q(t)$，则队列长度为 $l(t)=\int_0^t(p(\tau)-q(\tau))d\tau+C$ （C 是常数），其拉普拉斯变换是 $L(s)=\int_0^{\infty}l(t)e^{-st}dt,(s>0)$。假设 $p(t)$ 和 $q(t)$ 为常数函数（例如，$p(t)=C_p,q(t)=C_q$），则有 $L(s)=\frac{C_p-C_q}{s^2}+\frac{C}{s}$。根据拉普拉斯变换 $p(t)$ 变换为 $P(s)=\frac{C_p}{s}$，进而可以得到功能单元的传递函数，如式（　）。

$$G_1(s)=\frac{L(s)}{P(s)}=\frac{C_p-C_q}{C_ps}$$

利用一个线性反馈环 $G_2(s)=-\frac{C_f}{s}$，则全局的传递函数可以由式（　）给出

$$G(s)=\frac{G_1(s)}{1+G_1(s)G_2(s)}=\frac{(C_p-C_q)s}{C_ps^2-(C_p-C_q)C_f}$$

在这里，得到的传递函数与线性反馈回路是面向单服务器的。而要解决在多个服务器模块中调整工作量的问题，需要设计控制器的自适应算法。

6.4.2　控制器设计

控制器决策了在不同服务器之间的工作流负载调整的原则。为了获得适当的控制算法，需要首先建立问题的数学模型。假设数据中心可以支持某项服务的吞吐量为 $Q(t)$，第 i 个服务器可以支持的吞吐量为 $X_i(t)$，则 $Q(t)$ 和 $X_i(t)$ 需要满足条件（　），这也是优化目标的边界条件。

$$\sum_{i=1}^{n}X_i(t)=\lambda Q(t)\geq Q(t),\lambda\geq 1$$

另一方面，对于一台服务器来说，过重的工作负荷将导致机器超载停机，而太轻的负荷将导致资源的浪费。给定一个动态变化的工作负载输入，一台服务器状态（包括　　和内存利用率和其它资源的使用率等）的优化，实际上是一个多种因素的组合优化问题。对单台服务器，最佳状态点可以预先给出。然而，对于多个服务器的情况则更复杂。因为每个服务器由于配置的不同，可能拥有彼此不同的最佳状态点。因此，总体思路是通过解决问题（　）来确定控制输入 $u(t)$。

$$F(\rho)=\min_{u(t)}\int_0^{\infty}\left((\rho-\overline{\rho})^TQ(\rho-\overline{\rho})\right)dt,\rho=(\rho_1,\ldots\rho_n)$$

其中 $\overline{\rho}=\sum_{i=1}^{n}\frac{\rho_i}{n}I_n$，目标为

$$\sum_{i=1}^{n}\rho_iC_i(t)=Q(t),i=1,\ldots n$$

和

$$\rho_i(t) = \frac{X_i(t)}{C_i(t)}, i = 1, \ldots n$$

在式（　）中，$\rho_i(t)$ 是指服务器 i 繁忙的程度，而 $C_i(t)$ 是第 i 个服务器在时刻 t 时的服务能力。基于式（　）到式（　）的结果，可以设计自适应算法，在多个服务器之间进行工作量的分配，如算法 所示。

算法 6　　调整算法

输入： $\rho = (\rho_1 \ldots \rho_n)$

δ　初始化常量

$time_{sleep}$　初始化调整间隔

$F(\rho)$　参见式（　）

while $F(\rho) > \delta$ **do**

$\rho, i = 1, \ldots n$

$\rho_{max} = GetMax(\rho_i)$

$\rho_{min} = GetMin(\rho_i)$

获取最空和最忙的服务器标号

$id_{max} = GetId(\rho_{max})$

$id_{min} = GetId(\rho_{min})$

调整工作量

$X_{id_{max}} \leftarrow \frac{X_{id_{max}} + X_{id_{min}}}{2}$

$X_{id_{min}} \leftarrow \frac{X_{id_{max}} + X_{id_{min}}}{2}$

$time_{sleep}$

$F(\rho)$

end while

自适应算法可以在所有服务器都正常的情况下进行优化调整。一旦出现无法正常工作的服务器，则需要通过控制器进行调整。如果服务器 i 失败，将引入特定的校正功能，如式（　）所示。

$$\rho_i(t) = 1, 1 \leq i \leq n$$

式（　）中的校正函数表示服务器 i 已经是满负荷，因此，不能再分配额外的流量。

为了建立一个通用的模型，本文中在保证功能完备的前提下尽量简化了数据中心架构。真正的数据中心可能包含不同层次交换机之间的复杂连接结构，这些

结构在逻辑上都是为了实现互联。在本文中，设计功能单元结构作为　　模型的基本单位，该模型可以得到更精确的数据。另一方面，本文中模型也基于一些合理的假设，如感知器需要尽量准确地感知服务器状态，执行器响应尽量灵敏等。

6.5　实验分析

在本节中，分别使用　　仿真平台和数据中心的原型系统来评价　　的性能。实验专注于两个方面的结果：可靠性和性能。

6.5.1　仿真结果

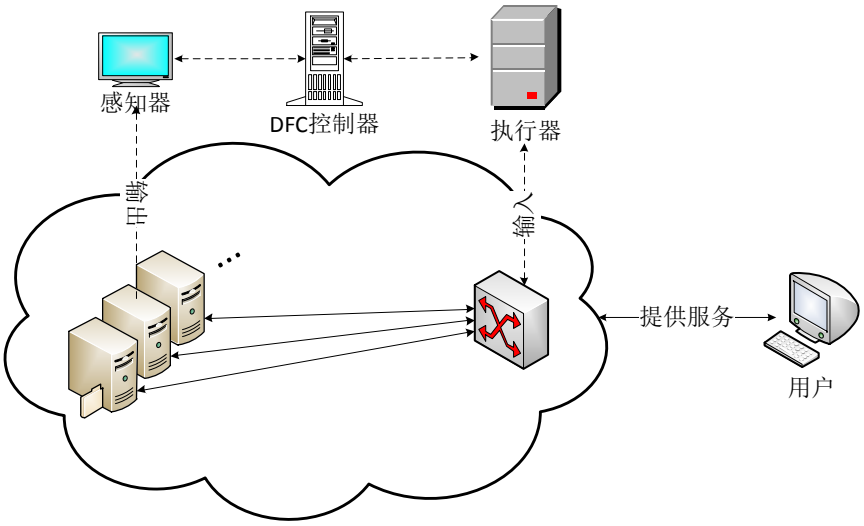


图　　流控下的数据中心模型

为了评估　　的可靠性，设计了基于　　平台　　的数据中心模型。该模型中包含若干功能单元，每个单元由　台服务器和一台交换机组成，如图　所示。控制器可以从感知器得到服务节点的状态信息，每　　秒刷新一次。

在实验中，参数的选择以验证为目的。设置每台服务器的最大吞吐量为　　的，而响应延迟是　毫秒，每台服务器处理队列为　　长度　　队列。从而每个功能单元的最大吞吐量为　　。实验中，让每功能单元提供　　的　　通信服务。实验测试在三种模式下数据中心的行为：正常　模式下（所有的服务器工作）、故障　模式（模拟过程中，有些服务器会发生故障，但无故障保护机制）和　　模式（故障机制，　　机制被启用来管理流量）。

图　中给出了仿真实验结果。其中三角节点曲线代表　正常　情况下的吞吐量，菱形节点曲线代表出现　故障　，而方形曲线代表的　　模式下的性能。从

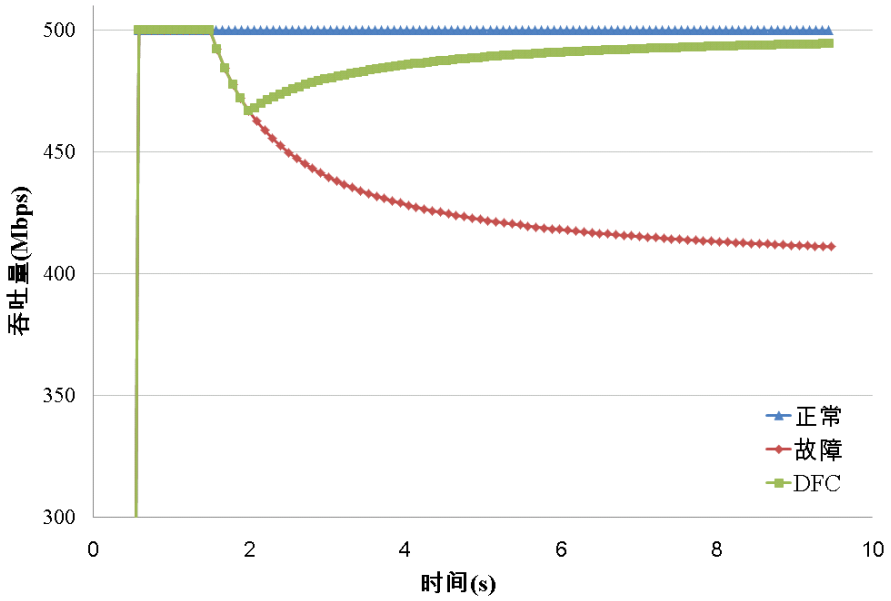


图　　可靠性比较的仿真结果

图中可以看到，　正常　模式下数据中心的功能单元从 0.5 秒到 9.5 秒提供了 500　　的正常吞吐量。　故障　时，吞吐量下降到了近 400　　（2 台服务器在　　秒时发生故障）。而在　　的　模式下，在发生故障的 1.5 秒时性能也会暂时下降，然而，由于　　的管理机制，吞吐量会迅速得到恢复。如图　所示，在 2.0 秒时恢复到接近正常的 500　　的吞吐量。实验结果验证了　　在检测出异常后采取行动的流量管理效果。

6.5.2　原型实验

6.5.2.1　发生故障时的响应

为了评估　　机制的有效性，设计了基于原型数据中心功能单元的实验。功能单元的拓扑结构与图　类似。每个单元由　台　　的普通服务器组成。每个服务器的　　是英特尔至强　　（　　通道，　核，　兆字节的二级高速缓存），存储是　　的　　内存。每台服务器可以支持　　左右的实时处理吞吐量，因此，每个单元总体提供约　　的吞吐量。在实验中，为了更准确地测试服务器的响应，让功能单元处于未满负荷状态，仅处理　　的流量，这远低于它可以支持的最大处理带宽。控制器的调整时间间隔仿真部分相同，为　　秒。

实验比较了在发生故障时的性能变化，进行了三种模式：　正常　、　故障　和　　模式下的测试。每种模式下，实验持续时间为　　秒。在　正常　模式下，所有服务器都可以一直正常工作。　故障　模式下，在第　　秒时，人工手动中断

其中一台服务器，　　模式下，启用了　　的流量管理功能，同样会在　秒时中断一台服务器。

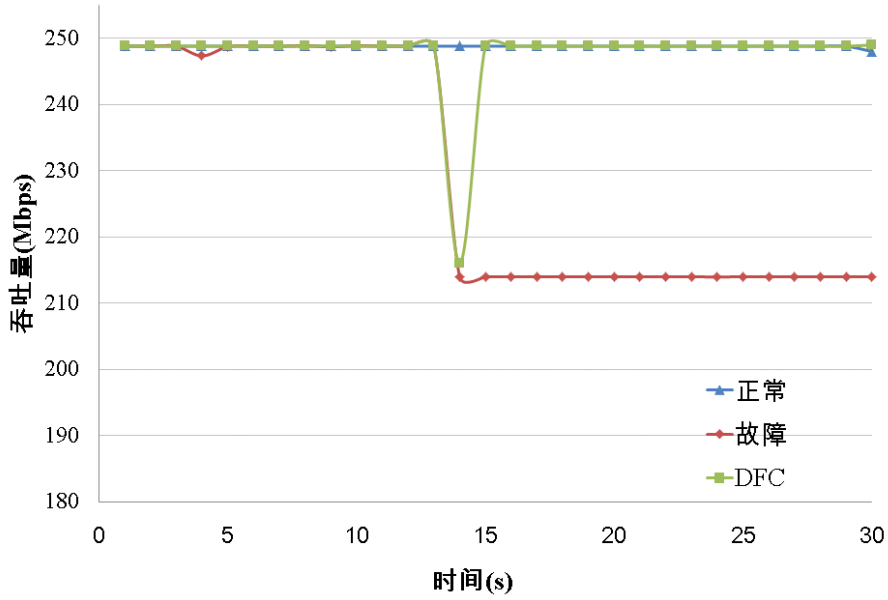


图　　可靠性比较的实验结果

图　中给出了三种模式下的吞吐量曲线。正常　模式下整个　秒内吞吐量保持在　　左右，这表明数据中心的功能单元工作正常。故障　模式下，吞吐量从　到　秒保持　　，但当一个服务器发生故障时吞吐量迅速下降到　　左右。在　　模式下，当有服务器发生故障时，吞吐量也暂时从近　　下降到约　　；但当反馈调整生效后，吞吐量在约　秒后恢复到正常的　　。

6.5.2.2　吞吐量比较

为了比较　　模型与传统的基于散列（　　）进行流量分配的算法性能，进行了吞吐量的性能比较。实验仍然采用了一个完整的数据中心功能单元，　台机器可以最大提供约　　的处理。

图　中给出了使用传统的基于散列的流量分配机制和　　机制的处理性能上的差异。基于散列机制（三角节点曲线）试图使用地址的哈希值来平均分发流量，而　　机制（菱形节点曲线）则根据服务器的状态进行动态分配。图　中，由于基于散列的方法不感知服务器的状态，只是简单的平均分配负载，导致部分较强处理性能的服务器无法满负荷运转，而部分若处理性能的服务器出现过载，整体的处理吞吐量仅为　　左右，低于理论值的　　。而　　模型

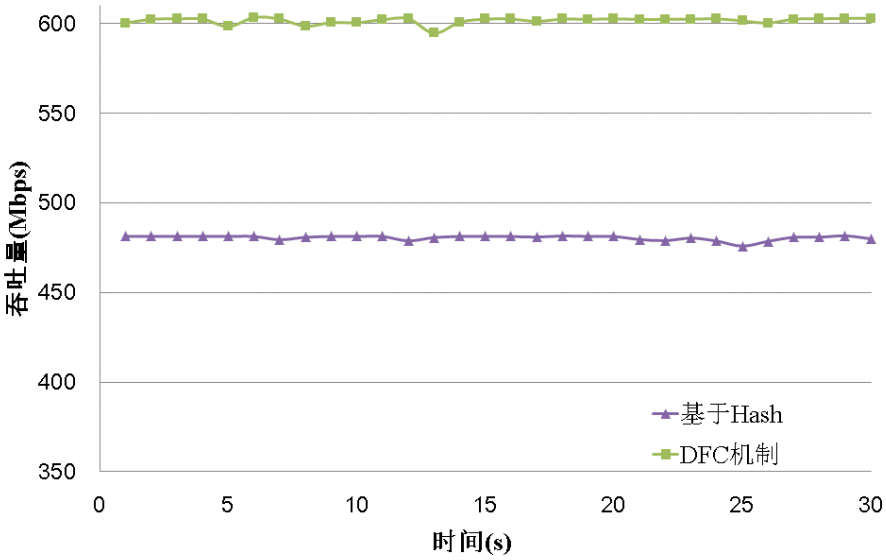


图　　性能比较实验结果

下，整体的处理吞吐量接近理论最大值。这一结果结果表明　　在没有故障发生时，也可以提高数据中心的最大处理能力。这是因为　　能更好的发掘每个服务器的处理潜力，最大化全局的吞吐性能。

所有实验结果表明，　　的模型为数据中心提供了可靠的保证，同时，还可以提高数据中心的处理性能。虽然　　的实验中，仍存在一些值得改进的地方。例如，如何设计响应灵敏的感知器等。但这些初步结果都说明了基于反馈的控制机制在提高数据中心性能的可靠性和吞吐量上的巨大优势。

6.6　本章小结

流量控制，一直以来就是网络管理员迫切需要解决的问题。基于流量分类的相关手段（网包分类、协议识别等），网络管理员可以及时地观测到网络的状态。基于这些状态，传统的控制手段往往侧重于网络局部（如端到端）的性能优化，而不能应对复杂多变的动态网络流量环境。

本文提出了基于动态反馈控制的数据中心流量管理模型　　。　　模型采用反馈控制回路组成的感知器、控制器和执行器来管理数据中心内的流量分配。为了提高数据中心的性能和可靠性，讨论了如何在多台服务器之间设计自适应算法取得整体的优化。仿真和实际实验结果证明，该模型可以有效地提高数据中心的性能和服务可靠性，尤其是当某些服务器出现故障时，　　能及时地迁移任务到其它可用服务器上。实验结果表明，基于反馈控制的模型在优化数据中心的流量管理方面具有很强的灵活性和实用性。

未来仍有一些有趣的课题值得研究。例如，如何设计响应灵敏的控制器件。在数据中心，对于视频会议和在线游戏等应用响应速度是十分重要的。在今后的工作将可以研究不同响应延迟环境下的自适应控制机制。

第 7 章　总结与展望

7.1　工作总结

计算机网络技术的飞速发展，给人们的生活和工作带来了诸多便利。然而随着各种网络应用的大量出现，人们对网络的需求越来越高，不可避免地产生了网络管理和网络安全上的诸多问题。一方面，网络服务提供商需要对网络流量进行检测和管理，以满足不同的用户需求，提高网络整体的服务质量。另一方面，网络安全设备需要依靠流量分类、分析等手段来识别网络流量中的恶意应用，进而利用访问控制技术和入侵检测、入侵防御技术等进行防护。

无论是从网络管理还是从网络安全的角度，网络流量管理与优化技术都是最为核心的技术。它通过网包分类、协议识别，有效分析网络流量中的业务和行为，为感知网络流量状态提供支持；基于网络流量状态感知的结果，通过可靠的转发设计和智能的流量控制，可以满足不同的网络服务需求，为管理网络提供了有效、可靠的手段。

本文研究工作系统地分析了当前网络流量管理与优化研究在检测和管理两个环节中的若干关键问题，包括网包分类、协议识别、可靠转发和流量控制，并分别提出了创新的算法和解决方案。最后，研究工作利用软件和硬件平台对所提出的算法和方案进行了全面地评测，进一步验证了所提出算法和方案的可行性。

本文的贡献主要体现在以下几个方面：

1）提出了基于动态离散位选取的高速网包分类算法

本文第　章提出了基于动态离散位选取的高速网包分类算法 D^2BS。该算法基于规则集内部的启发式信息，融合索引表结构的快速定位和线性表高效存储的特点，在实现高效存储的同时保证了高的分类速度。基于真实规则集上的实验结果表明，D^2BS 无论在时间性能、空间性能还是扩展性上都优于当前流行的其它高性能网包分类算法。

2）提出了基于半监督机器学习的实用协议识别技术

本文第　章提出了基于半监督机器学习的实用协议识别系统　　。通过分析实际应用需求并选择合适的分类特征，　　可以实现快速分类和早期识别。利用结合带标签训练数据和无标签训练数据的半监督学习机器，
实现了准确率上的优化。与目前广泛应用的基于深度内容检测的相关技术相比，
　　不需要检测网包载荷，在实现快速识别的同时保证了较高的准确率。另

外，对于发生网包乱序的情况， 也能正常工作。基于真实网络流量数据的实验验证了 系统的有效性。

3）提出了多链路故障容错的可靠流量转发方案

本文第 章提出了多链路故障容错的可靠流量转发方案 。 仅依靠本地的静态转发规则实现了对网络中多链路故障的容错。基于创新的偏构网络模型，充分发掘网络自身的容错性能，实现了高可靠的路由转发保障。同时 的计算代价和存储代价都是线性的，并保证了与现有路由架构的兼容性。基于真实网络拓扑上的测试表明， 实现了在多链路故障下接近 100% 的可靠转发，而引入的额外转发延迟代价平均仅有 2% 左右。这一性能远远优于基于 的链路故障保护方案。

4）提出了基于负反馈的智能流量控制模型

本文第 章提出了基于负反馈的动态流量控制模型 。该模型将负反馈控制引入到网络链路控制中，通过建立在服务节点上的感知器、流量分发节点上的执行器和控制器组件，能及时地基于网络流量状态进行智能地流量分配，实现全局的高性能。同时， 能自动将发生故障节点上的负载迁移到其它节点上，减少节点故障带来的损失。基于仿真和原型平台的实验， 展示出了在提供系统整体工作性能和降低故障损失方面的有效性。

7.2 未来工作展望

网络流量管理与优化是一个涉及到多方面的复杂问题，随着未来更多新环境和新需求的出现，仍存在多方面的挑战。一般来说，对流量分析越充分，则可实现的控制越灵活，但引入的性能代价往往也越大。因此，需要在实际应用中，根据具体的需求进行灵活设计。

未来的工作将包括如下几个方面：

1）融合不同粒度的流量分析方法

无论是网包分类还是协议识别，其目的都是基于网包的包头或内容进行流量分析，获取感兴趣的信息。而网络环境自身十分复杂，涉及到应用程序、主机、用户等多方面的因素。特别是数据中心等新网络环境的出现，使得传统的无智能感知的分析手段难以奏效。因此，未来需要融合多种粒度的分析方法，并将不同粒度的分析结果综合考虑，提供更准确的分析结论。

2）引入基于内容的深度检测技术

基于内容的深度检测技术可以尽可能全面地获取流量内容信息。因此，尝试引入内容检测技术将大幅提高流量分析所能获取的信息准确度。然而，传统的基

于特征串匹配的内容深度检测方法往往具有计算复杂度高、存储代价高等缺点。未来应该尝试采用智能方法，结合流量的统计特征来降低深度检测处理过程的复杂度，保证流量分析的实用性。

3）研究在未来网络环境中的新需求

面向未来网络的多种新兴网络技术的提出，如 、 、 、移动互联网、云计算数据中心等，在新的网络环境中都提出与当前网络所不同的需求。然而，进行网络流量管理与优化的目标仍将是一致的，即围绕提高对网络的感知性和控制性进行有效管理与优化。因此，未来的工作将考虑在未来网络环境中新出现的需求和挑战。

参考文献

致　谢

衷心感谢导师李军研究员对本人的精心指导。他的言传身教将使我终生受益。他不仅为我创造了良好的科研环境，在学习和科研中给予悉心指导，还帮助我争取到许多宝贵的国内外学术交流机会。李军老师开阔的视野、严谨的治学态度、渊博的专业知识和饱满的工作热情都将使我受益终生。

感谢清华大学微处理器与片上系统技术研究中心的薛一波老师、陈震老师、汪东升老师、客文洪老师以及清华大学网络安全实验室每一位同学，在学习和生活中给我的热情帮助和支持，特别是在科研工作中给我的宝贵建议和无私帮助。

感谢国家留学基金管理委员会公派研究生项目资助我一年的留学访问。在美国国际计算机科学中心和加州伯克利大学进行一年的合作研究期间，承蒙

教授、刘俊达学长等热心指导与帮助，不胜感激。

本课题部分工作承蒙国家　　项目资助，特此表示感谢。

最后，衷心感谢我的家人和女友在我求学期间对我的支持和鼓励！

声　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____ 日　期：_____

个人简历、在学期间发表的学术论文与研究成果

个人简历

年　月　日出生于山东省临朐县。

年　月考入清华大学自动化系控制科学与工程专业，　　年　月本科毕业并获得工学学士学位。

年　月免试进入清华大学自动化系攻读工学博士学位至今。期间曾获 清华之友——东芝奖学金（综合一等）、 中国电科十四所国睿奖学金（综合一等）、 清华大学优秀辅导员标兵、　　博士生英才计划（　　）　等荣誉。

发表的学术论文

源刊

收录

收录

收录

收录

杨保华，亓亚烜，薛一波，李军，　　结构在高性能网络算法设计中的应用，计算机工程与应用，　卷（　期），　页，　年　月（核心期刊）。

研究成果

杨保华，薛一波，李军。一种多域的网包分类的方法与装置：中国，　　（中国专利公开号）。

杨保华，李军。一种基于半监督学习的网络协议识别方法及系统：中国，　　（中国专利公开号公开号）。