本科毕业论文（设计）

基于集成学习的 CCSDS 空间链路层协议识别技术研究

RESEARCH ON CCSDS SPATIAL LINK LAYER PROTOCOL IDENTIFICATION TECHNOLOGY BASED ON ENSEMBLE LEARNING

曹鑫扬

哈尔滨工业大学

2024 年 5 月

密级：公开

本科毕业论文（设计）

基于集成学习的 CCSDS 空间链路层协议识别技术研究

本　　科　　生：曹鑫扬

学　　　　号：120L021508

指　导　教　师：韩帅教授

专　　　　业：通信工程

学　　　　院：电子与信息工程学院

答　辩　日　期：2024 年 5 月

学　　　　校：哈尔滨工业大学

摘　要

本课题来源于航天测控有限公司“基于 AI 的高级信号波形分析技术”项目，课题的实际应用场景为智能电子测试系统。 随着全球经济的持续增长、社会结构的深刻变革以及科学技术的飞速进步，航天科技作为国家战略的重要组成部分，其重要性在政治、军事、经济等多个领域日益凸显。与此同时，多样化的航天器平台导致了任务数据种类的爆炸式增长，进一步推动了航天任务功能需求的多元化。

国际空间数据系统咨询委员会，简称为 CCSDS(Consultative Committee for Space Data Systems)，其成立的主要目的就是为了使空间数据传输任务能够标准化，以此来加强世界各国之间在空间网络的合作和支持。CCSDS 提出了以分包遥测、分包遥控为核心的常规在轨系统 COS，以及在此基础之上的高级在轨系统 AOS。

本文在研究 CCSDS 体系的基础上，研究的主要协议为 CCSDS 链路层的 AOS 协议、TM 协议、TC 协议，研究它们的结构与功能，重点通过比较这几种协议的字段以及参考其它文献中对于通信协议识别方法的研究来研究如何更准确地识别这几种协议。此外，本文还对多种协议识别技术进行研究，主要包括机器学习技术以及集成学习技术，并探究它们的原理以及对于协议识别的优劣性。在链路层协议识别系统的实现过程中，本项目先对于协议识别的环境以及能够体现协议字段的数据库进行了构建，然后采用了随机森林算法、Adaboost 算法、GBDT 算法三种经典集成学习算法构建模型，并对数据集加入了一定的干扰因素，最后，本实验采用 F1-score 评价指标检验了识别协议模型的准确性，通过逐步加入噪声的方式，最终确定了随机森林算法在本环境下识别协议的性能更好。

关键词：CCSDS；链路层协议；协议识别；集成学习

Abstract

This topic comes from the "Advanced signal waveform Analysis Technology based on AI" project of Spaceflight Test Control Limited Company. The actual application scenario of the topic is intelligent electronic test system. With the continuous growth of global economy, the profound change of social structure and the swift advancement of science and technology, particularly in the field of space science and technology, as an important part of national strategy, has become increasingly prominent in many fields such as politics, military and economy. At the same time, the diversity of spacecraft platforms has led to an explosion of mission data types, further driving the diversification of space mission functional requirements.

The Consultative Committee for Space Data Systems (CCSDS) was established to standardize the transmission of space data in order to enhance cooperation and support for space networks among countries around the world. CCSDS puts forward the conventional on-orbit system COS with subcontract telemetry and subcontract remote control as the core, and the advanced on-orbit system AOS based on this.

Based on the study of CCSDS system, this paper mainly studies the AOS protocol, TM protocol and TC protocol of CCSDS link layer, and studies their structure and function. It focuses on how to identify these protocols more accurately by comparing the fields of these protocols and referring to the research of communication protocol identification methods in other literatures. In addition, this paper also studies a variety of protocol recognition technologies, including machine learning technology and ensemble learning technology, and explores their principles and advantages and disadvantages for protocol recognition. In the process of implementing the link layer protocol identification system, this project first constructs the environment of protocol identification and the database that can reflect the protocol fields, then adopts three classical integrated learning algorithms, namely random forest algorithm, Adaboost algorithm and GBDT algorithm, to build the model, and adds certain interference factors to the data set. Finally, In this experiment, F1-score evaluation index was used to test the accuracy of the identification protocol model. By gradually adding noise, it was finally determined that the random forest algorithm had better performance in this environment.

Keywords:CCSDS, link layer protocol, protocol identification, ensemble learning

目　录

第 1 章　绪　论

1.1　课题背景及研究的目的和意义

本课题来源于航天测控有限公司“基于 AI 的高级信号波形分析技术”项目，课题的实际应用场景为智能电子测试系统。随着全球经济的持续增长、社会结构的深刻变革以及科学技术的飞速进步，航天科技作为国家战略的重要组成部分，其重要性在政治、军事、经济等多个领域日益凸显。特别是在航天军事力量建设方面，已经成为各国竞相角逐的重要方向。与此同时，多样化的航天器平台导致了任务数据种类的爆炸式增长，进一步推动了航天任务功能需求的多元化。

卫星星座网络理念的早期构想可以追溯到 1945 年，当时由英国科学家和作家阿瑟·C·克拉克提出。他在他的文章《Extraterrestrial Relays》中首次详细描述了地球静止轨道上部署卫星以实现全球通讯的可能性。这一构想对后来的卫星通信技术和空间探索产生了深远的影响，并逐渐演变成了现代的卫星星座网络系统。阿瑟·C·克拉克的构想基于三颗地球静止卫星，这三颗卫星等距离分布在地球赤道上的地球静止轨道上。这种布局可以覆盖地球大部分表面，从而实现全球范围内的实时通讯。他的想法包括了使用卫星作为中继站来传输电视、电话和其他信号，这在当时是一种革命性的提议。1963 年，同步通信卫星项目发射了世界上第一颗成功的地球静止通信卫星，进而证明了克拉克的理论是可行的。随后的几十年中，多国和多个商业实体陆续建立了自己的地球静止卫星，用于通信、广播和数据传输。除了基本的通讯服务，地球静止卫星也开始被用于天气监测、军事侦察、全球定位系统等多个领域。这些应用展示了地球静止卫星在全球范围内数据和信息传输中的核心作用。进入 21 世纪，随着技术进步和成本降低，更多的非地球静止轨道卫星星座计划如 Iridium、Globalstar 以及近年的 Starlink 和 OneWeb 等项目开始实施，这些项目目的是提供更广泛覆盖、更高带宽和更低延迟的通信服务。[1,2] 这些卫星星座通过低地球轨道和中地球轨道的网络布局，提供了与地球静止卫星不同的服务特性，如全球高速互联网接入。

为了推动空间数据传输任务的标准化，加强国际间在空间网络领域的合作与支持，多国空间组织管理部门于 20 世纪 80 年代初联合成立了国际空间数据系统咨询委员会，简称为 CCSDS (Consultative Committee for Space Data Systems)，其成立的主要目的就是为了使空间数据传输任务能够标准化，以此

来加强世界各国之间在空间网络的合作和支持。[3] 天地一体化信息网络构建了一个全球范围的网络体系，包括天基的主干网、接入网和地基的节点网。这个网络系统与地面互联网及移动通信网络进行了互联，实现了全球性的覆盖，可以根据需求提供服务，随时随地接入，确保了网络的安全与可靠性。这一体系融合了天基网络和地基网络的优点，既发挥了天基网络广覆盖的优势，又利用了地面丰富的传输和处理能力，从而降低了整个系统的技术复杂度和成本。[4,5]

天地一体化信息网络对于空间数据传输任务具有重要的战略意义和实用价值，主要体现在以下几个方面：

（1）高效性和实时性：通过天地一体化信息网络，数据可以通过最快的路径传输，无论是从地面到空间，还是从空间到地面。这样的网络结构大大减少了数据传输的延迟，实现了几乎实时的数据交换，这对于灾害监测、环境监测等需要快速响应的任务至关重要。

（2）可靠性和稳定性：传统的单一通信路径，如仅依赖地面网络或单一卫星链路，容易受到自然灾害或人为干扰的影响。天地一体化网络通过多种通信平台的互补，可以在某一平台受阻时自动切换到其他可用通道，从而保证了数据传输的连续性和可靠性。

（3）全球覆盖能力：空间部分的卫星系统能够覆盖地球上的几乎每一个角落，包括偏远地区和海洋，而空中平台和地面基础设施则可以增强网络在人口密集区的性能。这种全方位的覆盖能力使得天地一体化网络非常适合执行全球范围内的数据收集和传输任务。

（4）数据处理和分析能力：天地一体化信息网络不仅仅是数据传输的管道，它还可以集成数据处理和分析功能。通过在地面、空中或空间平台预处理数据，网络能够优化数据流并减少需要传输的数据量，提高整体网络效率。

（5）灵活性和扩展性：随着技术的发展，天地一体化网络可以轻松集成新的通信技术和平台。无论是新增的低轨卫星还是更先进的无人机技术，都可以无缝接入现有网络，提供更广泛的服务和更强的数据处理能力。

通过这些特点，天地一体化信息网络为空间数据传输任务提供了一个强大、灵活且高效的解决方案，使得各种空间应用项目能够更加可靠和有效地执行。TCP/IP（Transmission Control Protocol/Internet Protocol）协议依赖于稳定的连接进行端到端的数据重传，因此对传输环境的稳定性有较高的依赖，如低误码率、低丢包率和短传输时延等。而在天基网络中，由于传输时延高、链路不稳定等因素，通常采用 CCSDS 协议族、DTN 协议等。因此，在天地一体化通信系统中，如何实现天基网络与地基网络的融合，确保异构网络间数据的高效

可靠传输，成为了一个亟待解决的问题[6]。

因此，卫星和地面通信设备的网关节点必须具备高效的空间链路层协议处理能力，能够适应具有高误码率的环境。CCSDS 的高级在轨系统标准，作为全球航天器用于空间数据系统的下一代技术框架，已获得多数空间机构和各国的认可和采纳。符合我国航天科技发展的战略规划，逐步实施这一标准在我国的空间数据系统中已显得尤为重要，这也符合国内在该领域发展的整体趋势。[7]

1.2 国内外在该方向的研究现状及分析

我国在空间通信领域正逐步采纳 CCSDS 所推荐的通信协议标准。CCSDS 为了满足日益复杂多变的空间任务需求，已经研制出一套完整的空间网络通信协议体系，并持续优化更新这些协议，以确保它们能更好地适应现代空间任务的要求。[3]近年来，全球范围内涌现出了众多先进且高效的协议识别算法。尽管天地一体化信息网络的概念在理论上具有广泛的应用前景，但根据现有的文献资料，当前的研究和技术发展往往还是集中在传统的地基互联网领域。这主要由以下几个因素导致：

（1）技术成熟度：地基互联网技术已经非常成熟，基础设施广泛建设，而空中和空间平台技术相对较新，仍在快速发展中。因此，研究和开发投入更倾向于已经稳定的地基技术。

（2）成本效益：地基网络的建设和维护成本相比于空中和空间平台要低得多。卫星和其他空间资产的发射和维护成本高昂，而无人机等空中平台则需要复杂的调度和管理。

（3）法规和政策限制：空中和空间领域的法规复杂，涉及国家安全和国际协议，这些都可能限制这些技术的发展和应用。相比之下，地基互联网受到的政策限制较少。

（4）技术难题：空间和空中平台面临的技术挑战也比地面复杂，例如信号的传输稳定性、环境因素的影响（如太空辐射、气象条件等）以及平台的长期可靠性。

（5）数据安全和隐私问题：随着数据传输范围的扩大和技术的融合，数据安全和隐私保护成为重要议题。地基网络相对容易实施更成熟的安全措施，而空中和空间网络的安全策略和技术尚处在发展阶段。

因此，虽然天地一体化信息网络潜在地提供了一种全新的通信和数据传输方式，能够极大地提高全球数据服务的可达性和效率，但其广泛实施和应用还需要解决上述问题，并且需要新的技术创新和政策支持。随着相关技术的进步

和成本的降低，预计未来这一领域将会有更大的发展和应用。[8]

传统的协议识别方法确实在很大程度上依赖于端口号识别，这是网络通信中一种较为基本且普遍的技术。端口号是网络地址的一部分，用于区分服务器上的不同服务。大多数网络服务在互联网上都有标准端口号，例如 HTTP 通常使用端口 80，而 HTTPS 使用端口 443。端口号识别方法是通过检查数据包的传输层头部信息来确定使用的协议。例如，TCP 或 UDP 头部会包含源端口和目的端口信息，系统通过这些信息来判断正在使用的网络服务类型。端口号识别的优点在于简单直观： 端口号识别方法简单易行，容易实现，不需要复杂的数据处理或额外的资源消耗；低成本： 由于其实现的简单性，这种方法不需要额外的硬件或昂贵的软件支持；速度快： 处理速度快，能够实时进行协议识别，适用于高速网络环境。但也有局限性，协议伪装或端口重映射： 黑客或用户可以更改服务的标准端口号，使得依赖端口号的识别方法失效。例如，某些应用可能会使用非标准端口运行 HTTP 服务；动态端口使用： 许多应用程序使用动态端口，这些端口在每次连接时都可能改变，使得基于端口的识别变得不可靠；加密和隧道技术： 隧道技术如 VPN 或 SSL 加密可能隐藏实际使用的应用层协议，使端口号识别方法无法准确识别数据包内容。

随着网络技术的发展和复杂性增加，单纯依赖端口号进行协议识别已不足以满足现代网络安全和管理的需求。因此，更高级的协议识别技术被开发出来：

（1）基于负载的协议识别：通过分析数据包的有效载荷（payload）来识别协议。这包括使用深度包检查 DPI（deep packet inspection）技术来检查数据包内容，识别特定的协议模式或关键字。

（2）基于行为的识别：通过观察网络流量的行为模式，如会话的时长、数据包大小、时间间隔等特征来识别协议。

（3）机器学习方法：利用机器学习算法训练模型，以自动识别和分类网络流量。这些方法可以根据已有数据不断优化，提高识别的准确性和效率。

通过结合这些高级技术，可以更准确地进行网络流量管理、安全监控和优化网络资源分配。随着 P2P 协议和加密网络流量的普及，它们不仅干扰了正常的网络秩序，还使得传统的基于端口的网络流量识别方法变得不再有效。特别是在 P2P 应用和加密流量上，基于端口号的识别方法准确率大幅下降。[9] 为了应对这一挑战，研究人员已开始采用基于内容的协议识别技术，这种方法通过详细分析数据包内的内容，来识别和区分各种不同的协议类型。不过，基于内容的识别方法面临着隐私保护和网络安全的挑战，因为很多协议数据都被加密，这制约了该方法的效用。此外，这种技术也显著提高了处理的时间复杂度，这

促使研究人员寻求新的技术方案继续进行研究。

为了改进协议识别方法的不足，研究者们提出了多种创新方法。2004 年，J.P. Early，C.E. Brodley，C. Rosenberg 等人的研究主要集中在利用网络流量的统计特征来识别网络协议。[10] 这种方法不依赖于传统的端口号识别，而是分析数据流的行为和模式，例如数据包大小、传输间隔和数据包方向等。这种方法的优点在于它可以适应动态端口和加密流量的挑战，使得协议识别更为灵活和鲁棒。他们的研究取得了超过 85%的识别率，这在当时是一个突破，表明基于流量特征的方法在实际应用中的有效性和可行性。第二年，Zander S, Nguyen T, Armitage G 等人在协议识别领域引入了机器学习方法。[11] 他们的工作利用了机器学习算法的强大能力来分析和学习流量数据的复杂模式，进一步提高了协议识别的准确率和效率。机器学习方法可以自动从大量数据中提取关键特征，这使得识别过程不仅限于人工预定义的规则，而是能够适应网络行为的变化和新协议的出现。这种方法的应用显著提高了识别的精确性和适应性，为网络管理和安全监控提供了更强的支持。这些研究成果为后来的协议识别研究奠定了坚实的基础。

近年来，基于流量的协议识别方法成为研究热点。金凌在 2011 年进行了面向比特流的未知帧头识别技术研究，结合了模式串匹配和频繁项集挖掘的方法。[12] 王和洲在 2014 年进行了面向比特流的链路协议识别与分析技术研究，但主要关注于地基网络的链路层协议。[13] 在 2016 年，张凤荔、周洪川等研究者的贡献是在网络协议识别领域一个重要的进展，尤其是他们开发的模型在零知识条件下进行比特流的协议分类。这项技术的成功主要体现在解决了聚类过程中遇到的核心问题，为机器学习方法在协议识别中的应用提供了新的方向。这项技术的应用对于网络安全监控尤为重要，因为它允许安全系统在不了解具体协议细节的情况下，还能识别潜在的恶意流量和异常行为。此外，这种方法也适用于云计算和大数据环境，其中数据的多样性和规模要求分类模型具有高度的自适应性和扩展性。[14] 同年 11 月，雷东，王韬等人分析了未知协议对网络安全的挑战，并比较了不同的未知协议格式推断方法。[15] 这些研究表明，协议识别的流量特征方法通过分析各种应用产生的流量的统计属性来识别应用类型，不需要详细检查数据包的内容。此方法即便在数据加密的情况下仍然保持有效。[16]

尽管近年来航天技术飞速发展，针对空间通信协议 CCSDS 的研究也在增多，但主要集中在性能提升方面，而关于空间协议识别的研究相对较少。[3] 李相迎等人在 2011 年研究了 CCSDS 协议中数据链路帧的识别技术，并提出了一种基于模式串匹配的命令链路传输单元盲同步方法。[17] 次年，姚秀娟、李雪等

研究者对 CCSDS 协议的识别技术的开发，提供了一个针对特定通信协议—尤其是用于空间通信的 CCSDS 协议—的定制化解决方案。这项技术通过深入分析 CCSDS 链路层协议的关键特征，并基于这些特征构建一个规则库来实现协议识别。首先，研究团队识别出 CCSDS 协议中链路层的关键特征，这些特征可能包括帧结构、同步头、控制字段等，这些都是协议识别中的重要线索。基于识别的特征，建立一个模型，该模型能够准确描述 CCSDS 链路层数据的特点。这个模型包括数据的物理和逻辑属性，以及它们如何在不同情况下变化。这种技术的开发不仅提高了在具挑战性环境中的协议识别准确性，也为类似的应用提供了一个范例，特别是在那些数据传输错误较为常见的场景中。通过这种方式，可以确保即便在数据质量受损的情况下，也能有效地进行数据通信和管理。这对于提高空间任务的数据处理能力、减少数据丢失和错误的可能性具有重要意义。此外，这种技术的实施也有助于优化数据传输协议，提高空间通信系统的整体性能和可靠性。[18]

从以上研究可以看出，当前，协议识别技术的研究主要聚焦于地面网络和互联网，并且通常关注网络层及更高层次。相比之下，对于天基网络中较低层协议的识别研究则相对较少，且 CCSDS 协议的识别相较于地基协议更为复杂。因此，研究 CCSDS 协议的空间链路层识别显得格外关键。

针对天地一体化通信系统中天基网络的高误码传输特性，开发和应用适当的技术和策略尤为重要，以确保数据的可靠传输和系统的稳定运行。天基网络通常面对的挑战包括极端的空间环境、长距离传输、复杂的信号衰减和干扰等问题，这些都可能导致数据传输过程中出现高误码率。本项目将深入研究空间链路层 CCSDS 协议体系，并借助集成学习方法的思想，设计一种基于集成学习的空间链路层 CCSDS 协议识别方法。该方法旨在实现高识别效率、强抗干扰能力等优势，为天地一体化通信系统的稳定运行提供有力支持。

1.3 本文的主要研究内容

该项目主要基于对空间链路层 CCSDS 协议体系及协议识别技术的研究和分析，设计一个具有强抗误码能力和高识别效率的协议识别系统，搭建该系统并进行实验验证该系统可行性，具体可以分为以下几个部分。

1.3.1 研究和分析空间链路层 CCSDS 协议体系与识别技术

具体来说，就是了解并学习 CCSDS 协议体系，包括应用层、传输层、网络层、链路层、物理层具体的结构、功能、协议、传输方式和它们之间的关系，

对于链路层结构、功能及其协议进行重点研究。在数据链路层，CCSDS 定义了两个关键的子层，分别是数据链路协议子层和同步和信道编码子层。[19] 这两个子层在空间数据系统中发挥着至关重要的作用，特别是在保证数据在恶劣的空间环境中可靠传输方面。对于数据链路协议子层而言，其主要负责确保在发送和接收设备之间可靠地传输数据帧。这包括错误检测和可能的错误恢复功能，确保数据传输的完整性和准确性。此子层使用的技术包括：

（1）帧同步：确保接收端可以正确识别发送端发送的帧的开始和结束。

（2）帧排序和重组：在接收端对接收到的帧进行正确排序和重组，尤其是在传输过程中可能出现的帧乱序情况下。

（3）错误控制：使用错误检测和纠错编码（如 CRC 编码）来检测和纠正在传输过程中可能产生的错误。

同步和信道编码子层处理的是如何在物理传输中尽可能有效地利用通信信道。这一子层包括以下关键功能：

（1）信道编码：通过增加冗余数据（如前向错误更正码），提高传输数据在恶劣信道条件下的抗干扰能力和错误恢复能力。

（2）帧同步：确保数据传输的同步，使接收器能够准确地识别出数据帧的边界。

（3）调制：将数字信号转换为适合通过物理介质（如无线电波）传输的形式。

CCSDS 在数据链路层的这两个子层设计确保了空间通信的高效性和可靠性，特别是在面对长距离和高干扰的空间环境时。这些协议的定义和实施对于各种空间任务，如卫星通信、深空探测任务以及地球观测等，都是至关重要的。

针对不同的传输需求和应用场景，数据链路协议子层在 CCSDS 框架下开发了多种标准协议。这些标准协议旨在优化各种空间通信任务的性能和可靠性，包括数据从地球到航天器、航天器之间，以及航天器返回地球的传输。以下是几种关键的 CCSDS 数据链路层协议：

（1）传输帧协议：这是一种非常通用的数据传输协议，设计用于通过各种空间通信系统传输数据。传输帧协议通过定义标准的帧结构，包括帧头、数据负载和帧尾，来实现数据的封装和同步。这种结构有助于接收方正确解码和排序接收到的数据，确保数据的完整性。

（2）高级数据链路协议：针对需要高吞吐量和高可靠性的任务，高级数据链路协议提供了增强的错误检测和纠错能力。这种协议常用于深空任务，其中传输条件极为恶劣，通信链路的稳定性和可靠性至关重要。

（3）无线链路控制协议：此协议专为无线空间环境设计，能够处理无线信号的问题，如信号衰减、干扰和多路径传播。无线链路控制协议通过特殊的编码和调制技术优化数据传输，提高数据在恶劣空间环境中的传输效率和可靠性。

（4）载波信号控制协议：这种协议处理的是如何管理和控制用于数据传输的载波信号。它涵盖了从载波的生成、同步到最终的调制解调过程。这对于保证在多种空间通信环境中数据信号的清晰传输非常关键。

（5）数据链路安全协议：随着空间数据传输安全性的日益重要，CCSDS 也开发了数据链路安全协议，以确保传输数据的机密性、完整性和可用性。这包括数据加密、完整性验证和身份认证机制。

这些协议的开发和实施，不仅提高了空间数据传输的效率和可靠性，还确保了不同空间应用在复杂多变的空间环境中的通信需求得到满足。CCSDS 的工作在全球范围内为空间数据通信提供了标准化的解决方案，有助于国际间的合作与数据共享。研究 CCSDS 以复杂的航天任务需求提出的常规在轨系统 COS（Conventional Orbiting System）协议标准以及在此基础上进一步提出的高级在轨系统 AOS（Advanced Orbiting System）协议标准及其对数据链路层提出的系统性的要求。在 COS 协议中，核心的部分包含遥测空间链路协议 TM（Telemetry）标准和遥控空间链路协议 TC（Telecommand）标准等标准，重点研究这些协议标准对于数据链路层提出的具体的、系统性的要求。在此基础上针对包括 CCSDS 协议在内的所有协议链路层的数据的一般特征进行研究和分析，以便于后续对于协议识别技术的研究。

具体来说，就是了解并研究当前主要的协议识别技术的分类以及不同协议识别技术的特点，当前协议识别技术主要分类如图 1-1 所示。

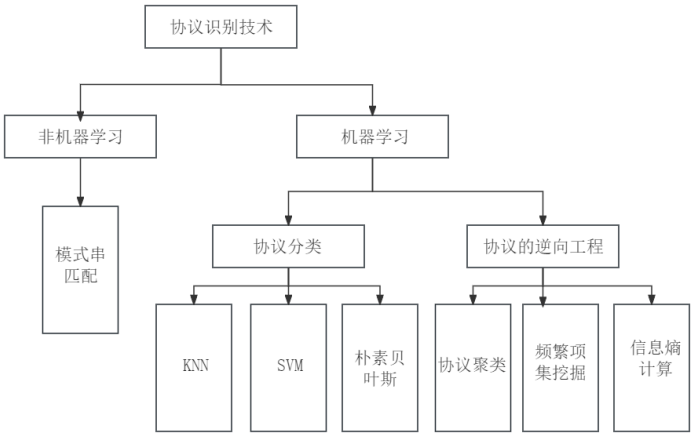


图 1-1　协议识别技术主要分类

链路层作为网络架构中的基础层，其主要功能是把来自网络上层的协议数据包装成帧，以便在硬件设备之间传输数据。当数据需要在设备之间传递时，发送端的网络层数据包会送到链路层进行额外的处理。在这一层，数据包将被附加上帧的头部和尾部，从而完成封装过程，形成完整的数据帧。封装完成后，数据帧会被传递至物理层，并最终通过物理媒介（如光纤、电缆等）以比特流的形式发送出去。

在接收端，物理层首先捕获这些比特流数据，并逐层向上传递至链路层。链路层在接收到比特流数据后，会根据帧头部和帧尾部的信息，对数据进行准确的解封装。这个过程确保了数据帧能够完整无误地从发送端传输到接收端。

在链路层，数据帧由固定字段和可变字段两部分构成。固定字段包含的是在每一帧中位置和内容都保持不变的信息序列，通常用来标记帧的种类、长度等重要信息。相对地，可变字段包括的是那些在帧中其内容或位置可能会变化的信息序列，这些字段主要用于装载具体的数据内容。

为了识别链路层的协议，关键任务是能够从大量的比特流中辨识出那些规律性的固定字段。这些固定字段往往具有特定的模式和特征，可以通过算法进行匹配和识别。一旦成功识别出固定字段，就可以进一步确定数据帧的类型和格式，进而实现对链路层协议的准确识别。

本项目对基于机器学习的协议识别方法进行了学习和研究，重点研究机器学习中如何选择适合训练的特征子集和学习模型，并对于针对机器学习提出的一些具有代表性的算法进行研究，分析这些算法的优缺点。在研究这些协议识别技术和算法的过程中，注重不同协议识别技术和算法的对比和对照，在查阅资料的过程中，注重分析为什么曾经提出的一些算法如今已经不适用了。最终，在对以上协议识别技术和算法的研究基础之上选出并采用适合本项目的协议识别技术和具体算法。

1.3.2 研究和分析协议数据特征并设计协议识别方法

首先，本项目对 CCSDS 协议数据的特征进行学习和研究，对于前面研究过的 CCSDS-AOS 协议、CCSDS-TM 协议和 CCSDS-TC 协议等 CCSDS 中主要的协议，协议的字段是有差异的，本项目需要了解这些协议字段的差异并能找出好的方法进行差异化地识别。然后，本项目对主要的识别技术进行研究和对照分析。本项目重点对集成学习的方法进行分析。

地基网络要能识别出天基网络发送的信号和指令，这时就需要对天基网络的协议做相应的识别，而天基网络由于高误码率的传输特点，传统的协议识别

技术，比如模式串匹配技术，就不适用于这种场景。因此，本项目采用集成学习的方法来识别协议，因为其具有较高的抗干扰能力。

集成学习模型的识别准确率以及识别的效率较高， 在集成学习中，本项目将研究目前具有代表性的 Bagging 思想和 Boosting 思想以及针对这两种思想提出的几种具有代表性的算法，研究这些集成学习思想与对应算法的优缺点和适用性。并尝试利用这两种思想中选取出来的比较合适本项目的算法，以及对于定长帧协议识别系统和变长帧协议识别系统的研究和设计，设计基于集成学习思想的空间链路层 CCSDS 协议识别模型，在对不同模型的性能通过实验进行比较后选取出抗误码能力更强、更高效的协议识别模型。

1.3.3 搭建系统模型并得出结论

本项目成功构建了一个针对空间链路层 CCSDS 协议的识别系统，通过精细调整模型参数，确定了最佳配置，并进行了实验以验证系统的有效性。在实验开始前，本项目对链路层协议识别系统的整体流程进行了详细的规划。实验的核心环节包括数据的前期处理和识别数据模型的搭建。数据预处理参照了 CCSDS 蓝皮书对于 AOS 和 TM 帧字段的规定以及填充方法，模型构建采用了随机森林算法、Adaboost 算法、GBDT（Gradient Boosting Decision Tree）算法三种经典集成学习算法构建模型，并对数据集加入了一定的干扰因素，最后用一定标准检验了识别协议模型的准确性，通过逐步加入噪声的方式，最终确定了随机森林算法在本环境下识别协议的性能更好。

在实验验证的过程中，先对于实验数据进行研究与分析，然后对于实验环境和实验的评价指标进行研究，并确定适用于本实验的实验环境和评价指标，本实验采用 F1-score 评价指标。在通过对构建模型的大量数据进行实验验证与分析后，能验证本项目采用的协议识别方法具有较强的抗干扰能力和较高的识别效率。

第 2 章　CCSDS 协议体系与识别技术研究

2.1　CCSDS 协议体系研究

CCSDS 是一个非常重要的国际组织，成立于 1982 年，旨在为各种空间通信协议提供标准化，以支持国际空间飞行和地面系统的数据和信息通信。CCSDS 由多个成员国的空间机构组成，包括美国国家航空航天局（NASA）、欧洲空间局（ESA）和其他国家的空间机构。制定和维护一系列技术标准，以确保全球空间任务的数据通信互操作性。这些标准涵盖了从地面系统到空间载体的各种通信方面，包括数据格式、通信协议和网络架构。从 1982 年起发布和采纳了一系列标准化的系统技术建议书，这些建议书不仅具有前瞻性，而且充满创新性，近百份建议书中包含的新概念系统和技术不断推动全球空间数据系统领域技术的快速发展。

2.1.1　CCSDS 协议体系结构

CCSDS 的协议标准体系采用了类似于地面网络的五层架构，包括应用层、传输层、网络层、数据链路层以及物理层以提供一种系统的方法来处理空间数据通信的复杂性。[20]，如图 2-1 所示，这种层级结构确保了数据传输的条理性和高效性，这种分层架构不仅有助于标准化各种空间通信任务，还使得不同层次的技术能够独立发展，同时保持整个系统的整体兼容性各层次的相关说明如下：

（1）物理层：物理层是最底层，负责通过物理媒体传输原始的比特流。这包括定义信号的电气、机械、时间特性以及物理连接方式，确保数据能够在空间环境中可靠传输。物理层的标准化是关键，因为它直接影响到信号的质量和整个通信系统的可靠性。

（2）数据链路层：数据链路层负责在物理连接上实现可靠的帧传输。这一层包括帧的同步、差错控制、帧排序和重传机制等。数据链路层是空间数据传输中非常关键的一层，因为它确保了数据包在复杂的空间环境中的完整性和正确性。

（3）网络层：网络层处理数据包在网络中的传输，包括路由选择和数据分段等功能。在空间通信中，网络层特别关注如何有效地通过多个节点（可能包括地面站、卫星和其他空间资产）传输数据，以及如何处理由于空间环境引起的信号延迟和丢失问题。

（4）传输层：传输层确保数据的端到端传输可靠性。这一层主要负责全面的差错恢复、数据流控制和确保完整的数据传输。在空间通信中，传输层的设计必须能够适应极端的传输延迟和高误码率的环境。

（5）应用层：应用层是最顶层，直接为用户提供服务。它定义了数据的表示、编码、处理、存储和显示方式。在空间数据系统中，应用层协议包括数据管理、文件传输、遥测处理和任务控制等多种服务。

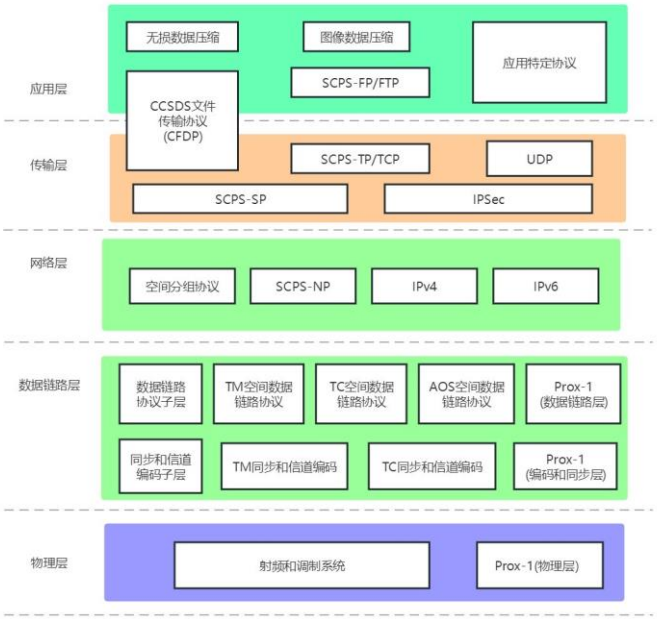


图 2-1　CCSDS 协议分层示意图

这种分层架构允许 CCSDS 在不同的层面上适应特定的技术需求和挑战，从而优化空间数据的传输和处理。通过这种方法，CCSDS 能够为不同的空间通信和数据处理任务提供灵活、高效和可靠的解决方案。[20]

2.1.2　CCSDS 链路层协议结构

在 CCSDS 的协议架构中，空间链路层包括了 CCSDS-AOS、CCSDS-TM 以及 CCSDS-TC 这三种协议。这些协议各自应对不同的数据传输需求和功能，确保空间任务中的数据可以安全、可靠地传输。CCSDS-AOS 协议主要设计用于更高级的轨道系统，如科学卫星和探测器。该协议提供了高度灵活的数据传输服务，支持多种数据类型的传输，包括实时、非实时以及包含多种数据源的混合数据。AOS 协议支持高数据率传输，并可适应各种轨道和通信环境的需求，

非常适合用于深空和地球轨道任务；CCSDS-TM 协议专为遥测数据设计，主要用于从空间飞行器向地面站传输监控数据。这些数据通常包括飞行器的状态信息、科学数据以及其他关键的性能指标。TM 协议注重数据的实时传输和高可靠性，确保飞行任务控制中心能够实时监控和分析空间飞行器的状况；[21] CCSDS-TC 协议用于从地面向空间飞行器发送指令，是一种反向的数据通信方式。TC 协议关注指令的准确传输和执行，包括数据的安全性和抗干扰性。[22] 通过这种协议，地面控制中心可以对飞行器进行操作控制，如调整轨道、启动科学仪器或执行飞行计划的修改。CCSDS 空间数据链路层协议的结构如图 2-2 所示：

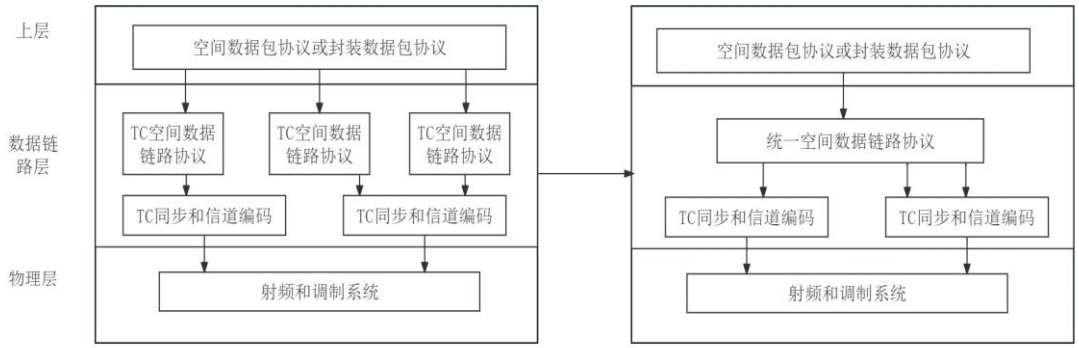


图 2-2　空间数据链路层协议结构图

这三种协议共同构成了 CCSDS 在空间链路层的核心框架，它们通过确保数据在极端空间环境下的准确性和可靠性，支持各种空间探测和运营任务的成功。CCSDS 的这一层次化和专业化的协议设计允许不同类型的数据在空间环境中有效地管理和传输，极大地提升了空间数据通信的效率和安全性。通过模式串匹配，我们可以轻松区分 TC 协议与 AOS 和 TM 协议。然而，由于 AOS 和 TM 协议拥有相同的同步头，仅凭这一特征难以彻底区分两者。[8] 考虑到这两种协议在字段上的差异，我们可以利用集成学习的技术来进一步区分它们。

2.1.3　TM 协议与 TC 协议

在 CCSDS 标准的初期阶段，遥测（TM）和遥控（TC）协议是最早被定义和实施的部分，这两种协议分别针对的是数据从空间飞行器到地面的传输（TM），以及从地面到空间飞行器的指令传输（TC）。这两种协议的开发和标准化是为了解决国际间空间任务在数据交换和操作指令传送方面的互操作性和可靠性问题。遥测协议的主要目的是提供一种标准化的方法来传输从空间飞行

器采集的数据回地面。这包括科学数据、工程数据以及状态监控数据。遥测协议特别关注数据的实时性和可靠性，确保地面操作团队能及时获得飞行器的状态信息，进行必要的监控和控制。为了提高数据传输的可靠性，遥测协议包括了错误检测和可能的错误纠正机制。遥控协议则关注如何安全、可靠地将控制指令从地面传输到空间飞行器。在空间任务中，发送给飞行器的每一个指令都至关重要，且必须确保精确无误地执行。因此，TC 协议包括了多重验证机制，以保证指令的正确性和完整性。此外，遥控协议还设计了防止指令重复执行的机制，以及确保指令在规定的时间窗口内执行的同步机制。

CCSDS 在标准的初期阶段确定这两种协议，为国际空间任务的设计和操作提供了极大的便利和标准化基础。通过这些协议的实施，不同国家的空间机构可以更容易地进行协作，共享数据，并统一指令控制方法。这不仅提高了任务的安全性和效率，也减少了设计和开发成本，因为机构们可以采用共通的技术解决方案和接口。随着时间的推移，这些协议得到了进一步的发展和扩展，以适应更复杂的任务需求和技术进步，如引入高级编码和调制技术，提高数据传输速率和距离，以及增加对加密和数据安全的支持。这些初始阶段的协议为后续更多高级功能的加入奠定了基础，证明了 CCSDS 在空间通信标准化方面的前瞻性和持续影响力。[19]

图 2-3　TM 协议传输帧结构

TM 协议的帧结构在图 2-3 中得到了详细展示。该帧结构主要由四个部分组成：主帧头、副帧头（可选）、数据域以及帧尾。主帧头是固定存在的，其长度为 6 字节；数据域则是每一帧中必不可少的元素，但其长度可根据具体数据需求进行调整。而副帧头和帧尾作为可选项，可以根据需要选择是否添加。副帧头的最大长度为 64 字节，而帧尾则固定为 6 字节，其中包括了 4 字节的操作控制信息和 2 字节的帧差错控制信息。[21] 这样的设计确保了 TM 协议在数据传输过程中的灵活性和可靠性。

<table>
<tr><td colspan="2">主信道ID（MCID）</td><td rowspan="2">虚拟信道标识符（VCID）</td><td rowspan="2">操作控制域标志（OCFF）</td><td rowspan="2">主信道帧计数（MCFC）</td><td rowspan="2">虚拟信道帧计数（VCFC）</td><td colspan="5">传输帧数据域状态</td></tr>
<tr><td>帧版本号（TFVN）</td><td>航天器标识符（SCID）</td><td>传输帧副导头标志</td><td>同步标志</td><td>包顺序标志</td><td>段长度标识符</td><td>首导头指针</td></tr>
</table>

图 2-4　TM 协议传输帧主帧头结构

TM 协议传输帧的主帧头结构如图 2-4 所示。传输帧主帧头包含几个必需字段，包括六个主要字段：主信道 ID（MCID）、虚拟信道标识符（VCID）、操作控制域标志（OCFF）、主信道帧计数（M_Count）、虚拟信道帧计数（VC_Count）和传输帧数据域状态。这些字段的具体构成如下：MCID 包含帧版本号（TFVN）和航天器标识符（SCID），分别占 2 位和 10 位；VCID、OCFF、M_Count 和 VC_Count 的长度分别为 3 位、1 位、1 字节和 1 字节；传输帧数据域状态则包括传输帧副导头标志、同步标志、包顺序标志、段长度标识符和首导头指针，这些分别占 1 位、1 位、1 位、2 位和 11 位。

<table>
<tr><td>传输帧主帧头</td><td>传输帧副帧头（可选）</td><td>帧差错控制域（可选）</td></tr>
</table>

图 2-5　TC 协议传输帧结构

TC 协议的传输帧结构如图 2-5 所示。遥控（TC）协议的传输帧结构被设计得非常精确，以确保从地面站到空间飞行器的命令传输既可靠又安全。TC 帧的结构反映了对数据完整性、错误检测、安全需求和实时传输性的高要求。[22] TC 协议的传输帧结构包括传输帧主帧头、传输帧数据域和帧差错控制域三个部分，通常又可以如 1. 帧头（Frame Header）

（1）帧头：包含了控制信息，如帧的序列号和虚拟信道标识符，这有助于飞行器和地面站跟踪每个帧的状态，并确保正确的帧排序和处理。帧头通常还包括协议版本号，以确保接收设备正确解析数据。

（2）命令数据区：这是帧的主体部分，包含了要传输的实际控制命令。命令数据区可以包含一个或多个命令包，每个命令包含特定的操作信息和参数。这部分数据通常采用数据分组的形式进行封装，每个数据包包含足够的信息以独立执行一个或多个操作。

（3）错误控制字段：为了增强数据的可靠性，TC 传输帧包含一个错误控制字段，通常是循环冗余检验（CRC）或其他类型的校验码。这个字段使得接收端能够检测到传输过程中可能产生的错误，并采取相应的纠错措施。

（4）帧尾部：某些协议版本中，帧的结尾可能包括额外的同步或安全相关字段。帧尾部的存在取决于特定的实施和安全要求，例如增加时间戳或加密验证数据以确保传输的安全性。

TC 协议特别关注命令帧的安全性和同步。为此，可能会实施加密措施来保护命令数据区，防止未授权的访问或篡改。此外，同步技术确保命令在预定的时间内有效执行，防止由于延迟或重复执行造成的问题。这种精心设计的传输帧结构使得 TC 协议能够在严峻的空间通信环境中提供高度可靠和安全的遥控操作能力，是空间任务中不可或缺的一个组成部分。

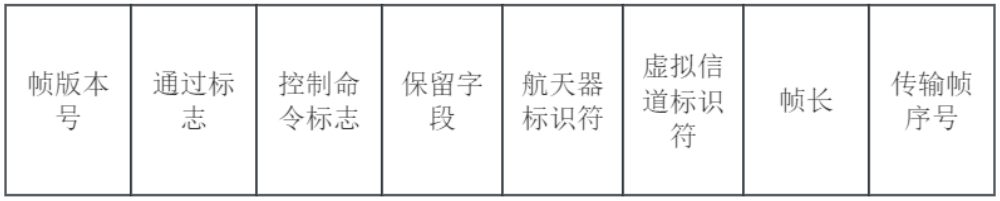


图 2-6　TC 协议传输帧结构

TC 协议传输帧主帧头结构如图 2-6 所示。传输帧的主帧头是整个帧结构中的核心部分，负责包含必要的控制信息来确保命令数据的正确路由和处理。主帧头的设计旨在提供关键的帧标识和同步信息，以支持高效且可靠的数据通信。主帧头具有如下功能：

（1）同步和对齐：主帧头提供必要的信息来支持帧的同步和对齐，这是在高误差环境下确保数据完整性的关键。

（2）误差检测：主帧头通常包含或与误差检测代码（如 CRC）配合使用，以检测和纠正可能在信号传输过程中发生的错误。

（3）安全和加密：在安全敏感的应用中，主帧头标识安全头的存在，指示数据如何被加密以及如何验证数据完整性和真实性。

这个主帧头结构支持了 TC 协议在确保空间通信中数据准确性和安全性方面的需求，为地面控制中心与空间飞行器之间的可靠通信提供了坚实的基础。

2.1.4　AOS 协议

在 20 世纪 90 年代，CCSDS 针对复杂航天器的数据通信挑战，基于 COS

进一步推出了 AOS。AOS 协议已进行了多方面的增强和改进，以适应不断增长的业务需求。这些改进旨在支持多种业务类型，并提高数据处理速度和传输能力，从而满足大数据量和高传输速率的要求。[23] CCSDS 的 AOS 协议中定义了一种标准数据单元，称为 AOS 传输帧（AOS Transfer Frame）。这种数据单元被设计用于在地面与先进空间系统之间传输指令和遥测信息，优化了数据的传输效率并提高了通信的可靠性。AOS 传输帧能够整合多种类型的数据流，如指令、遥测、以及科学数据，这种整合为复杂的空间任务提供了灵活和高效的数据处理能力。以下是 AOS 传输帧的一些关键特性：

（1）多用途性：AOS 传输帧可以包含各种类型的数据，如遥测信息、指令序列、科学数据等，这使得它可以被用于不同的通信需求和任务类型。

（2）可配置性：AOS 传输帧允许不同的配置选项，以适应不同的任务需求。例如，帧长、插入区内容和操作控制字段都可以根据特定的任务要求进行调整。

（3）帧同步：帧头包含同步和顺序控制信息，确保数据可以在接收时正确地被识别和处理。

（4）错误检测与纠正：通过帧错误控制字段（如 CRC），提高数据传输过程中的可靠性和完整性。

AOS 协议遵循固定长度的传输帧作为其数据交换的基本单元，其结构详见图 2-7。这个传输帧由四个主要部分构成：主帧头、插入字段（可选且长度可变）、数据域以及帧尾。特别地，帧尾作为一个可选项，总共占据 6 个字节，其中包含了 4 字节的操作控制区域和 2 字节的帧错误检测区域。

图 2-7　AOS 协议传输帧结构

如图 2-8 所示，AOS 传输帧的核心头部结构包含五个关键部分，分别是：主信道识别码（MCID），此结构包括帧版本编号（TFVN）和航天器识别码（SCID），它们占用 2 位和 8 位；虚拟通道标识符（VCID）则占 6 位；虚拟通道帧计数器（VC_Count）使用了 3 字节；信号域占 1 字节，包括重播指示（1

位）、VC 帧计数使用标记（1 位）、RSVD 保留位（2 位）以及 VC 帧计数循环标记（4 位）。还有一个可选的帧头错误控制字段，该字段占 2 字节。

主信道ID（MCID）		虚拟信道标识符（VCID）	虚拟信道帧计数	信号域				帧头差错控制（可选）
帧版本号（TFVN）	航天器标识符（SCID）			重播标志	VC帧计数使用标记	包顺序标志	段长度标识符	

图 2-8　AOS 协议传输帧主帧头结构

2.2　协议识别技术研究

当前，协议识别技术的研究主要聚焦于地基网络互联网领域的高层次，而对天基网络低层协议，尤其是 CCSDS 协议的识别研究相对匮乏。CCSDS 协议的识别复杂度远超地基协议，因此，深入研究 CCSDS 协议的空间链路层识别至关重要。

协议识别技术可以大致分为两大类：非机器学习方法和机器学习方法。这两种方法各有其特点和适用场景，通常针对不同的网络环境和识别需求被采用。以下是这两类方法的概述：

（1）非机器学习方法

基于端口号识别：这是最传统和简单的方法，依赖于预定义的端口号来识别协议。由于很多应用程序遵循标准的端口号分配，这种方法可以快速识别出常见的应用协议，如 HTTP（端口 80）、HTTPS（端口 443）等。局限性：许多现代应用使用动态端口或在标准端口上运行非标准协议，使得端口号识别方法的准确性下降。

基于协议特征和模式匹配：这种方法通过分析数据包中的特定字段和模式来识别协议。例如，可以通过检查数据包头部的特定位模式、关键字或者协议结构特征进行识别。局限性：加密和混淆技术的普及使得这种方法的有效性受限，特别是当无法直接访问有效载荷内容时。

（2）机器学习方法

监督学习：使用已标记的数据集训练模型来识别网络协议。常用的算法包括支持向量机（SVM）、决策树、随机森林等。[24] 这些方法通过学习数据集中的模式来预测未知数据的协议类型。优点是能够处理更复杂的模式和特征组合，提高识别的准确性。同时也存在局限性，需要大量的标记数据，并且对新协议或变化的网络环境的适应性较弱。

无监督学习：在没有标记数据的情况下，通过分析流量数据的统计特性来识别未知协议。常用方法包括聚类分析等。优点是不依赖于预先标记的数据，适用于探索性分析和未知协议的识别。局限性为可能难以确定具体的协议类型，只能识别出不同的流量模式或类别。

深度学习：利用深度神经网络，如卷积神经网络（CNN）和循环神经网络（RNN），进行协议识别。[25] 这些模型能够自动从数据中学习更深层次的特征。优点突出，对于加密和封装协议的识别具有更强的能力，能处理大规模数据，但需要较大的计算资源，且模型的解释性较差。

选择哪种协议识别技术取决于具体的应用场景、可用资源、所需的准确性和识别速度。随着技术的发展，机器学习方法尤其是深度学习在协议识别领域显示出了巨大的潜力和优势，尽管其复杂性和资源需求也相对较高。非机器学习方法则因其简单和速度快，仍被广泛用于一些基础和较为传统的应用。[26]

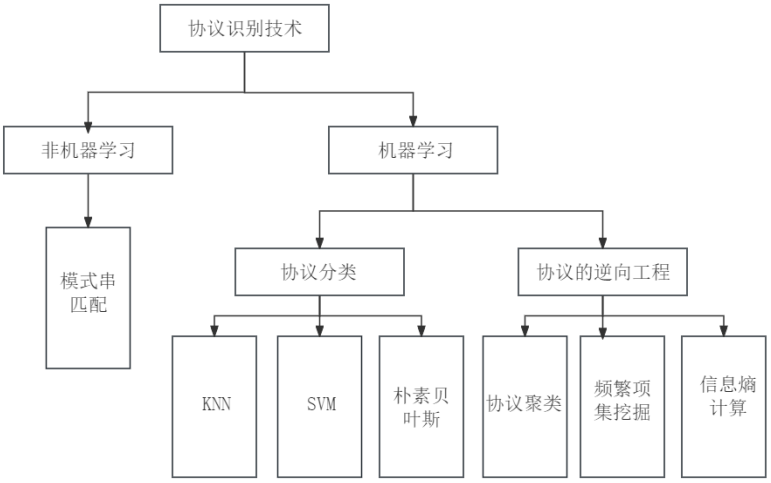


图 2-9　主要协议识别技术分类

2.3　基于机器学习的协议识别技术研究

机器学习驱动的协议识别技术，简而言之，是通过分析历史数据流数据集来提取统计模式，并利用这些模式来识别新数据集中的协议类型。这种方法也常被称为数据流导向的识别技术。它巧妙地绕过了动态端口和网络地址转换的干扰，无需深入解析协议内容，因此展现出高度的灵活性和实用性。机器学习识别流程大致如图 2-10 所示。

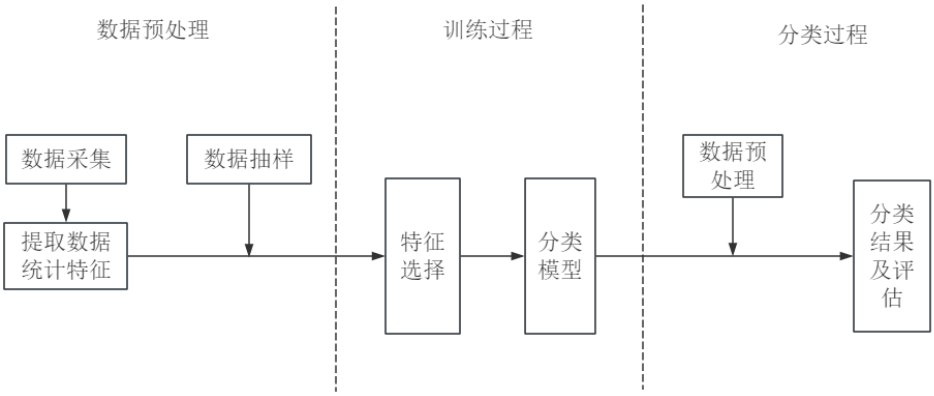


图 2-10　基于机器学习的识别方法流程图

机器学习识别技术涵盖数据预处理、模型训练与分类识别三个关键阶段。在预处理阶段，数据抽样至关重要，它通过筛选和精简大量原始数据，剔除冗余部分，以形成理想的训练数据集，从而简化后续的训练和分类过程。进入训练阶段后，特征选择成为核心步骤，它旨在去除无关或冗余特征，并确定对结果有显著影响的特征子集。利用这些特征子集训练模型，不仅提高了效率，还增强了识别的准确性。最终，经过多次迭代训练的有效样本集将生成高精度的分类识别模型。[27]

2.4　基于集成学习的协议识别技术研究

结合前文的分析与参考文献可得，各种识别技术的特点如表 2-1 所示：

表 2-1　识别技术特点分析表

名称	时长	内存	运行效率	特征选择	异常数据	数据集大小
模式串匹配	较慢（依赖串长度）	较高	较低	无	较敏感	敏感
简单机器学习	较慢	较高	较低	需要	敏感	敏感
集成学习	较快	相对较低	较高	不需要	不敏感	不敏感

在探讨模式串匹配技术时，其显著的不足体现在识别效率上，尤其是随着

目标串和模式串长度的增加，处理时间将急剧上升。目前，这项技术主要用于定长帧的协议识别，但在变长帧日益普遍的今天，其局限性愈发明显。

对于基于机器学习的协议识别技术，尽管在训练阶段需要较长时间，但在实际识别过程中则展现出了较高的效率。不过，SVM 模型在处理大规模数据时，由于需要执行二次规划和大规模矩阵计算，会显著消耗计算资源。而 KNN 模型在计算测试样本与所有训练样本的相似度并排序时，同样面临计算量大和内存占用高的问题。两者还存在过拟合风险，需额外进行特征选择，这进一步降低了识别效率。

然而，集成学习模型却以其高识别准确率脱颖而出。Bagging 和 Boosting 作为两种主流的集成学习方法，不仅能够有效避免过拟合，提升模型的泛化能力，还无需额外的特征选择步骤，从而加快了识别速度。此外，集成学习模型对异常数据有较好的鲁棒性，即使传输信道中存在误码，也能保持较高的识别准确性。它还能适应不同规模的数据集，无论是大数据集还是小数据集，都能通过相应策略进行有效处理。[5]

基于上述分析，本项目计划采用集成学习方法来识别 CCSDS 链路层协议。这一方法主要分为并行集成（如随机森林）和序列集成（如 Adaboost、GBDT）两类，旨在通过组合多个学习器来提高整体的识别性能和效率。

2.4.1 Bagging 思想

利用 Bagging 原理的协议识别过程在图 2-11 中进行了清晰的呈现。其核心步骤包括从原始训练集中随机抽取数据，每次抽取后都放回原集合，确保每个子集的大小相同但包含的训练样本各异。接着，利用这些子集分别训练出独立的基识别器。当需要预测新个体时，各个基识别器会给出各自的预测结果，最终通过投票的方式来确定一个综合且可靠的结论。

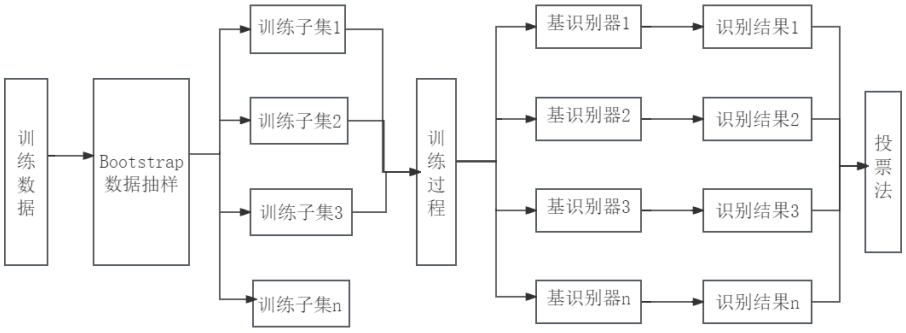


图 2-11　基于 Bagging 思想的协议识别方法流程图

Bagging 思想的算法表示如下所示

输入：

训练集 D={(x1,y1),(x2,y2),……,(xm,ym)}；

基学习算法 C；

基分类器个数 T；

集成学习 E；

过程：

1：For t = 1 to T；

2：ht = C(D,Dt)，依据 Bootstrap 分布，随机建立子集训练基分类器；

3：E = E∪Ct，将训练完的基分类器集成；

4：end For

输出：集成学习 E 及预测结果

2.4.2 Boosting 思想

如图 2-12 所示，基于 Boosting 策略的协议识别方法首先将训练数据集拆分成多个子集。接着，这些子集被用来训练基识别器，其中每一个后续的基识别器都是针对前一个基识别器产生的误差进行优化的。经过多次迭代后，得到一组基识别器，最后通过赋予不同的权重并将它们结合，形成一个集成识别器，以实现更精确的协议识别。

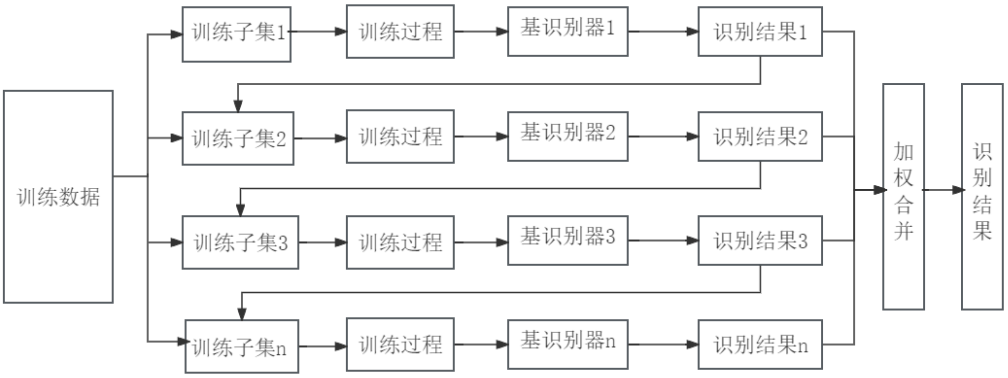


图 2-12　基于 Boosting 思想的协议识别方法流程图

Boosting 思想的算法表示如下所示

输入：

训练集 D={(x1,y1),(x2,y2),……,(xm,ym)}；

基学习算法 C；

基分类器个数 T；

集成学习 E；

过程：

1：For t = 1 to T；

2：ht = C(Dt)，训练基分类器；

3：θ(t) = P(ht(x)≠f(x))，计算 h(t)误差；

4：Dt+1 = Adjust(Dt,θt)，调整样本分布；

5：E = E∪Ct，将训练完的基分类器集成；

6：end For

输出：集成学习 E 及预测结果

2.5 本章小结

CCSDS 成立的主要目的就是为了使空间数据传输任务能够标准化，以此来加强世界各国之间在空间网络的合作和支持。CCSDS 协议标准体系采用了五层的架构体系，这种层级结构确保了数据传输的条理性和高效性。在 CCSDS 协议体系中，空间链路层协议主要涵盖 CCSDS-AOS、CCSDS-TM 以及 CCSDS-TC 协议。每种协议的特性主要通过其特有的字段来体现。

当前的协议识别方法主要可分为两种类型，机器学习驱动的协议识别技术，是通过分析历史数据流数据集来提取统计模式，并利用这些模式来识别新数据集中的协议类型。集成学习方法在识别任务中展现出较高的准确性，通常可以划分为两种主要类型：一种是基于 Bagging 的并行集成策略，这种方法中的基本学习器是同时生成的；另一种是基于 Boosting 的序列集成策略，其中基本学习器是依次生成的。

第 3 章　基于集成学习的协议识别的前置工作

3.1　引言

在前面的章节中，本项目研究了 CCSDS 链路层协议的特点，研究并比较几种协议识别技术的优劣性，最终决定本项目采用基于集成学习的链路层协议识别方法，具体来说，主要是采用基于 Bagging 思想的并行集成方法，包括随机森林算法等；基于 Boosting 思想的序列集成方法典型的算法是随机森林算法，包括 Adaboost 算法、GBDT 算法等。在本章中，本项目将先对链路层协议识别系统进行设计，然后完成一些协议识别的前置工作。

3.2　链路层协议识别系统的设计

在实现协议识别系统之前，本项目预设了一个链路层协议识别系统的流程，如图 3-1 所示。该系统由两大核心部分组成：数据准备与模型训练。在数据准备阶段，我们需要执行一系列步骤来确保数据的完整性和可用性，这涵盖了数据整理、数据扩展、特征筛选和数据标准化等关键环节。随后，进入模型训练阶段，我们将基于特定的协议识别方法，构建并训练出能够准确识别协议的模型，并通过参数优化来进一步提升模型的识别效率和性能。

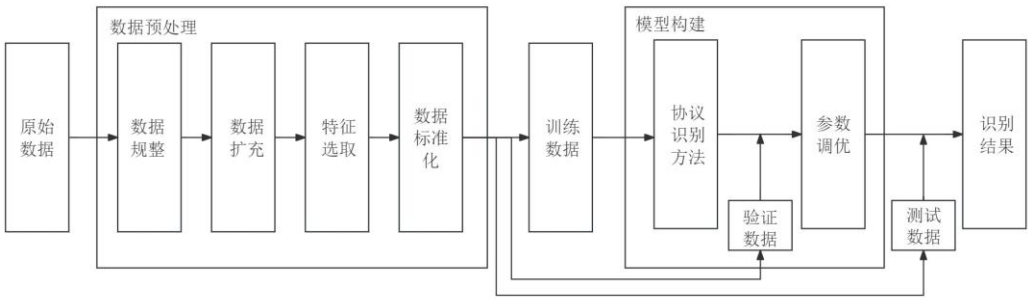


图 3-1　预设计的链路层协议识别系统流程图

3.2.1　数据预处理的设计

（1）数据规整：在实验准备阶段，我们可能会从文本或二进制文件中获取原始协议帧数据。为确保数据的结构化并方便后续分析，这些原始数据需要经过一系列的规整处理。数据预处理是网络通信中确保数据完整性和正确性的关

键步骤，尤其是在进行协议分析和数据传输任务时。主要包括识别同步头、确定帧长度、校验帧长度以及截取传输帧这四个关键步骤。同步头是数据帧中用于标识帧开始的一段特定的位模式。通过识别这个同步头，系统能够定位数据流中帧的起始位置。通常使用模式匹配技术来检测同步头的存在。在数据流中搜索与预定义同步头模式相匹配的序列，一旦检测到，即可确定帧的起始点；确定帧长度是知道每一帧的长度对于正确解析和处理每个数据帧至关重要。帧长度可能固定也可能变化，具体取决于协议规范。帧长度信息通常在帧头部中给出，需要解析帧头的相关字段以获取长度信息。对于固定长度的帧，这个步骤可能简单或甚至不需要；对于可变长度帧，则需要动态解析；校验帧长度是为了确保解析过程中没有发生错误，如帧的截断或解析错误。通过比较实际接收到的数据长度与帧头中指定的长度，验证两者是否一致。不一致可能指示数据传输错误或帧损坏；截取传输帧为正确地切分完整的数据帧，为后续的处理步骤（如解码、分析）提供准备。一旦帧的起始点和长度被确认并校验无误，下一步就是从数据流中截取完整的帧。这涉及到从同步头开始，根据确定的帧长度截取数据。

这个预处理流程对于任何需要精确数据传输和解析的应用都是非常重要的，比如卫星通信、网络监控、以及数据链路层的通信协议。有效的数据预处理不仅确保了数据的准确性和可用性，还能显著提高后续处理步骤的效率和可靠性。通过这种方式，可以最大限度地减少数据处理中的错误和不一致，确保通信和数据分析的高质量。

（2）数据扩充：为确保正负样本（即 AOS 协议的正例和 TM 协议的负例）的数据结构一致性，本项目计划基于各自数据字段的类别，对正例和负例数据进行扩展，使其具备统一的数据结构。

（3）特征选取：虽然“Data”字段包含了很多传输数据，它主要涵盖链路层以上的信息，但因协议细节主要集中在传输帧头部，因此“Data”字段对协议分类的直接贡献较小，不适合作为训练特征使用。同时，我们将采用嵌入式特征选择技术，在模型训练过程中直接评估并选择那些贡献度较高的特征，从而节省时间，提高协议识别的效率。

（4）数据标准化：由于 AOS 协议和 TM 协议的特征字段虽然相同，但比特位数可能不同，为避免这种差异对训练产生影响，我们需要在正式训练前对数据集进行规范化处理。本项目所采用的规范化方法已在公式（3-1）中给出。

$$X' = \frac{X - Mean}{Standard\ Deviation} \tag{3-1}$$

其中，*X* 为原始特征值，*X'*则为对应的新特征值，*Mean* 表征均值，*Standard Deviation* 为该特征的标准差。

3.2.2 模型构建的设计

本项目预计构建集成学习的数据处理模型，在得到经过处理的大量数据或数据库后，用随机森林，GBDT，Adaboost 等比较经典的集成学习算法构建模型，将数据中的一部分拿出来让模型学习，然后用训练好的模型识别其它的数据，并最终得出结论。

3.3 协议帧数据分析

在 CCSDS 协议的同步与信道编码子层中，AOS 和 TM 协议使用相同的 4 字节同步头“1ACFFC1D”，而 TC 协议采用了独有的同步头“EB90”。这使得使用模式匹配技术可以容易地将 TC 协议与 AOS 和 TM 协议区分开。但是，由于 AOS 和 TM 协议共享同一个同步头，这使得单靠这一特征难以对它们进行准确的识别。不过，因对于这两种协议在设计方面，字段的差异可用作区分两者，。在表中所列的 21 个字段特征中，有 7 个字段是可选的。为了消除这些可选字段对传输帧长度判断和截取的不确定影响，本项目将不会选择这些字段作为模型训练的特征。

在 AOS 协议（正例）和 TM 协议（负例）之间，数据特征表现出明显的不同。为保持数据结构在正例和负例之间的一致性，本研究通过根据数据字段的种类进行了数据扩充，从而形成了结构统一的数据集。以下是数据扩充的具体步骤：

a.将 AOS 协议的 SYN 字段与 TM 协议的 SYN 字段统一为 SYN 字段特征。

b.合并 AOS 协议的 TFVN 字段和 TM 协议的 TFVN 字段，形成统一的 TFVN 字段特征。

c.对 AOS 协议的 SCID 字段和 TM 协议的 SCID 字段进行同样的合并操作，形成统一的 SCID 字段特征。

d.类似地，合并了 AOS 协议和 TM 协议中的 VCID 字段，创建了一个统一的 VCID 特征字段。

e.由于 AOS 协议没有 OCFF 字段，而 TM 协议中包含该字段，因此将 OCFF

字段独立出来，作为一个单独的特征字段，并在 AOS 协议的数据中补 0 以填补空缺。

f.对于 M_Count 字段，也采取相同处理，即在 AOS 协议的数据中补 0 以填补空缺。

g.结合了 AOS 协议和 TM 协议中的 VC_Count 字段，形成了统一的 VC_Count 特征字段。

h. 同时，将 AOS 协议中的 Signal 字段与 TM 协议中的 TFDFS 字段进行了合并。

i.将 AOS 协议的 Data 字段与 TM 协议的 Data 字段合并，形成统一的 Data 字段特征。

经过上述数据扩充后，AOS 协议和 TM 协议的帧数据结构如图 3-3 所示，包含上述 9 个部分，并在末尾添加一个标签 y，其中 AOS 协议的 y 值为 1，TM 协议的 y 值为 0。

SYN	TFVN	SCID	VCID	OCFF	M_Count	VC_count	Signal	Data	y

图 3-3　AOS 和 TM 统一数据结构

本项目中的 AOS 协议的实验数据帧结构如图 3-4 所示：

AOS协议仿真数据结构									
帧同步码	帧版本号	航天器标识符	虚拟信道标识符	虚拟信道帧计数	重播标志	VC帧计数使用标记	RSVD预留位	VC帧计数循环	传输帧数据域
不定	2bit	8bit	6bit	3字节	1bit	1bit	2bit	4bit	可变

图 3-4　AOS 协议仿真数据结构

根据 CCSDS-AOS 协议蓝皮书的指南，字段值设置如下：

a. 帧同步码（SYN, 可变长度）：在特定任务中，该值保持不变。CCSDS 建议使用 4 字节的“0x1ACFFC1D”。

b. 帧版本号（TFVN, 2 位）：虽然 CCSDS 推荐设定为“01”，但由于是 2 位字段，实际上有四种可能的配置。

c. 航天器标识符（SCID, 8 位）：由 CCSDS 指派，与传输帧数据紧密相关，

整个任务期间保持不变。这 8 位提供了 256 个不同的标识符。

d. 虚拟信道标识符（VCID, 6 位）：能够标识最多 64 个虚拟信道。

e. 虚拟信道帧计数（VC_Count, 3 字节）：计数范围从 0 到“224-1”，每传输一个数据单元计数增加 1，采用循环计数方法。

f. 信号域（Signal, 8 位）：包含以下子字段：重播标志（RF, 1 位）用于标识帧是实时传输还是重放，其中“0”表示实时，“1”表示重放；VC 帧计数使用标志（VCFCUF, 1 位）指示是否启用虚拟信道帧计数循环，“0”表示未启用，“1”表示启用；保留位（RSVD, 2 位）固定为“00”；虚拟信道帧计数循环标志（VCFCC, 4 位）在虚拟信道帧计数归零时增加，将计数从 24 位扩展到 28 位，如果未使用循环计数，则这 4 位设为 0。

g.传输帧数据内容（Data, 长度可变）：数据域填充为正弦波形数据。

本项目中的 TM 协议的实验数据帧结构如图 3-5 所示：

TM协议仿真数据结构												
帧同步码	帧版本号	航天器标识符	虚拟信道标识符	OCF标记	主信道帧计数	VC帧计数	传输帧副导头标志	同步标识	包顺序标志	段长标识符	首导头指针	传输帧数据域
不定	2bit	10bit	3bit	1bit	1字节	1字节	1bit	1bit	1bit	2bit	11bit	可变

图 3-5　TM 协议仿真数据结构

根据 CCSDS-TM 协议蓝皮书的建议，字段配置如下：

a. 帧同步码（SYN, 可变长度）：整个任务期间，此字段保持固定。CCSDS 推荐使用 4 字节的“0x1ACFFC1D”作为同步码。

b. 帧版本号（TFVN, 2 位）：虽然推荐固定为“00”，但这个两位字段实际上可以有四种不同的设置。

c. 航天器标识符（SCID, 10 位）：由 CCSDS 分配的 ID，与帧传输数据相关联，在整个任务中保持不变。这 10 位允许最多 1024 个不同的航天器识别。

d. 虚拟信道标识符（VCID, 3 位）：能够标识最多 8 个不同的虚拟信道。

e. 操作控制域标志（OCFF, 1 位）：标识操作控制域是否存在。当设置为“1”时表示存在；设置为“0”时表示不存在。

f. 主信道帧计数器（M_Count, 1 字节）：从 0 开始，每传输一个数据单元增加 1，实现循环计数，范围是“0-255”。

g. 虚拟信道帧计数器（VC_Count, 1 字节）：同主信道帧计数器，从 0 开始计数，每传输一个数据单元增加 1，同样实现循环计数，范围也是“0-255”。

h. 传输帧数据状态字段（TFDFS, 2 字节）：包括五个部分：传输帧副导头标志（TFSHF, 1 位）表示是否有副导头；同步标志（SF, 1 位）用于标识插入到数据域的数据类型；保留标志（POF, 1 位）在 SF 为“0”时保留，SF 为“1”时未定义；段长标识符（SLI, 2 位）根据 SF 值有不同定义；首导头指针（FHP, 11 位）在 SF 为“0”时指示数据域中第一个数据包的起始位置，若 SF 全为“0”，则未定义。

i. 数据域（Data, 可变长度）：填充为正弦数据。

AOS (Advanced Orbiting Systems) 协议和 TM (Telemetry) 协议是由国际空间数据系统咨询委员会（CCSDS）开发的两种关键通信协议，主要用于空间任务中的数据传输。这两种协议各自具有独特的数据特征，支持不同类型的空间通信需求。以下是 AOS 协议和 TM 协议的主要数据特征对比。

对于 AOS 协议的数据特征，具有以下特点：

（1）灵活性和扩展性：AOS 协议设计用于支持高带宽和高数据速率的应用，如高分辨率图像传输和科学数据传回。它具有高度的灵活性和可配置性，可以适应各种不同的任务需求。

（2）数据单元结构：AOS 数据传输使用称为“传输帧”（Transfer Frames）的标准数据单元，这些帧可以动态调整以适配不同类型的有效载荷数据。

（3）虚拟信道：AOS 支持多个虚拟信道，这允许在单个物理链路上同时传输多种类型的数据流，如遥测数据、命令和科学信息。

（4）同步和异步传输：AOS 协议支持同步和异步数据传输，提供了灵活的时间管理和数据处理方式，以优化通信效率。

（5）控制：AOS 协议包括内置的错误检测和纠正机制，确保数据在恶劣的空间环境中的传输可靠性。

对于 TM 协议的数据特征，具有以下特点：

（1）主要用途：TM 协议主要设计用于传输遥测数据，它适用于各种空间任务中的状态监控和环境数据回传。

（2）帧结构：TM 数据也使用传输帧，但这些帧结构相对简单，专为遥测数据的高效传输而优化。

（3）数据速率：TM 协议通常适用于低至中等数据速率的传输，特别是在带宽较为受限的应用场景中。

（4）错误控制：类似于 AOS，TM 协议也提供了错误控制功能，但通常以简化的形式实现，以满足遥测数据传输的可靠性需求。

（5）实时性：TM 协议强调实时数据传输的重要性，以确保地面操作员能

够及时获得航天器状态和科学数据。

3.4 本章小结

在进行实验前，本项目对链路层协议识别系统的工作流程进行了初步设计。实验部分主要涉及数据的预处理和识别过程模型的构建两部分。本项目在构建模型前先对协议帧数据进行了分析，数据预处理参照了 CCSDS 蓝皮书对于 AOS 和 TM 帧字段的规定以及填充方法。AOS 协议提供了更高的灵活性和数据传输能力，适用于数据量大和多种数据类型的空间任务。相比之下，TM 协议则更加专注于实时性和遥测数据的高效传输，适用于数据速率较低且对带宽有限制的情况。两种协议都有强大的错误控制能力，以适应空间环境中的挑战。

第 4 章　基于集成学习的协议识别的系统实现

4.1　系统环境

处理器：Intel(R) Core(TM) i7-10710U CPU @ 1.10GHz　　1.60 GHz

操作系统版本：windows 11

编辑器：Pycharm Community Edition 2022

编程语言：Python 3.11

4.2　系统实现过程

4.2.1　生成数据集

在进行实验前，本项目对链路层协议识别系统的流程进行了预设计，主要包括数据预处理和识别数据模型的构建两部分

在数据预处理时，具体 AOS 和 TM 的实验数据参照 5.3 部分的详细介绍，本项目利用随机数生成函数和数组生成函数，将生成的大量数组沿着一定方向堆叠在一起。最后将其存入 Excel 逗号分隔值文件中，形成一个可读取，可修改，可存储的数据集，该数据集中包含大量的 AOS 和 TM 协议实验数据。生成的数据集截取一小部分如图 4-1 所示。

	SYN	TFVN	SCID	VCID	OCFF	M_Count	VC_Count	Signal	y
2	4.5E+08	0	0	0	0	0	0	64	1
3	4.5E+08	1	53	51	0	0	1	64	1
4	4.5E+08	1	160	7	0	0	2	64	1
5	4.5E+08	2	242	29	0	0	3	64	1
6	4.5E+08	3	10	38	0	0	4	64	1
7	4.5E+08	0	176	45	0	0	5	64	1
8	4.5E+08	0	226	8	0	0	6	64	1
9	4.5E+08	3	208	22	0	0	7	64	1
10	4.5E+08	2	237	20	0	0	8	64	1
11	4.5E+08	1	156	3	0	0	9	64	1
12	4.5E+08	1	226	57	0	0	10	64	1
13	4.5E+08	3	76	14	0	0	11	64	1
14	4.5E+08	3	233	30	0	0	12	64	1
15	4.5E+08	3	196	13	0	0	13	64	1
16	4.5E+08	1	102	22	0	0	14	64	1
17	4.5E+08	1	151	17	0	0	15	64	1
18	4.5E+08	1	91	40	0	0	16	64	1
19	4.5E+08	3	153	27	0	0	17	64	1
20	4.5E+08	0	85	54	0	0	18	64	1

图 4-1　实验生成的数据集示例

4.2.2 构建模型

进行完数据预处理后，开始构建识别数据集的模型。具体来说，本项目主要是采用基于 Bagging 思想的并行集成方法，包括随机森林算法；以及基于 Boosting 思想的序列集成方法，典型的算法包括 Adaboost 算法、GBDT 算法。利用这三种集成学习算法构建模型，模型内容包括读取数据集中的数据，然后取其中一部分数据进行学习，另一部分数据用于识别，最终根据一定的指标来验证模型识别协议的准确性。

本项目采用了 F1-score 作为主要的性能评估标准。在 AOS 协议与 TM 协议识别过程中，在评估中，本项目区分四种结果：正确识别的正例（True Positive, TP）、被错误地识别为正例的负例（False Positive, FP）、被错误地识别为负例的正例（False Negative, FN）以及正确识别的负例（True Negative, TN）。具体地，TP 表示预测结果和实际情况都是正例；FP 表示预测结果是正例但实际上是负例；FN 表示预测结果是负例但实际上是正例；TN 表示预测结果和实际情况都是负例。

F1-score 作为二分类模型效果评估的关键指标，其计算公式见式（4-1）。

$$F1 = 2 * \frac{precision * recall}{precision + recall} \tag{4-1}$$

其中，*precision*（精确度）的计算公式为 TP/(TP+FP)，它衡量了在所有被预测为正例的样本中，真正为正例的比例；而 *recall*（召回率）的计算公式为 TP/(TP+FN)，它则反映了模型能够找出所有真实正例的能力。F1-score 是一个结合了精确度和召回率的指标，其值域在 0 到 1 之间，数值越高，表示模型的性能越好，预测结果越准确。[4]

在用同一个数据集对三种算法的模型运行后，结果是相同的，即三种算法的正确率都能够达到 100%，如图 4-2 和图 4-3 所示，这可能是由于数据集太过于标准和简单所导致的，因此需要加入一定的干扰因素，使得数据集变得更加复杂或更加难以识别，这样才比较接近实际情况。

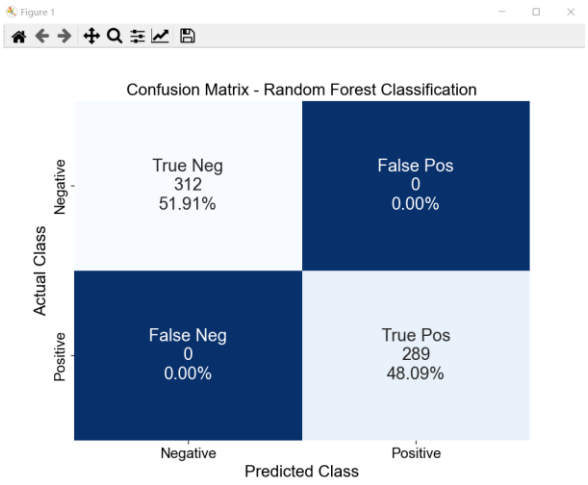


图 4-2　三种算法模型运行结果 1

```
RandomForestClassifier(random_state=42) , Accuracy score: 1.0
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       312
           1       1.00      1.00      1.00       289

    accuracy                           1.00       601
   macro avg       1.00      1.00      1.00       601
weighted avg       1.00      1.00      1.00       601
```

图 4-3　三种算法模型运行结果 2

数据从天基网络发送至地基网络的过程中，途中会有各种各样的干扰。如前文所说，数据集需要加入一定的干扰因素，使得数据集变得更加复杂或更加难以识别，这样才比较接近实际情况。因此，本项目在原有标准数据集的基础之上，在数据进行归一化之后加入高斯噪声，以此来模拟在数据生成或传输过程中可能遇到的噪声或其它干扰。以此来验证该识别系统的抗干扰能力并比较三种集成学习算法的优劣。

4.3　系统识别结果及优化

4.3.1　抗干扰性

图 4-4 和图 4-5 为随机森林算法在加入 1 单位高斯噪声后的运行结果，图 4-6 和图 4-7 为随机森林算法在加入 1.2 单位高斯噪声后的运行结果，图 4-8 和图 4-9 为随机森林算法在加入 1.4 单位高斯噪声后的运行结果。在下面的运行结果图中，两个浅蓝色的区域为识别正确（即预测结果与实际数据相同）的部分，

而两个深蓝色的区域为识别错误（即预测结果与实际数据不相同）的部分。图中的四部分分别对应了 F1-score 评价指标中的四种结果，即真阳性、假阳性、真阴性、假阴性。由下面的运行结果可知，随机森林算法模型加入 1 单位噪声后的识别准确率为 99.33%，加入 1.2 单位噪声后的识别准确率为 98.17%，加入 1.4 单位噪声后的识别准确率为 96.34%。

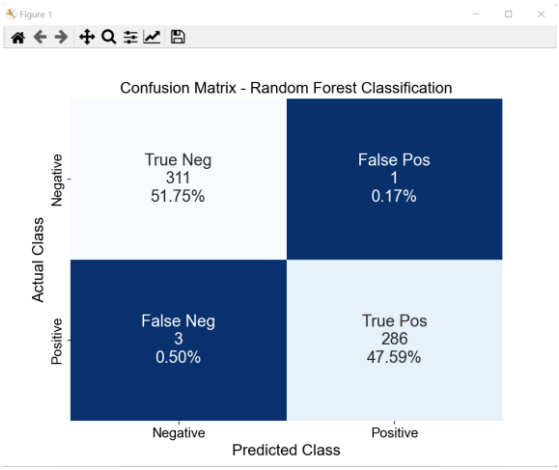


图 4-4　随机森林算法模型 1 单位噪声运行结果 1

```
RandomForestClassifier(random_state=42) , Accuracy score: 0.9933444259567388
Classification report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       312
           1       1.00      0.99      0.99       289

    accuracy                           0.99       601
   macro avg       0.99      0.99      0.99       601
weighted avg       0.99      0.99      0.99       601
```

图 4-5　随机森林算法模型 1 单位噪声运行结果 2

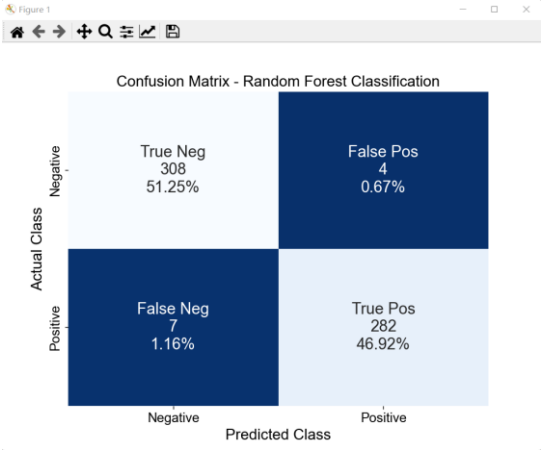


图 4-6　随机森林算法模型 1.2 单位噪声运行结果 1

```
RandomForestClassifier(random_state=42) , Accuracy score: 0.9816971713810316
Classification report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.98       312
           1       0.99      0.98      0.98       289

    accuracy                           0.98       601
   macro avg       0.98      0.98      0.98       601
weighted avg       0.98      0.98      0.98       601
```

图 4-7　随机森林算法模型 1.2 单位噪声运行结果 2

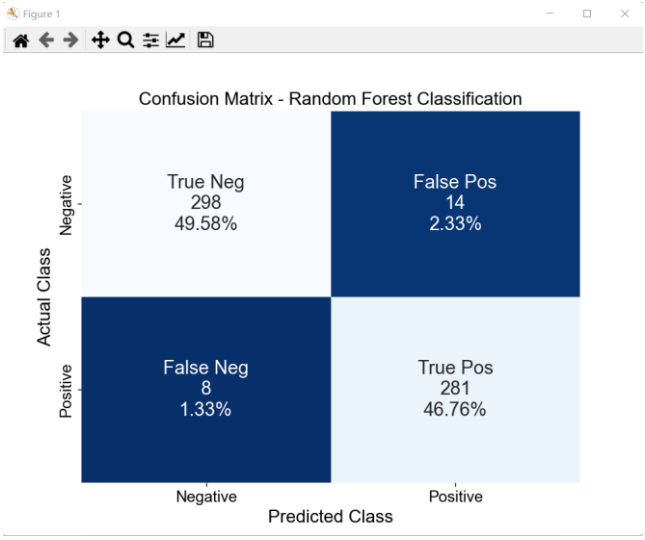


图 4-8　随机森林算法模型 1.4 单位噪声运行结果 1

```
RandomForestClassifier(random_state=42) , Accuracy score: 0.9633943427620633
Classification report:
              precision    recall  f1-score   support

           0       0.97      0.96      0.96       312
           1       0.95      0.97      0.96       289

    accuracy                           0.96       601
   macro avg       0.96      0.96      0.96       601
weighted avg       0.96      0.96      0.96       601
```

图 4-9　随机森林算法模型 1.4 单位噪声运行结果 2

图 4-10 和图 4-11 为 GDBT 算法在加入 1 单位高斯噪声后的运行结果，图 4-12 和图 4-13 为 GDBT 算法在加入 1.2 单位高斯噪声后的运行结果，图 4-14 和图 4-15 为 GDBT 算法在加入 1.4 单位高斯噪声后的运行结果。在下面的运行结果图中，两个浅蓝色的区域为识别正确（即预测结果与实际数据相同）的部分，而两个深蓝色的区域为识别错误（即预测结果与实际数据不相同）的部分。

图中的四部分分别对应了 F1-score 评价指标中的四种结果，即真阳性、假阳性、真阴性、假阴性。由下面的运行结果可知，GDBT 模型加入 1 单位噪声后的识别准确率为 98.50%，加入 1.2 单位噪声后的识别准确率为 96.84%，加入 1.4 单位噪声后的识别准确率为 96.34%。

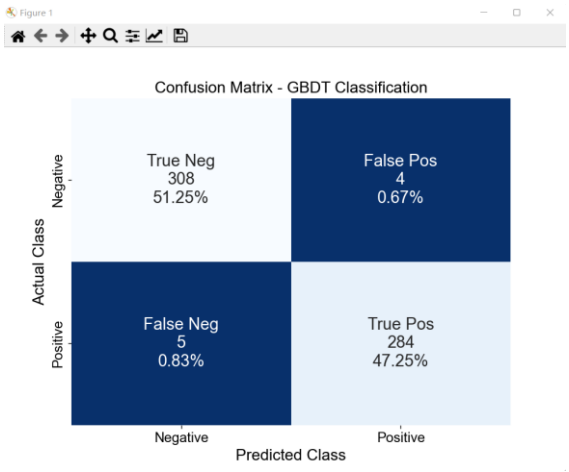


图 4-10　GDBT 算法模型 1 单位噪声运行结果 1

```
GradientBoostingClassifier(random_state=42) , Accuracy score: 0.9850249584026622
Classification report:
              precision    recall  f1-score   support

           0       0.98      0.99      0.99       312
           1       0.99      0.98      0.98       289

    accuracy                           0.99       601
   macro avg       0.99      0.98      0.99       601
weighted avg       0.99      0.99      0.99       601
```

图 4-11　GDBT 算法模型 1 单位噪声运行结果 2

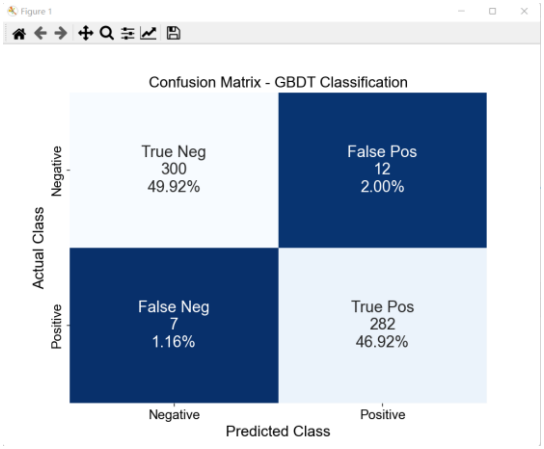


图 4-12　GDBT 算法模型 1.2 单位噪声运行结果 1

```
GradientBoostingClassifier(random_state=42) , Accuracy score: 0.9683860232945092
Classification report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       312
           1       0.96      0.98      0.97       289

    accuracy                           0.97       601
   macro avg       0.97      0.97      0.97       601
weighted avg       0.97      0.97      0.97       601
```

图 4-13　GDBT 算法模型 1.2 单位噪声运行结果 2

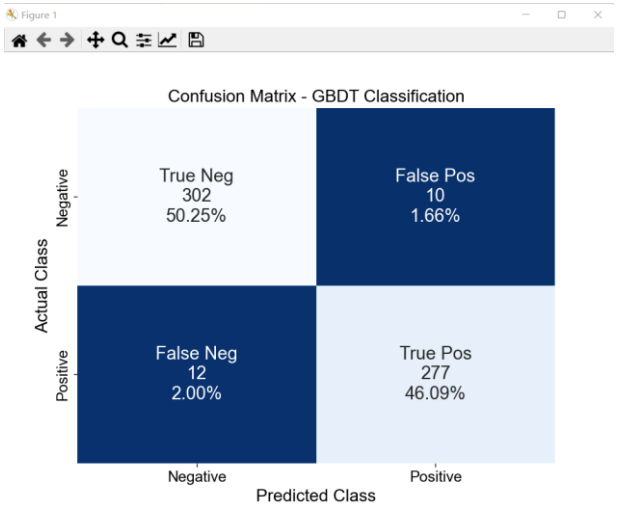


图 4-14　GDBT 算法模型 1.4 单位噪声运行结果 1

```
GradientBoostingClassifier(random_state=42) , Accuracy score: 0.9633943427620633
Classification report:
              precision    recall  f1-score   support

           0       0.96      0.97      0.96       312
           1       0.97      0.96      0.96       289

    accuracy                           0.96       601
   macro avg       0.96      0.96      0.96       601
weighted avg       0.96      0.96      0.96       601
```

图 4-15　GDBT 算法模型 1.4 单位噪声运行结果 2

图 4-16 和图 4-17 为 Adaboost 算法在加入 1 单位高斯噪声后的运行结果，图 4-18 和图 4-19 为 Adaboost 算法在加入 1.2 单位高斯噪声后的运行结果，图 4-20 和图 4-21 为 Adaboost 算法在加入 1.4 单位高斯噪声后的运行结果。在下面的运行结果图中，两个浅蓝色的区域为识别正确（即预测结果与实际数据相同）的部分，而两个深蓝色的区域为识别错误（即预测结果与实际数据不相同）的

部分。图中的四部分分别对应了 F1-score 评价指标中的四种结果，即真阳性、假阳性、真阴性、假阴性。由下面的运行结果可知，随机森林算法模型加入 1 单位噪声后的识别准确率为 98.67%，加入 1.2 单位噪声后的识别准确率为 97.17%，加入 1.4 单位噪声后的识别准确率为 96.01%。

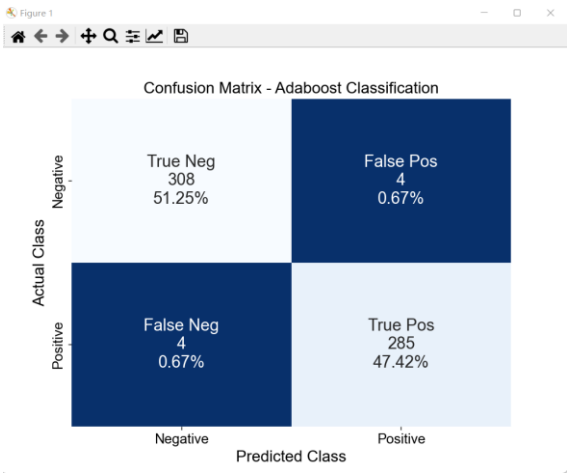


图 4-16　Adaboost 算法模型 1 单位噪声运行结果 1

```
AdaBoostClassifier(n_estimators=100, random_state=42) , Accuracy score: 0.9866888519134775
Classification report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99       312
           1       0.99      0.99      0.99       289

    accuracy                           0.99       601
   macro avg       0.99      0.99      0.99       601
weighted avg       0.99      0.99      0.99       601
```

图 4-17　Adaboost 算法模型 1 单位噪声运行结果 2

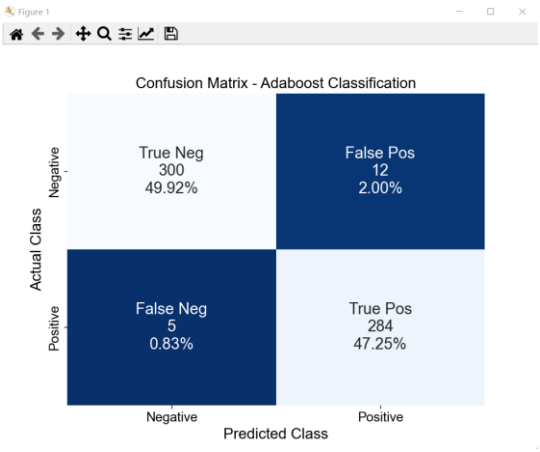


图 4-18　Adaboost 算法模型 1.2 单位噪声运行结果 1

```
AdaBoostClassifier(n_estimators=100, random_state=42) , Accuracy score: 0.9717138103161398
Classification report:
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       312
           1       0.96      0.98      0.97       289

    accuracy                           0.97       601
   macro avg       0.97      0.97      0.97       601
weighted avg       0.97      0.97      0.97       601
```

图 4-19　Adaboost 算法模型 1.2 单位噪声运行结果 2

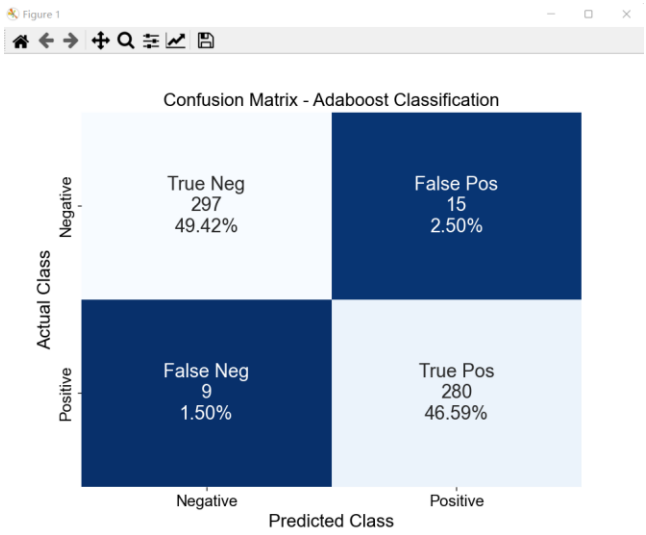


图 4-20　Adaboost 算法模型 1.4 单位噪声运行结果 1

```
AdaBoostClassifier(n_estimators=100, random_state=42) , Accuracy score: 0.9600665557404326
Classification report:
              precision    recall  f1-score   support

           0       0.97      0.95      0.96       312
           1       0.95      0.97      0.96       289

    accuracy                           0.96       601
   macro avg       0.96      0.96      0.96       601
weighted avg       0.96      0.96      0.96       601
```

图 4-21　Adaboost 算法模型 1.4 单位噪声运行结果 2

最终三种算法模型加入不同噪声后的识别准确率如表 4-1。

表 4-1　不同噪声下不同算法识别准确率

噪声单位	随机森林	GBDT	Adaboost
1	99.33%	98.50%	98.67%
1.2	98.17%	96.84%	97.17%
1.4	96.34%	96.34%	96.17%

根据这些运行图以及运行结果的比较可知，三种算法都能在较强的噪声环境下保持较高的识别准确率，虽然随着噪声的比例增大，识别的准确率有所下降，不过整体都能维持在 96%以上，且通过对三种算法的比较可知，随机森林算法的准确率比其它两种算法更高一些，因此最终本项目采取算法识别协议。

4.3.2　调整参数

假如随机森林模型是一片森林，那么它做出的决策是由里面的所有树木共同决定的，因此，子决策树的数量，也就是基评估器的数量是随机森林模型中一个十分重要的参数。一般情况下，如果子决策树过多或过少，都可能会出现决策不准，也就是识别准确率下降的情况。因此，本项目在模拟一定噪声的环境下对随机森林模型的基评估器数量参数从 1 到 100 进行调整，调整结果如表 4-2 所示：

表 4-2　不同基评估器数量下随机森林算法识别准确率

基评估器数量	识别准确率
100	95.17%
90	94.68%
80	96.84%
70	96.84%
60	96.01%
50	94.84%
40	95.67%
30	95.67%
20	94.34%
10	94.34%
1	86.36%

根据表 4-2 对于不同基评估器数量下随机森林模型识别准确率比较可知，在基评估器数量从 100 减少到 1 的过程中，识别准确率有一定的波动，但是大体上来看，基评估器数量从 100 减少到 70 的过程中识别准确率是上升的，基评估器数量从 70 减少到 1 的过程中识别准确率是下降的，尤其是基评估器数量为

1 时识别准确率很低。因此把基评估器数量这个参数设置在 70 是比较合适的。

4.4 本章小结

在进行实验的过程中，模型构建采用了随机森林算法、Adaboost 算法、GBDT 算法三种经典集成学习算法构建模型，并对数据集加入了一定的干扰因素，最后，本实验采用 F1-score 评价指标检验了识别协议模型的准确性，通过逐步加入噪声的方式，最终确定了随机森林算法在本环境下识别协议的性能更好，并对基评估器数量这个重要的参数进行了调整。

结　论

本文在研究 CCSDS 体系的基础上，研究的主要协议为 CCSDS 链路层的 AOS 协议、TM 协议、TC 协议，研究它们的结构与功能。此外，本文还对多种协议识别技术进行研究，主要包括模式串匹配技术，机器学习技术以及集成学习技术，并探究它们的原理以及对于协议识别的优劣性。最终，本项目采用集成学习的方法搭建协议识别的模型并完成了协议识别，并采用特定的标准检验了识别的准确性。

本论文的主要工作归纳如下：

1. 研究并分析了 CCSDS 协议体系，研究其结构与功能，并对其中链路层主要的几种协议进行研究，研究它们的结构、功能与异同。通过对 AOS 协议、TM 协议、TC 协议的字段比较可以得知，使用模式匹配技术可以容易地将 TC 协议与 AOS 和 TM 协议区分开，而由于 AOS 和 TM 协议共享同一个同步头，还需要使用集成学习的方法对它们进行准确的识别。

2. 研究并分析了多种协议识别的方法，主要包括包括简单的机器学习技术，以及几种经典的集成学习技术（比如 Bagging 思想和 Boosting 思想），并探究它们的原理以及对于协议识别的优劣性。通过本项目的比较可以得知，集成学习技术相较于传统的模式串匹配技术和简单的机器学习技术在识别协议上性能更好

3. 设计协议识别模型，分析需要识别的协议帧数据。采用集成学习的方法搭建协议识别的模型并完成了协议识别，并采用特定的标准检验了识别的准确性。通过逐步加入一定量的噪声来确定识别系统的抗干扰性，并对识别模型的参数进行了调整，最终本项目在几种经典的集成学习算法中确定了随机森林算法识别协议的性能更好，并且确定了本项目搭建好的识别系统具有高效性和较强的抗干扰性。

参考文献

［1］孟贤,秦大力,汪宇,等. 卫星星座网络容量密度评估［J］.天地一体化信息网络，2023，4(03)：67-78.

［2］郭晓旭,徐兆斌,徐可笛,等. 大规模星座组网通信的多址接入方式优化［D］.西北工业大学学报，2023，41(04)：644-653.

［3］郑杰. CCSDS-TC 协议的识别方法研究［D］.哈尔滨工业大学，2015.

［4］李新，李曦，彭雄根. 天地一体化网络应用及组网方案研究［J］. 电信快报. 2024(03)：1-4.

［5］韩帅，郭航，孟维晓等. 智能天地一体化网络的卫星跟踪测控技术综述［J］.遥测遥控，2024，45（1）：1–11.

［6］朱明，王春梅，姚秀娟，等. 集成学习在高误码率下 AOS 协议识别中的应用研究［A］. 宇航计测技术，2020，40(03):80-87.

［7］白云飞，陈晓敏，安军社，等. CCSDS 高级在轨系统协议及其应用介绍［J］.飞行器测控学报，2011，30（S1）：16-21.

［8］朱明. 基于集成学习的 CCSDS 空间链路层协议识别技术研究［D］. 中国科学院大学(中国科学院国家空间科学中心)，2020.DOI:10.27562/d.cnki.gkyyz.2020.000019.

［9］李旭航. 基于 XGBoost 的 SSH 流量识别研究［D］.哈尔滨理工大学，2019.

［10］Early J P , Brodley C E , Rosenberg C . Behavioral authentication of server flows［C］.Computer Security Applications Conference, 2003. Proceedings. 19th Annual. IEEE, 2004.

［11］Zander S, Nguyen T T, Armitage G, et al. Automated traffic classification and application identification using machine learning［C］. local computer networks, 2005：250-257.

［12］金凌. 面向比特流的未知帧头识别技术研究［D］.上海交通大学，2011.

［13］王和洲. 面向比特流的链路协议识别与分析技术［D］.中国科学技术大学，2014.

［14］张凤荔，周洪川，张俊娇，等. 零知识下的比特流未知协议分类模型［J］.计算机科学，2016，43(08)：39-44.

［15］雷东，王韬，赵建鹏，等. 面向比特流的未知协议识别与分析技术综述

［J］.计算机应用研究，2016,33(11)：3206-3210+3250.

［16］王旭芳. 基于模式匹配和机器学习的协议识别技术研究［D］.电子科技大学，2014.

［17］李相迎. CCSDS 数据链路层协议识别关键技术研究［D］. 北京：中国科学院研究生院博士学位论文，2011.

［18］姚秀娟,李雪. CCSDS 空间链路层协议识别技术研究［J］.航天电子对抗，2012，28(02).

［19］周军，吴侃侃，李林伟，等. CCSDS 统一空间数据链路协议应用分析［J］. 遥测遥控，2023，44（3）：40–46.

［20］李雪梅. 天地一体化异构网络融合技术研究［D］.西安电子科技大学，2018.

［21］TM Space Data Link Protocol. Recommendation for Space Data System Standards, CCSDS 132.0-B-1［S］. Blue Book. Issue 1. Washington, D.C.: CCSDS, September 2003.

［22］TC Space Data Link Protocol. Recommendation for Space Data System Standards, CCSDS 232.0-B-1［S］. Blue Book. Issue 1. Washington, D.C.: CCSDS, September 2003.

［23］郑娟,祝彬,陆静,周玉霞.AOS(高级在轨系统)协议跟踪研究［J］.航天标准化，2016(04)：33-38+45.

［24］明泽.基于主机日志的恶意登录异常检测方法研究［D］.中北大学，2021.

［25］杨青朋.高分辨率显著性目标检测算法研究［D］.河南大学，2022.

［26］陈露艳.卫星异构网络协议识别技术研究［D］.北京邮电大学，2017.

［27］林荣强.网络协议识别关键技术研究［D］.解放军信息工程大学，2015.

哈尔滨工业大学本科毕业论文（设计）
原创性声明和使用权限

本科毕业论文（设计）原创性声明

本人郑重声明：此处所提交的本科毕业论文（设计）《基于集成学习的CCSDS 空间链路层协议识别技术研究》，是本人在导师指导下，在哈尔滨工业大学攻读学士学位期间独立进行研究工作所取得的成果，且毕业论文（设计）中除已标注引用文献的部分外不包含他人完成或已发表的研究成果。对本毕业论文（设计）的研究工作做出重要贡献的个人和集体，均已在文中以明确方式注明。

作者签名：  日期： 2024 年 5 月 29 日

本科毕业论文（设计）使用权限

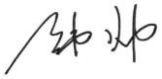
本科毕业论文（设计）是本科生在哈尔滨工业大学攻读学士学位期间完成的成果，知识产权归属哈尔滨工业大学。本科毕业论文（设计）的使用权限如下：

（1）学校可以采用影印、缩印或其他复制手段保存本科生上交的毕业论文（设计），并向有关部门报送本科毕业论文（设计）；（2）根据需要，学校可以将本科毕业论文（设计）部分或全部内容编入有关数据库进行检索和提供相应阅览服务；（3）本科生毕业后发表与此毕业论文（设计）研究成果相关的学术论文和其他成果时，应征得导师同意，且第一署名单位为哈尔滨工业大学。

保密论文在保密期内遵守有关保密规定，解密后适用于此使用权限规定。

本人知悉本科毕业论文（设计）的使用权限，并将遵守有关规定。

作者签名：  日期： 2024 年 5 月 29 日

导师签名：  日期： 2024 年 5 月 29 日

致　谢

大学四年的时光转瞬即逝，回想起大一还是个懵懵懂懂的学生，如今即将踏入社会的学习、工作中去，在毕设的完成过程中也是个很好的学习机会。值此毕设论文完成之际，我谨向这四年期间给予了我指导、帮助和支持的所有老师、同学、朋友和家人门致以最诚挚的感谢之意。

首先，要感谢我的毕设指导老师韩帅教授，从论文选题、相关资料查询到毕设设计工作开展，再到最后的论文构思写作，在这些过程中都给予了悉心、认真、负责的指导，在此期间我也收获了良多。在完成一项工作前要查找足够的资料，并进一步整理和调研，为完成工作打好基础。然后指定合理的计划，按照计划一步步脚踏实地的认真完成，不得粗心大意，遇到困难要勇于直面并坚信自己可以解决，这样才能交出一份满意的答卷。这些都是受指导老师的治学态度和工作作风感染，不仅是完成了一个毕设论文，也是学到了很多能够对以后学习、生活和工作产生长远积极影响的做人、做事方面的知识和能力。

其次，还要感谢一路走来给我大学生活增贴丰富和精彩的同学和朋友们。遇到难过的事会给我带来慰藉，遇到开心的事一起分享喜悦，遇到困难的事则共同协作应对。回首往昔，那些痛苦或快乐的日子历历在目，这些都会是我人生最值得怀念的时光，收获了友谊的同时也祝愿我们能够在未来的日子里继续努力、继续进步。

最后，肯定更要感谢我的父母和亲人们，父母给我创造了宝贵的学习机会，亲人们也提供很多关心和支持，这些都是我不断努力的力量源泉。母亲的关怀总是细致入微，关心学校的学习生活如何，关注各种小事，而父亲的关心是不善言辞但总能在自己力所能及范围之内给予最好的直接帮助。非常感谢父母的付出，也是我很好的学习榜样，我今天也会努力将自己的所学投入今天工作中去，通过实际行动去回报他们。

人生是一条蜿蜒前行的道路，路上会有很多风景，会遇到很多人和事，走走停停，也是到了开启下一段旅途的时间点了。总之，我在此次毕设中有所感、有所学、有所得，我会再接再厉，继续在人生的路上留下平稳而又深刻的足迹！

附　录 I

服务器流的行为认证

James P. Early，Carla E. Brodley，Catherine Rosenberg

摘要：理解流入和流出系统或网络的信息的本质是确定是否遵守使用策略的基础。确定流量类型的传统方法依赖于包头中携带的端口标签。然而，如果存在重新映射端口号的代理服务器或已被破坏的主机服务作为后门或隐蔽通道，则此方法可能会失败。

我们提出了一种基于在训练阶段学习到的决策树对服务器流量进行分类的方法。这些树是由使用我们设计的一组特征来描述的流量构建的，这些特征用于捕获流行为。因为我们对流量类型的分类是独立于端口标签的，所以它在存在恶意活动的情况下提供了更准确的分类。一项经验评估表明，聚合协议行为和主机特定协议行为的模型都获得了 82-100%的分类精度。

1．服务器流的身份验证需求

了解进出系统或网络的信息的性质对于确定是否遵守使用策略是至关重要的。如果没有这种理解，系统或网络的管理员就无法知道信息是否正在被泄露，他们的系统资源是否正在被适当地使用，或者攻击者是否正在使用服务对主机进行未经授权的访问。在这篇文章中，我们讨论了服务器流认证的问题，即对网络数据包流的服务器类型进行持续识别。具体地说，我们解决了一个问题，即我们是否可以根据测量流行为的特征正确识别流的 TCP 应用程序协议。

确定客户机-服务器协议的传统方法是检查 TCP 报头中的源和目的端口号。端口号与其对应的服务之间的映射关系就是众所周知的[2]。例如，HTTP 服务器的流量使用 80 端口，SMTP 服务器的流量使用 25 端口。本质上，我们依靠对流量的正确标记来准确确定其类型。端口标签和流量类型之间的绑定是一种约定俗成的绑定。这个标签也被用作防火墙过滤和入侵检测的依据[26,29]。

问题在于，在几种不同的攻击场景中，端口号可能并不能反映服务器流量的真实性质。

代理(proxy):这些服务器被用来整合一组用户对特定服务的访问。例如，web 代理用于处理对外部服务器的所有 HTTP 客户端请求。然而，也有代理存在的特定目的，以规避防火墙过滤规则的应用程序集[13]。在这种情况下，代理接收通常会被防火墙丢弃的流量，并重新映射端口号，使流量看起来像是 HTTP 流量。由于通常允许 HTTP 流量通过防火墙，因此该代理的用户能够绕过网络

策略。

服务器后门:当服务器被入侵时，攻击者通常会在一个或多个正在运行的服务[7]中放置一个“后门”。其目的是为攻击者提供一个入口，他/她可以使用它在以后的时间重新获得访问权限。来自此门户的流量将与受攻击服务的合法流量具有相同的端口号标签。攻击者可以将授权服务的二进制文件替换为可以同时作为和 Telnet 使用的二进制文件。当一个数据包从一个特定的源 IP 地址被接收时，流氓服务器知道执行 Telnet，否则执行服务。

用户安装的服务器:此类别包括安装未经授权的 Telnet、HTTP 或其他服务器，用于某些非法目的。它也代表着不断增长的对等文件共享网络[20]。这些服务器是由用户发起的，可以配置使用几乎任何端口。这一类还包括最近出现的“超级蠕虫”——通过电子邮件传播并携带自己的邮件服务器[12]的蠕虫。一旦安装，这些蠕虫利用它们的流氓邮件服务器转发未经请求的电子邮件信息，即垃圾邮件。如果事先不知道端口到服务的映射，流量的真实性质就无法确定。

这些场景中的每一个都代表了一个端口号标签无法准确指示流量类型的实例。更糟糕的是，正是在这些场景中，对流量的准确识别将揭示出受到损害的服务或策略违反。因此，存在着对与特定服务相关联的流量进行分类的需求，我们今后将称之为服务器流，使用一种方法而不是简单的标签，这种标签容易被修改，模棱两可，或隐藏未经授权的活动。

在检测主机上未授权服务存在的工具的设计上已经投入了大量的精力。这些工具的范围从检测服务器应用程序文件被修改的文件系统完整性工具(例如 Tripwire[15])到寻找成功入侵的工件的工具(例如 ChkRootKit[22])。成功使用这些工具需要适当的配置，在某些情况下，还需要怀疑已经发生了攻击。但是，运行这些工具的机器被入侵的事实使人们对这些工具报告的信息产生了怀疑。例如，如果一个 Linux 系统已经被一个可加载内核模块(LKM)[21]进行了根套件(root-kit)，那么从被入侵的主机[32]内部可能无法检测到这一点。这就提出了一种明显的可能性，即未经授权的服务可以无限期地不被检测到。

对于我们不能信任来自受攻击系统的结果，或者操作员对成功的攻击不知情的情况，有一个外部审计器以确保正确的服务器操作和/或检测未经授权的服务将是有益的。该审计人员使用的识别方法应避免使用端口号标签。相反，标识应该指示给定服务器流的适当行为。

在本文中，我们研究了如何根据服务器流的行为进行分类。其结果是一个监测网络流量的系统，以检查与预期的网络服务的一致性，并检测服务异常。本文的其余部分组织如下。第 2 节研究是否可以测量服务器流的行为特征。第

3 节讨论了测量这些特征的特征如何用于服务器识别。第 4 节提出了一个实证评估，说明我们可以根据它们的流行为的特征来区分服务器。第 5 节讨论了我们的分类方法如何可以与网络入侵检测系统集成。攻击者可能用来颠覆我们的分类系统的方法在第 6 节中介绍。相关工作在第 7 节中介绍。最后，在第 8 节中讨论了结论和未来的工作。

2．理解服务器流的本质

服务器流行为认证的关键问题是应该监控流量的哪些特征或特征。在担心用户隐私的环境中，或者使用加密来隐藏网络数据包中携带的数据时，我们不能依赖有效载荷的内容作为特征的来源。相反，我们通过检查数据包头和流量本身的操作特征来定义我们的特征集。

出于分析和实验的目的，我们将重点放在 HTTP、FTP、Telnet、SMTP 和 SSH 应用程序协议上。这些协议被很好地理解、稳定、广泛地实现，并代表了绝大多数用户流量[14]。

根据我们最初的观察，我们得出结论，基于 TCP 状态标志的特性(URG - Urgent, ACK - acknowledgement, PSH - Push, RST - Reset, SYN - synchronize 和 FIN - Finish)[3]可以在操作上区分服务器流行为。例如，HTTP 流量通常比 Telnet 流量包含更少的带有 PSH 标志的数据包。具体来说，对于每个标志，我们计算设置了该标志的大小包窗口中的包的百分比。除了这六个特征之外，我们还计算了数据包窗口的平均到达间隔时间和平均数据包长度。在监测过程中，分类方法会使用这些特征来确定之前的数据包是否与学习到的服务器流的行为相匹配。在下一节中，我们将描述如何为服务器流形成分类器。

3．服务器流的分类

在本节中，我们将描述如何将服务器流的行为认证视为有监督的机器学习问题。在监督学习中，给学习者一组观测值，每个观测值都标记为其中一个类。学习者的任务是从训练集中形成一个分类器，该分类器可用于将以前未见过的(和未标记的)观察结果分类为其中一个类。对许多基于数据挖掘/机器学习的异常检测系统的批评是，它们假设自己正在处理一个监督学习问题。也就是说，会给学习者提供正常和攻击数据[5]的例子。因为生成标记数据的行为是人力密集型的，所以认为一个人会收到特定主机的标记攻击数据是不现实的。在这种情况下，一个人会应用无监督学习来形成预期行为的模型。在监测过程中，人们寻找学习到的模型的异常。

然而，服务器身份验证可以自然地转换为监督学习任务或异常检测任务。要将问题转换为监督学习问题，我们必须选择可能的服务器应用程序，为每个

应用程序收集训练数据，然后应用监督学习算法来形成分类器。给定一个新的服务器流，然后我们可以将其分类为这些类型的服务器之一。为了将问题转换为异常检测问题，我们单独查看每个服务。对于每个感兴趣的服务器应用程序，我们形成一个正常行为的模型。给定一个新的服务器流程，我们将新流程与每个模型进行比较，以确定它是否符合这些模型中的任何一个。将问题铸造为异常检测问题使用与用户行为认证相同的框架[16,17]。在用户身份验证中，目标是确定用户在学习到的行为轮廓方面是否行为正常。

在本文中，我们选择了基于监督学习框架来研究服务器流认证，因为我们假设存在一个策略来指定要在给定主机上运行的服务。这种假设的一个缺点是，如果攻击者替换或更改现有的服务，它可能不会像任何允许的服务一样行为，这可能不会很容易被检测到。然而，它不太可能与任何被允许的服务表现相同，但我们计划在未来的工作中检查这一猜想。

4．实证评估

我们的实验旨在研究我们是否可以根据行为特征对服务器流进行分类。我们首先描述了实验中使用的数据和我们选择的监督学习算法。然后，我们展示了使用合成和真实网络流量学习聚合流和 by-host 流的实验结果。

4.1 数据来源

我们实验选择的第一个数据集是 1999 年 MIT Lincoln Labs 入侵检测评估数据集[1]。虽然是为特定的评估练习而创建的，但这些数据集随后被广泛用于研究其他后来的入侵检测系统，这些系统不属于原始评估的一部分[18,19,36]。

这些数据代表了一个虚构的空军基地五周的模拟网络流量。第一周到第三周构成了基于异常的入侵检测系统用于建模行为的训练数据。第一周和第三周的数据是无攻击的。每个星期有 5 个网络跟踪文件——每个工作日一个，代表大约从早上 8 点到下午 5 点的网络使用情况。每个文件都是 libpcap 格式(使用 tcp-dump 可读)，然后使用 gzip 压缩。平均来说，每周由大约 1 GB 的压缩数据组成，代表 2200 万个网络数据包。我们在训练集中使用了第一周的数据，在测试集中使用了第三周的数据。注意，我们没有使用攻击数据，因为我们的目的是评估我们是否可以对服务器行为进行分类——而不是我们是否可以检测入侵。我们的假设是入侵已经发生，并且攻击者已经实现了第 1 节中描述的场景之一。

除了林肯实验室的数据外，我们还包括使用从我们自己的网络获得的数据进行的实验。这里的目的是测试我们的方法在“真实世界”网络流量上的适用性。特别是，我们对一些较新的点对点文件共享协议的流量分类很感兴趣——

这是林肯实验室数据集不包含的。林肯实验室数据的人为性质引起了一些关注[4]，因此，一个额外的目标是确定这两个数据集的实验之间的任何显著差异。

4. 2 决策树分类器

我们选择使用决策树，因为它们提供了对其分类决策的易于理解的表示。尽管诸如 boosting[11,30]或支持向量机[6]之类的技术可能会获得稍高的分类精度，但它们在分类过程中需要更多的计算，并且进一步模糊了决策过程。

决策树是一种树结构，其中每个内部节点表示对一个特征的测试，每个分支表示测试的一个结果，叶节点表示类标签。图 1 展示了一个决策树的例子。为了对观测值进行分类，根节点测试特征 a 的值。如果结果大于某个值 x，则将该观测值标记为类 1。如果不是，我们下降右子树并测试特征 b 的值。测试继续进行，直到达到一个叶节点。叶节点上的标签提供了该观察的类标签。
我们选择使用 C5.0 决策树算法[27]，这是一种广泛使用并经过测试的实现。关于 C5.0 的具体细节，请参考文献[27,28]。在这里，我们只提供与决策树估计相关的算法的关键方面，特别是当它涉及特征选择时。决策树估计算法最重要的元素是用于估计树的每个内部节点上的分裂的方法。

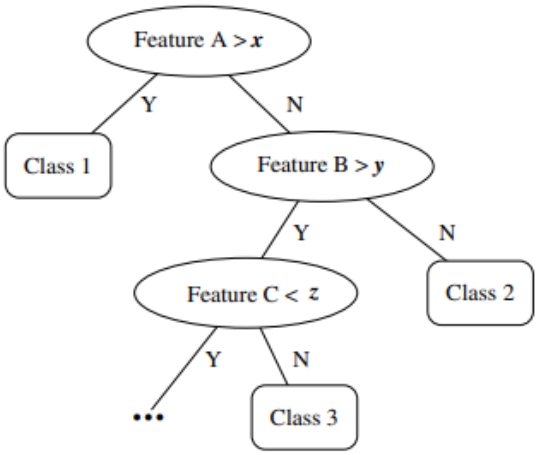


图 1：决策树抽象图显示了与某些特征相关联的值如何确定类标签。在本例中，特征 A 的值大于 x 的观测值被分配为类 1 的类标签。其他的分类是基于特征 B 和 C 的值。

要做到这一点，C5.0 使用一种称为信息增益比的度量来度量拆分产生的数据中熵的减少。在这个框架中，树内每个节点的测试是根据训练数据的分割来选择的，这些分割最大限度地减少了后代节点的熵。利用这个标准，训练数据被递归分割，这样在树的每个节点上增益比就最大化了。这个过程继续，直到每个叶节点只包含单个类的例子，或者通过进一步的测试没有信息增益。结果往往是一个非常大、复杂的树，过度拟合训练数据。如果训练数据包含错误，

那么以这种方式对数据过度拟合树会导致在看不见的数据上表现不佳。因此，当要对训练集之外的数据进行分类时，必须对树进行剪枝，以减少分类错误。为了解决这个问题，C5.0 使用了基于置信度的修剪，详细信息请参见[27]。

当使用决策树对未见过的示例进行分类时，C5.0 为其预测提供了类标签和置信度值。置信度值是一个从 0 到 1 的十进制数字- 1 意味着最高的置信度-它是针对每个实例给出的。

4.3 聚合服务器流模型

我们的第一个实验是设计用来确定 FTP、SSH、Telnet、SMTP 和 HTTP 流量可以在多大程度上使用决策树分类器进行区分。我们使用了林肯实验室第一周的数据构建我们的训练数据集。该集是通过首先为五个协议中的每一个随机选择 50 个服务器流来创建的。每个服务器流由从服务器到特定客户端主机/端口的数据包组成。最大的流包含大约 37000 个数据包，最小的流包含 5 个数据包。这 250 个流代表了大约 290,000 个包。我们将其称为聚合模型，因为流的集合来自许多不同的服务器。

这个数据被认证为无攻击的事实意味着我们可以对作为流量类型指示的端口号有信心。我们使用服务器端口来标记训练集中的每一个流量。然后，每个服务器流被用来根据我们的特征集生成数据观察。结果是一个由大约 29 万个标记的观测组成的数据集。我们对 7 个包窗口大小中的每一个都重复了这个过程。窗口大小是用于计算平均值和百分比的数据包数量的上界。如果单个流包含的包数少于包窗口大小，则可用包数用于计算每个观测值。

然后使用 C5.0 对七个训练集中的每个集构建决策树。我们以同样的方式构建测试集——每个协议的 50 个服务器流是从林肯实验室的第三周数据中随机选择的。这些然后传递给我们的特征提取算法使用相同的 7 个窗口大小。

```
tcpPerFIN > 0.01:
:...tcpPerPSH <= 0.4: www (45)
:   tcpPerPSH > 0.4:
:   :...tcpPerPSH <= 0.797619: smtp (13)
:       tcpPerPSH > 0.797619: ftp (38)
tcpPerFIN <= 0.01:
:...meanIAT > 546773.2:
    :...tcpPerSYN <= 0.03225806: telnet (6090)
    :   tcpPerSYN > 0.03225806:
    :   :...meanipTLen > 73.33: ftp (21)
    :       meanipTLen <= 73.33:
    :       :...tcpPerPSH > 0.7945206: smtp (8)
```

图 2：由 C5.0 生成的决策树的一部分

在描述如何使用树对流进行分类之前，我们在图 2 中给出了由 C5.0 生成的

决策树的一部分示例。在这个例子中，根节点用 FIN 标志集(tcpPerFIN)测试包窗口中包的百分比。如果这个百分比超过 1%，则对设置 PSH 标志(tcpPerPSH)的数据包的百分比进行测试。如果该值小于或等于 40%，则观察结果被归类为“www”，表示 HTTP 流量。圆括号中的数字表示用这个叶节点分类的训练的数量观测值。可以看到涉及平均到达间隔时间(meanIAT)和平均包长度(meanIPTLen)的其他测试。

Window Size	FTP	SSH	Telnet	SMTP	WWW
1000	100%	88%	94%	82%	100%
500	100%	96%	94%	86%	100%
200	98%	96%	96%	84%	98%
100	100%	96%	96%	86%	100%
50	98%	96%	96%	82%	100%
20	100%	98%	98%	82%	98%
10	100%	100%	100%	82%	98%

表 1：聚合模型决策树在未见过的单个服务器流上的分类精度。每个值代表每个协议的 50 个流中正确分类流的百分比。

在测试过程中，通过对流中每个观测值的置信度值求和来计算给定流的类标签。总置信度最高的类被分配给该流。分类结果如表 1 所示。对于 7 个窗口大小中的每一个，我们报告了每个协议的 50 个流的集合中正确分类的服务器流的百分比。从表中可以看出，分类准确率在 82%到 100%之间。

一般来说，SMTP 服务器流的分类精度低于其他协议。我们对错误分类的流进行了更详细的检查，发现这些流通常比正确分类的流长 2-4 倍。较长的 SMTP 服务器流表示较长的交互周期，因此包含越来越多的分类为 Telnet 或 FTP 的观察。在这些少数情况下，我们的特征集不足以区分这些流的行为。

使用更小的窗口大小是更可取的，因为这减少了检测服务正在发生异常的时间。实际上，对于 SSH，我们看到过大的数据包窗口大小(1000)会损害分类准确性。对于 FTP、SSH 和 Telnet，窗口大小小到 10 个数据包集就可以达到 100%的分类准确率。

因为提出的方法会被用来实时监控流量，所以我们粗略计算了一下分类时间。C5.0 对整个流程进行分类的平均时间长度为 70mS.1 训练是离线完成的，因此计算时间不太重要，但请注意，C5.0 创建每个决策树所用的平均时间长度为 22 秒。最后，我们需要解决维护一个值窗口来计算每个特征值的储存要求。我们可以通过仅保留每个特征的均值来近似存储所有值所创建的值，并对每个新数据包使用以下更新规则：

$$\frac{(n-1)\mu_{F_i} + new_{F_i}}{n}$$

在未来的工作中，我们将研究这种技术是否会显著降低性能。

我们从实验结果中得出结论，可以使用基于聚合流构建的决策树分类器来区分五种协议的服务器流的行为。稍后我们将讨论如何使用这种方法来补充入侵检测系统。

4.4 寄主专一性的模型

我们的第二个实验探讨了为特定主机创建模型是否提供了比聚合模型更好的性能。使用特定于主机的模型有三个优点：

1. 通过为单个服务器流创建模型，我们可以监控这些流的行为变化。

2. 特定于主机的模型可以捕捉在主机上运行的特定服务的实现微妙之处。这个解决方案在由许多服务器流组成的聚合模型中是缺失的。

3. 聚合模型中的训练示例将由产生最多流量的服务器主导。这可能会稀释来自其他服务器的示例。特定于主机的模型解决了这个问题。

我们首先在林肯实验室数据中确定了一组主机，每个主机运行三个或更多服务器协议。每台主机的训练数据是通过为运行的每个协议随机选择第一周的服务器流来收集的这些主机。每个模型中使用的流的数量被选择，这样每个协议都由相同数量的流表示。表 2 列出了每个主机的训练流和测试流的数量。

Host	Training Flows	Test Flows
172.16.112.100	20	20
172.16.112.50	30	25
172.16.113.50	35	23
172.16.114.50	10	20
197.218.177.69	25	35

表 2：每个主机模型在训练集和测试集上用于每个协议的流数。

根据我们使用聚合模型的结果，我们选择了一个 100 的包窗口大小来生成观测。选择是由以下事实驱动的:在聚合模型中使用这个窗口大小时，SMTP 的准确率是最高的，而其他协议分类的准确率在 96%到 100%之间。然后我们为每个主机训练了一个决策树，可以用来区分来自该主机的服务器流。测试数据从第三周开始以与训练数据相同的方式收集。

Host	FTP	SSH	Telnet	SMTP	WWW
172.16.112.100	95%	–	100%	90%	100%
172.16.112.50	92%	100%	84%	100%	–
172.16.113.50	100%	–	100%	100%	–
172.16.114.50	100%	95%	100%	95%	95%
197.218.177.69	100%	–	100%	100%	–

表 3：主机模型决策树在未见服务器流上的分类准确率。每行报告每个协议的主机地址和正确分类流的百分比。带“-”的字段表示该主机没有该协议类型的流量。

表 3 中的结果表明，一般情况下，主机特定模型取得了与聚合模型大致相同的分类精度。我们观察到的一个差异是，分类精度因协议而异。例如，对主机 172.16.112.50 的 Telnet 流的分类准确率为 84%，而对聚合模型中的 Telnet 流的分类准确率平均为 96.2%。对错误分类的 Telnet 流中的数据包进行检查，发现了一个有趣的现象。我们经常观察到数据包之间有很大的时间差距。时间间隔表示在 Telnet 服务器没有回显字符或对命令提供响应的情况下，用户活动出现了失误。在我们的框架中，单个大的间隔可以从根本上改变数据包的平均到达间隔时间的值，从而导致后续观察的错误分类。我们把这种现象称为“饮水机效应”——用户暂时离开交互会话，然后过一会儿又恢复。我们正在调查我们的分类器对这种效果的敏感度。一个可能的解决方案是基于一些时间间隔阈值细分流，并使用交互式子流来构建我们的分类器。

4.5 来自真实网络流量的模型

在本节中，我们将展示真实网络流量的实验。我们使用所描述的协议收集了大量的服务器流量。我们对这个集合进行了扩充，以包括来自充当 Kazaa 服务器的主机的流。Kazaa[20,31]是一个越来越受欢迎的点对点文件共享系统[8,33]。点对点网络流量不是林肯实验室数据集的一部分。

我们的目标是确定在使用合成流量与真实流量时，分类精度是否存在显著差异。我们观察到聚合模型和主机模型按协议的分类准确率从 85%到 100%不等。对于 100%的未见流，点对点流量被正确分类。这是一个特别有趣的结果，因为 Kazaa 流携带一个用户定义的端口标签。因此，我们能够正确地对端到端流进行行为分类-不需要使用端口号。这些结果表明，我们的分类方法对真实网络流量是有效的。其精度范围与合成数据的观测结果相吻合。因此，我们可以确定合成林肯实验室数据与真实网络流量中的每流行为之间没有明显差异。

5．入侵和误用检测的分类

这里提出的两类分类模型在入侵和误用检测的背景下产生了新的功能。聚合模型试图根据给定类型的许多流的一般行为对流进行分类。

聚合模型试图回答的问题是:“这个流类似于其他什么流?”相比之下，宿主模型是基于先前观察到的特定宿主的流量行为。给定一个看不见的流，主机模型试图回答这个问题:“这个流是否类似于先前来自这个主机的服务器流?”

入侵/误用检测系统和防火墙试图识别对系统或网络有害的先验行为。IDS可以被动地监视流量，并在出现某些攻击情况时发出警报。防火墙会主动丢弃违反某些网络策略的网络数据包。我们的分类方法试图在此类事件发生后识别表明入侵或滥用的活动。与先验机制协同工作，我们可以尝试在任何时间确定是否有即将发生的攻击或成功攻击的工件。

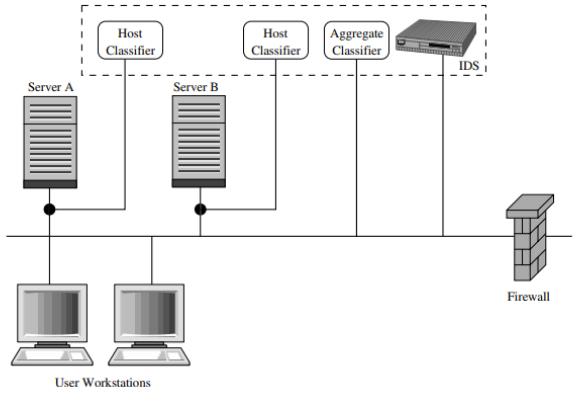


图 3：主机和聚合分类器的网络放置。主机分类器监控特定的服务器流，以查看与预期行为的偏差。聚合分类器监测用户流量，以确定流行为是否匹配其他同类型流的广义行为。

图 3 显示了如何将我们的分类方法集成到具有现有 IDS 的网络中。该组织使用服务器向某些用户社区提供网络服务(内部、外部或两者兼而有之)。我们的主机流分类系统直接监控这些服务器的输出。目的是确定当前观察到的流是否继续按照预期行为。如果攻击者设法控制了某个特定的服务，他/她将需要以一种精确匹配先前行为的方式与服务器进行交互。当与选定的主机地址组通信时，行为类似 Telnet 服务器的木马 web 服务器将与预期的主机模型不匹配，从而被检测到。

该网络将额外的用户流量传送到该组织外部的服务器。这些流量由聚合模型监控。在这里，我们对流量进行一般分类，并将其与端口标签进行比较。观察到类似于 Telnet 到某些非标准服务器端口的流量可能是安装了后门的指示。标记为 web 流量(服务器端口为 80)的流量行为更像 Telnet 流量，这可能表明存在用于逃避防火墙规则的代理。在某些用户定义的端口上运行的端到端的客户

端可能违反了网络策略。在每一种情况下，聚合分类器可以指示给定流是否以与其端口标签一致的方式表现。可能没有必要监视每个流——系统可以配置为随机选择一个流并尝试对其进行分类。如果该流通常与一个端口范围内不寻常或不受欢迎的流匹配，则可以识别和调查。

我们的方法可以在它自己的物理系统上运行，也可以是 IDS 或防火墙的一部分。这一决定将由系统预期监控的流量数量来驱动。在构建模型之后，系统会进行简单而快速的分类。

6．颠覆分类

鉴于上面描述的监控系统的存在，我们研究了攻击者操纵会话以影响服务器流分类的方法。我们之前在讨论饮水机效果时看到过一个这样的例子，尽管是一个无害的例子。在这里，用户暂停交互，从而导致数据包到达时间的变化，从而在服务器流的数据包窗口测量的平均到达间隔时间中可能出现很大的波动。攻击者可以做同样的事情。然而，目前还不清楚他/她是否能够使特定的分类被选择。更有可能的是，他/她将改变观察结果，从而导致某些不确定的类别被选择。如果使用宿主分类器，与预期行为的偏差将触发警报。

另一种方法可能涉及在发送到服务器的数据包中使用无关的 TCP 标志。一个例子就是在 HTTP 数据包中使用 URG 标志。TCP 标志在相应服务器流中的分布可能受到影响，也可能不受影响，这取决于该主机上服务器的实现。与时间的影响一样，我们将在未来的工作中调查分类器对这种操作的敏感性。

7．相关工作

之前在流识别方面的工作采用了各种技术和特征集。Dunigan 和 Ostrouchov 使用了两个特征(包到达间隔时间和长度)上的主成分分析(PCA)来为各种流类型[10]创建签名。他们报告的分类精度与我们的方法相当。然而，他们的方法需要离线分析。相比之下，我们的方法可以实时进行分类。

Tan 和 Collie 使用了基于单一特征(传输的总字节数)[35]构建的改进神经网络。他们的分析仅限于 Telnet 和 FTP 协议的分类。对于这些协议，他们报告的分类准确率通常低于我们的方法。

Daniels[9]使用基于单一特征(数据包的前 100 字节)构建的决策树对流进行分类。然而，分类精度被发现不足以实际使用。

有许多商业产品试图识别流量类型[23-25,34]。这些主要用于带宽分配的环境。例如，网络管理员创建一个策略，声明 web 流量不得超过总带宽的某个百分比，并在违反该策略时使用这些产品中的一种来有选择地丢弃流量。这些产品所使用的分类方法的许多细节是不公开的，因为它们是专有的。因此，我们

无法将我们的方法与这些产品中使用的方法进行比较。目前尚不清楚这些产品中是否有能够在存在恶意活动的情况下正确分类流(如第 1 节所述)。Packeteer[25]公司报告称，他们的产品使用“来自协议栈的所有 7 层”的信息来创建用于分类流的应用程序签名。如前所述，这样的系统在使用载荷加密或对用户隐私存在担忧的环境中可能合适，也可能不合适。

我们还确定了 Snort IDS[29]的一个组件，该组件用于对服务器流进行分类。然而，这个系统依赖于端口号和 TCP 3 次握手的检测。如前所述，在代理或妥协服务存在的情况下，此系统不太可能对流进行正确分类。

关于我们的特征集，NATE(网络异常流量事件分析)[36]系统也是基于 TCP 标志。NATE 使用主成分分析来识别 TCP 状态标志可以检测到某些类型的攻击。我们的方法在两个方面有所不同。首先，NATE 系统尝试对正常流量和攻击流量之间的差异进行建模。他们不试图对协议之间的行为差异进行建模。第二个区别涉及 NATE 使用聚类来识别异常。这种方法必须离线完成，因此限制了系统在需要接近实时检测的环境中的有用性。相比之下，一旦创建了决策树，我们的系统就可以实时监测数据包。

8．结论

我们提出了一种新颖的方法来定义一组特征来模拟服务器流流量的操作行为。我们通过使用 C5.0 决策树算法演示了我们的特征可以以 82%到 100%的准确率区分服务器协议的行为。我们通过经验说明，聚合模型可以将一个未见过的服务器流分类为属于一个以前见过的流的家族，并且主机模型可以确定来自给定服务器的流是否与以前见过的该服务器的流的行为匹配。这些分类器可以增强传统的入侵检测系统，以检测成功攻击的伪影。我们的分类技术是独立于数据包标记的，因此不受修改端口号以隐藏活动的技术的影响。

决策树分类器可能对数据包到达时间间隔的波动很敏感。这在我们所说的饮水机效应中得到了例证。我们计划研究如何减轻这种敏感性，以提高某些协议的分类精度。

我们打算进一步研究我们的功能集的使用和扩展，以对其他类型的服务器流进行建模。我们的技术应用将扩展到其他传输协议，特别是那些由点对点文件共享系统使用的协议。其他功能集正在开发中，以模拟 UDP 流量、ICMP 流量和选定的路由协议。

9．致谢

本研究得到 AFRL 资助，资助编号为 F30602-02-2-0217。

参考文献

[1] 1999 darpa intrusion detection evaluation data set. URL http://www.ll.mit.edu/IST/ ideval/data/1999/1999_data_index. html.

[2] RFC 1700: Assigned Port Numbers. URL ftp:// ftp.internic.net/rfc/rfc1700.txt.

[3] RFC 793 - Transmission Control protocol. URL ftp: //ftp.internic.net/rfc/rfc0793.txt.

[4] Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. ACM Transactions on Information and System Security (TISSEC), 3(4):262–294, 2000. ISSN 1094-9224.

[5] S. Barbara and S. Jajodia. Applications of Data Mining in Computer Security. Kluwer Academic Publishers, 2002.

[6] K. Bennett and C. Campbell. Support Vector Machines: Hype or Hallelujah? SIGKDD Explorations, 2:1–13, 2000.

[7] J. Boxmeyer. ONCTec - List of Possible Trojan/Backdoor Port Activity. ONCTek, llc. URL http://www.onctek.com/trojanports.
html.

[8] Cornell University Student Assembly Committee on Information and Technologies and ResNet. Cornell Internet Usage Statistics. URL http: //www.cit.cornell.edu/computer/
students/bandwidth/charts.html.

[9] Thomas E. Daniels. personal communication, September 2003.

[10] Tom Dunigan and George Ostrouchov. Flow Characterization for Intrusion Detection. Technical report, Oak Ridge National Laboratory, November 2000.

[11] Y. Freund. Boosting a Weak Learning Algorithm by Majority. Information and Computation, 121(2):256– 285, 1995.

[12] Saul Hansell. E-mail’s Backdoor Open to Spammers. New York Times, May 20 2003.

[13] iNetPrivacy Software Inc. Antifirewall. URL http: //www.antifirewall.com/intro.htm.

[14] Internet Technical Resources. Traffic Statistics. URL http://www.cs.columbia.edu/~hgs/ internet/traffic.html.

[15] G. H. Kim and G. Spafford. The Design and Implementation of Tripwire: A File System Integrity Checker. In ACM Conference on Computer and Communications Security, pages 18–29, 1994. URL citeseer.nj.nec.com/article/ kim94design.html.

[16] T. Lane and C. E. Brodley. Temporal Sequence Learning and Data Reduction

for Anomaly Detection. In Proceedings of the Fifth ACM Conference on Com- puter and Communications Security, pages 150–158. Association for Computing Machinery, Nov 1998.

[17] T. Lane and C. E. Brodley. Temporal Sequence Learning and Data Reduction for Anomaly Detection. ACM Transactions on Computer Security, 2(3):295– 331, 1999.

[18] W. Lee and S. Stolfo. A Framework for Construct- ing Features and Models for Intrusion Detection Systems. ACM Transactions on Information and System Security, 3(4):227–261, November 2000.

[19] M. Mahoney and P. K Chan. PHAD: Packet Header Anomaly Detection for Indentifying Hostile Network Traffic. Technical Report CS-2001-4, Florida Tech, 2001.

[20] Evangelos P. Markatos. Tracing a Large-scale Peer to Peer System : an Hour in the Life of Gnutella. Technical Report 298, 2001. URL citeseer.nj.nec. com/markatos01tracing.html.

[21] T.Miller.Detecting Loadable Kernel Modules (LKM). URL http://www. incident-response.org/LKM.htm.

[22] N. Murilo and K. Steding-Jessen. chkrootkit: A Tool that Locally Checks for Signs of a Rootkit. URL http://www.chkrootkit.org/.

[23] NetScreen Technologies, Inc. NetScreen-5000 Series. URL http://www.netscreen.com/ products/datasheets/ds_ns_5000.jsp.

[24] Captus Networks. Captus ips 4000 series. URL http://www.captusnetworks.com/.

[25] Packeteer, Inc. Packeteer PacketShaper. URL http://www.packeteer.com/resources/prod-sol/PSDS.pdf.

[26] P. A. Porras and P. G. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In Proc. 20th NIST-NCSC National In- formation Systems Security Conference, pages 353– 365, 1997. URL http://citeseer.nj.nec. com/porras97emerald.html.

[27] J. R. Quinlan. C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.

[28] Ross Quinlan. Data Mining Tools See5 and C5.0. URL http://www.rulequest.com/ see5-info.html.

[29] M. Roesch and C. Green. Snort - The Open Source Network Intrusion Detection System. URL http:// www.snort.org/.

[30] Robert E. Schapire. A Brief Introduction to Boosting. In IJCAI, pages 1401–1406, 1999. URL citeseer. nj.nec.com/schapire99brief.html.

[31] Sharman Networks Ltd. . Kazaa Media. URL http: //www.kazaa.com/us/.

[32] E. Skoudis. Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses. Prentice Hall, 2002.

[33] J. St. Sauver. Percentage of Total Internet2 Traf- fic Consisting of Kazaa/Morpheus/FastTrack - Uni- versity of Oregon. In Collaborative Computing in Higher Education: Peer-to-Peer and Beyond Work- shop. URL http://darkwing.uoregon.edu/ ˜joe/kazaa.html.

[34] Stampede Technologies, Inc. Turbogold enterprise edition. URL http://www.stampede.com/ productsclienttoserverfeaturesTraffic. html.

[35] K. M. C. Tan and B. S. Collie. Detection and Classi- fication of TCP/IP Network Services. In Proceedings of the Thirteenth Annual Computer Security Applica- tions Conference, pages 99–107, San Diego, Califor- nia, December 1997.

[36] Carol Taylor and Jim Alves-Foss. NATE: Network Analysis of Anomalous Traffic Events, a low-cost ap- proach. In Proceedings of the 2001 Workshop on New Security Paradigms, pages 89–96. ACM Press, 2001. ISBN 1-58113-457-6.

附　录 II

Behavioral Authentication of Server Flows

James P. Early[1]　　Carla E. Brodley[2]　　Catherine Rosenberg[2]

[1]CERIAS
Purdue University
West Lafayette IN 47907-2086
earlyjp@cerias.purdue.edu

[2]School of Electrical and Computer Engineering
Purdue University
West Lafayette, Indiana 47907-2035
{brodley, cath}@ecn.purdue.edu

Abstract

Understanding the nature of the information flowing into and out of a system or network is fundamental to determining if there is adherence to a usage policy. Traditional methods of determining traffic type rely on the port label carried in the packet header. This method can fail, however, in the presence of proxy servers that re-map port numbers or host services that have been compromised to act as backdoors or covert channels.

We present an approach to classify server traffic based on decision trees learned during a training phase. The trees are constructed from traffic described using a set of features we designed to capture stream behavior. Because our classification of the traffic type is independent of port label, it provides a more accurate classification in the presence of malicious activity. An empirical evaluation illustrates that models of both aggregate protocol behavior and host-specific protocol behavior obtain classification accuracies ranging from 82-100%.

1. The Need for Authentication of Server Flows

Understanding the nature of the information flowing into and out of a system or network is fundamental to determining if there is adherence to a usage policy. Without this understanding, administrators of systems or networks can not know if information is being compromised, if their system resources are being used appropriately, or if an attacker is using a service for unauthorized access to a host. In this paper we address the problem of server flow authentication – the on-going identification of server type for a stream of network packets. Specifically, we address the question of whether we can correctly identify the TCP application protocol of a flow based on features that measure the behavior of the flow.

The traditional method of determining the client-server protocol is by inspecting the source and destination port numbers in the TCP header. The mappings between port numbers and their corresponding service are well known [2]. For example, HTTP server traffic uses port 80, and SMTP server traffic uses port 25. In essence, we rely on a correct *labeling* of the traffic to accurately determine its type. The binding between the port label and the type of traffic is a *binding by convention*. This label is also used as a basis for firewall filtering and intrusion detection [26, 29].

The problem lies in that there are several different attack scenarios for which the port number may not be indicative of the true nature of the server traffic.

Proxies: These servers are used to consolidate access to a particular service for a group of users. For example, web proxies are used to handle all HTTP client requests to external servers. However, there are also proxies that exist for the specific purpose of evading firewall filtering rules for set of applications [13]. In this case, the proxy takes traffic that would normally be dropped by the firewall and remaps the port numbers to make the traffic appear to be HTTP traffic. Because HTTP traffic is routinely allowed to pass through firewalls, the user of this proxy is able to circumvent the network policy.

Server Backdoors: When a server has been compromised, the attacker often places a "backdoor" in one or more of the running services [7]. The purpose is provide the attacker with a portal that he/she can use to regain access at a later time. Traffic from this portal will have the same port number label as legitimate traffic for the compromised service. The attacker may replace the binary of an authorized service X with a binary that can function as both X and Telnet. When a packet is received from a particular source IP, the rogue server knows to execute Telnet, otherwise it executes service X.

User-Installed Servers: This category includes the installation of unauthorized Telnet, HTTP, or other servers for some illicit purpose. It also represents the increasing numbers of peer-to-peer file sharing networks [20]. These servers are initiated by the user and can be configured to use almost any port. This category also includes the recent appearance of "super worms" - worms that propagate via e-mail and carry their own mail server [12]. Once installed, these worms utilize their rogue mail server to forward unsolicited e-mail messages i.e., Spam. Without prior knowledge of a port to service mapping, the true nature of the traffic cannot be determined.

Each of these scenarios represents an instance where the port number label fails to accurately indicate the type of traffic. Worse yet, it is precisely these scenarios where an accurate identification of the traffic would reveal a compromised service or policy violation. Thus, there exists a need to classify traffic associated with a particular service, what we will henceforth refer to as a *server flow*, using a method other than a mere label that is easily modified, ambiguous, or conceals unauthorized activity.

Significant effort has been invested in the design of tools for detecting the presence of unauthorized services on a host. These range from file system integrity tools that detect modification to server application files (e.g., Tripwire [15]) to tools that look for artifacts of successful intrusions (e.g., ChkRootKit [22]). Successful use of these tools requires proper configuration and, in some cases, a suspicion that an attack has occurred. But, *the fact that a machine running these tools was compromised casts doubt on the information these tools report.* For example, if a Linux system has been root-kitted by a Loadable Kernel Module (LKM) [21], it may be impossible to detect this from *inside* the compromised host [32]. This raises the distinct possibility that an unauthorized service can go undetected indefinitely.

For situations where we cannot trust the results from a compromised system, or the operator is unaware of a successful attack, it would be beneficial to have an external auditor for the purpose of ensuring proper server operation and/or detecting unauthorized services. The identification method used by this auditor should eschew port number labels. Rather, the identification should be indicative of the proper *behavior* of a given server flow.

In this paper, we investigate how server flows can be classified based on their behavior. The result is a system that monitors network traffic to check conformity with expected network services and to detect service anomalies. The remainder of this paper is organized as follows. Section 2 investigates whether behavioral characteristics of server flows can be measured. Section 3 discusses how features measuring these characteristics can be used for server identification. Section 4 presents an empirical evaluation that illustrates that we can discriminate among servers based on characteristics of their flow behavior. Section 5 discusses how our classification method can be integrated with network intrusion detection systems. Methods an attacker might use to subvert our classification system are presented in Section 6. Related work is presented in Section 7. Finally, conclusions and future work are discussed in Section 8.

2. Understanding the Nature of Server Flows

The key issue in the behavioral authentication of server flows is what characteristics or *features* of the traffic should be monitored. In environments where there are concerns about user privacy, or where encryption is used to hide the data carried in network packets, we cannot rely on the contents of the payload as a source of features. Rather, we examine the packet header and the operational characteristics of the traffic itself to define our feature set.

For the purposes of our analysis and experiments, we focused on the HTTP, FTP, Telnet, SMTP, and SSH application protocols. These protocols are well understood, stable, widely implemented, and represent the vast majority of user traffic [14].

Based on our initial observations, we concluded that features based on the TCP state flags (URG - Urgent, ACK - Acknowledgment, PSH - Push, RST - Reset, SYN - Synchronize, and FIN - Finish) [3] can operationally differentiate server flow behavior. For example, HTTP traffic generally contains far fewer packets with the PSH flag than does Telnet traffic. Specifically, for each of the flags, we calculate the percentage of packets in a window of size n packets with that flag set. In addition to these six features we calculate the mean inter-arrival time and the mean packet length for the window of n packets. During monitoring, these features are used by the classification method to determine whether the previous n packets match the learned behavior of the server flows. In the next section we describe how we form a classifier for server flows.

3. Classification of Server Flows

In this section we describe how we can view behavioral authentication of server flows as a supervised machine learning problem. In supervised learning, the learner is given a set of observations each labeled as one of k classes. The learner's task is to form a classifier from the *training set* that can be used to classify previously unseen (and unlabeled) observations as one of the k classes. A criticism of many anomaly detection systems based on data mining/machine learning is that they assume that they are dealing with a supervised learning problem. That is, the learner will be given examples of both normal and attack

data [5]. It is unrealistic to think that one will receive labeled attack data for a particular host because the act of generating labeled data is human intensive. In such cases, one applies unsupervised learning to form a model of expected behaviors. During monitoring one looks for anomalies with respect to the learned model.

However, server authentication can be naturally cast as either a supervised learning task or an anomaly detection task. To cast the problem as a supervised learning problem we must choose k possible server applications, collect training data for each, and then apply a supervised learning algorithm to form a classifier. Given a new server flow we can then classify it as one of these k types of servers. To cast the problem as an anomaly detection problem we look at each service individually. For each of the k server applications of interest we form a model of normal behavior. Given a new server flow, we compare the new flow to each of the models to determine whether it conforms to any of these models. Casting the problem as an anomaly detection problem uses the same framework as user behavioral authentication [16, 17]. In user authentication the goal is to identify whether the user is behaving normally with respect to a learned profile of behavior.

In this paper we have chosen to investigate server flow authentication based on the supervised learning framework, because we assume a policy exists specifying the services that are to be run on a given host. A drawback of this assumption is that if an attacker replaces or alters an existing service it may not behave like any of the permitted services, and this may not be readily detectable. However, it is unlikely that it will behave *identically* to any of the permitted services, but we plan to examine this conjecture in future work.

4. An Empirical Evaluation

Our experiments are designed to investigate whether we can classify server flows based on features of behavior. We first describe the data used in the experiments and the supervised learning algorithm we chose. We then present experimental results with learning aggregate flows and by-host flows using both synthetic and real network traffic.

4.1. Data Sources

The first data set chosen for our experiments is the 1999 MIT Lincoln Labs Intrusion Detection Evaluation Datasets [1]. Although created for a specific evaluation exercise, these datasets have subsequently been widely used for research into other later intrusion detection systems not part of the original evaluation [18, 19, 36].

The data represent five weeks of simulated network traffic from a fictional Air Force base. Weeks one through three constitute the *training* data used by anomaly-based intrusion detection systems to model behavior. The data in week one and week three are attack-free. There are five network trace files for each week – one for each business day representing network usage from approximately 8:00 AM to 5:00 PM. Each file is in libpcap format (readable with tcpdump), then compressed using gzip. On average, each week consists of roughly 1 GB of compressed data representing 22 million network packets. We used data from week one in our training sets and data from week three in our test sets. Note that we do not use the attack data, since our purpose is to evaluate whether we can classify server behavior – not whether we can detect intrusions. Our assumption is that the intrusion has already occurred and that an attacker has implemented one of the scenarios described in Section 1.

In addition to the Lincoln Labs data, we include experiments using data obtained from our own network. The purpose here is to test the applicability of our method on "real world" network traffic. In particular, we are interested in classifying traffic from some of the newer peer-to-peer file sharing protocols – something that the Lincoln Labs data sets do not contain. Some concerns have been raised about the artificial nature of the Lincoln Labs data [4], and thus an additional objective was to identify any marked differences between experiments with these two data sets.

4.2. Decision Tree Classifier

We chose to use decision trees because they provide a comprehensible representation of their classification decisions. Although techniques such as boosting [11, 30] or support vector machines [6] might obtain slightly higher classification accuracy, they require more computation during classification and further they obscure the decision making process.

A decision tree is a tree structure where each internal node denotes a test on a feature, each branch indicates an outcome of the test, and the leaf nodes represent class labels. An example decision tree is shown in Figure 1. To classify an observation, the *root* node tests the the value of feature A. If the outcome is greater than some value x, the observation is given a label of *Class 1*. If not, we descend the right subtree and test the value for feature B. Tests continue until a leaf node is reached. The label at the leaf node provides the class label for that observation.

We chose to use the C5.0 decision tree algorithm [27] – a widely used and tested implementation. For details regarding the specifics of C5.0 the reader is referred to [27, 28]. Here we provide only the key aspects of the algorithm related to decision tree estimation, particularly as it pertains to feature selection. The most important element of the decision tree estimation algorithm is the method used to estimate splits at each internal node of the tree. To do this,

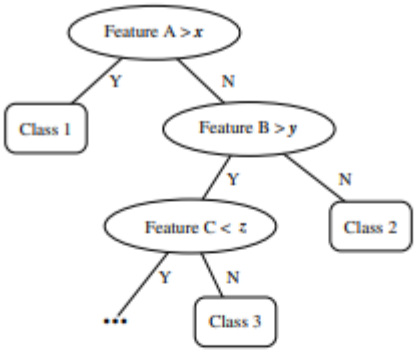


Figure 1. Decision tree abstraction showing how the values associated with certain features determine the class label. In this example, observations whose value for feature *A* is greater than *x* are assigned a class label of *Class 1*. Other classifications are based on the values of features *B* and *C*

C5.0 uses a metric called the *information gain ratio* that measures the reduction in entropy in the data produced by a split. In this framework, the test at each node within a tree is selected based on splits of the training data that maximize the reduction in entropy of the descendant nodes. Using this criteria, the training data is recursively split such that the gain ratio is maximized at each node of the tree. This procedure continues until each leaf node contains only examples of a single class or no gain in information is given by further testing. The result is often a very large, complex tree that overfits the training data. If the training data contains errors, then overfitting the tree to the data in this manner can lead to poor performance on unseen data. Therefore, the tree must be pruned back to reduce classification errors when data outside of the training set are to be classified. To address this problem C5.0 uses confidence-based pruning, and details can be found in [27].

When using the decision tree to classify unseen examples, C5.0 supplies both a class label and a confidence value for its prediction. The confidence value is a decimal number ranging from zero to one – one meaning the highest confidence – and it is given for each instance.

4.3. Aggregate Server Flow Model

Our first experiment was designed to determine the extent to which FTP, SSH, Telnet, SMTP, and HTTP traffic can be differentiated using a decision tree classifier. We used the data from week one of the Lincoln Labs data to

```
tcpPerFIN > 0.01:
:...tcpPerPSH <= 0.4: www (45)
:   tcpPerPSH > 0.4:
:   :...tcpPerPSH <= 0.797619: smtp (13)
:       tcpPerPSH > 0.797619: ftp (38)
tcpPerFIN <= 0.01:
:...meanIAT > 546773.2:
    :...tcpPerSYN <= 0.03225806: telnet (6090)
    :   tcpPerSYN > 0.03225806:
    :   :...meanipTLen > 73.33: ftp (21)
    :       meanipTLen <= 73.33:
    :       :...tcpPerPSH > 0.7945206: smtp (8)
```

Figure 2. Portion of a decision tree generated by C5.0.

build our training dataset. The set was created by first randomly selecting fifty server flows for each of the five protocols. Each server flow consists of the packets from a server to a particular client host/port. The largest flow contained roughly 37,000 packets, and the smallest flow contained 5 packets. The 250 flows represented a total of approximately 290,000 packets. We refer to this as an *aggregate model* because the collection of flows came from many different servers.

The fact that this data is certified as attack-free meant that we could have confidence in the port numbers as indicative of the type of traffic. We used the server port to label each of flows in the training set. Each server flow was then used to generate data observations based on our feature set. The result is a data set consisting of approximately 290,000 thousand labeled observations. We repeated this process for each of seven packet window sizes. The window size is an upper bound on the number of packets used to compute the means and percentages. If an individual flow contains fewer packets than the packet window size, the number of available packets is used to calculate each observation.

Each of the seven training sets was then used to build a decision tree using C5.0. We constructed test sets in the same manner – fifty server flows from each protocol were randomly selected from week three of the Lincoln Labs data. These were then passed to our feature extraction algorithm using the same seven window sizes.

Before describing how a tree is used to classify a flow, we give an example of a portion of a decision tree generated by C5.0 in Figure 2. In this example, the root node tests the percentage of packets in the packet window with the FIN flag set (tcpPerFIN). If this percentage exceeds 1%, a test is made on the percentage of packets with the PSH flag set (tcpPerPSH). If this value is less than or equal to 40%, the observation is classified as "www", indicating HTTP traffic. The numbers in parenthesis indicate the number of train-

Window Size	FTP	SSH	Telnet	SMTP	WWW
1000	100%	88%	94%	82%	100%
500	100%	96%	94%	86%	100%
200	98%	96%	96%	84%	98%
100	100%	96%	96%	86%	100%
50	98%	96%	96%	82%	100%
20	100%	98%	98%	82%	98%
10	100%	100%	100%	82%	98%

Table 1. Classification accuracy of the aggregate model decision trees on unseen individual server flows. Each value represents the percentage of correctly classified flows out of the fifty flows for each protocol

ing observations classified with this leaf node. Other tests can be seen involving the mean inter-arrival time (meanIAT) and mean packet length (meanIPTLen).

During testing, the class label for a given flow was calculated by summing the confidence values for each observation in the flow. The class with the highest total confidence was assigned to that flow. The classification results are shown in Table 1. For each of seven window sizes, we report the percentage of correctly classified server flows out of the set of fifty flows for each protocol. As can be seen in the table, the classification accuracy ranges from 82% to 100%.

In general, the classification accuracy was lower for SMTP server flows than for other protocols. We examined the misclassified flows in more detail and discovered that these flows were generally 2-4 times longer than correctly classified flows. Longer SMTP server flows represented longer periods of interaction, and thus contain increasing numbers of observations classified as Telnet or FTP. In these few cases, our feature set is not adequate for discriminating between the behaviors of these flows.

It is more desirable to use a smaller window size because this decreases the time to detect that a service is behaving abnormally. Indeed for SSH we see that too large a packet window size (1000) hurts classification accuracy. For FTP, SSH and Telnet, a window size as small as ten packets achieves 100% classification accuracy.

Because the proposed method would be used to monitor traffic in real time, we did a rough calculation of classification time. The average length of time used by C5.0 to classify an entire flow was 70mS.[1] Training is done offline so computation time is of lesser importance, but note that the average length of time used by C5.0 to create each decision tree was 22 seconds. Finally, we need to address the storage requirements for maintaining a window of n values to compute the value of each of the features. We can approximate the value created by storing all n values by retaining only the mean for each feature, μ_{F_i} and using the following update rule for each new packet:

$$\frac{(n-1)\mu_{F_i} + new_{F_i}}{n}$$

In future work we will investigate whether this technique significantly degrades performance.

We conclude from our experimental results that the behavior of server flows for the five protocols can be differentiated using a decision tree classifier built on aggregate flows. We will later discuss how this method can be used to compliment an intrusion detection system.

4.4. Host-Specific Models

Our second experiment addresses whether creating models for specific hosts provides better performance than the aggregate model. There are three advantages to using host-specific models:

1. By creating models for individual server flows, we can monitor these flows for changes in behavior.
2. A host-specific model can capture the implementation subtleties of a particular service running on a host. This resolution is missing in the aggregate model consisting of many server flows.
3. The training examples in an aggregate model will be dominated by the server generating the most traffic. This may dilute examples from other servers. The host-specific model solves this problem.

We first identified a set of hosts in the Lincoln Labs data that each ran three or more server protocols. Training data for each host was collected by randomly selecting server flows from week one for each of the protocols running on

1 The hardware platform used for building the decision trees and classifying observations was a 500Mhz Dual Pentium III PC with 772MB of RAM running Red Hat Linux (kernel version 2.4.18).

Host	FTP	SSH	Telnet	SMTP	WWW
172.16.112.100	95%	–	100%	90%	100%
172.16.112.50	92%	100%	84%	100%	–
172.16.113.50	100%	–	100%	100%	–
172.16.114.50	100%	95%	100%	95%	95%
197.218.177.69	100%	–	100%	100%	–

Table 3. Classification accuracy of host model decision trees on unseen server flows. Each row reports the host address and the percentage of correctly classified flows for each protocol. Fields with a "–" indicate there was no traffic of this protocol type for this host.

Host	Training Flows	Test Flows
172.16.112.100	20	20
172.16.112.50	30	25
172.16.113.50	35	23
172.16.114.50	10	20
197.218.177.69	25	35

Table 2. Number of flows used for each protocol in training and test sets for each host model

these hosts. The number of flows used in each model was chosen such that each protocol was represented by the same number of flows. Table 2 lists the number of training and test flows per host.

Based on our results using the aggregate models, we chose a packet window size of 100 for generating observations. The selection was driven by the fact that SMTP accuracy was greatest using this window size with the aggregate models, and other protocol classifications accuracies were between 96% and 100%. We then trained a decision tree for each host that could be used to differentiate the server flows coming from that host. Test data was collected from week three in the same manner as the training data.

The results in Table 3 indicate that, in general, the host specific models achieve approximately the same classification accuracy as the aggregate models. One difference observed is that classification accuracy varies by protocol. For example, the classification accuracy of Telnet flows for host *172.16.112.50* is 84% whereas the classification of Telnet flows in the aggregate models averaged 96.2%. Examination of the packets in the misclassified Telnet flows revealed an interesting phenomenon. We often observed large time gaps between packets. The time gaps indicate lapses in user activity where the Telnet server is not echoing characters or supplying responses to commands. In our framework, a single large gap can radically alter the values for the mean inter-arrival time of packets, thus resulting in misclassification of the subsequent observations. We refer to this as the *Water Cooler Effect* – the user temporarily leaves the interactive session, then resumes a short while later. We are investigating the sensitivity of our classifiers to this effect. One possible solution would be to subdivide flows based on some time gap threshold and use the interactive sub-flows to build our classifiers.

4.5. Models from Real Network Traffic

In this section we present experiments with real network traffic. We collected a number of server flows using the protocols described. We augmented this set to include flows from hosts acting as Kazaa servers. Kazaa [20, 31] is a peer-to-peer file sharing system that is growing in popularity [8, 33]. Peer-to-peer network traffic was not part of the Lincoln Labs dataset.

Our goal was to determine if there was a significant difference in classification accuracy when using synthetic versus real traffic. We observed classification accuracies by protocol ranging from 85% to 100% for both the aggregate and host models. The peer-to-peer traffic was classified correctly for 100% of the unseen flows. This is an especially interesting result because Kazaa flows carry a port label that is user-defined. Thus, we are able to correctly classify peer-to-peer flows behaviorally – without the use of the port number. These results indicate that our classification method is effective for real network traffic. The range of accuracies match those observed with the synthetic data. Thus, we can identify no appreciable difference in the per-flow behavior in the synthetic Lincoln Labs data versus those in real network traffic.

5. Classification for Intrusion and Misuse Detection

The two types of classification models presented here give rise to new functionality in the context of intrusion and misuse detection. Aggregate models try to classify a flow based on the general behavior of many flows of a given type.

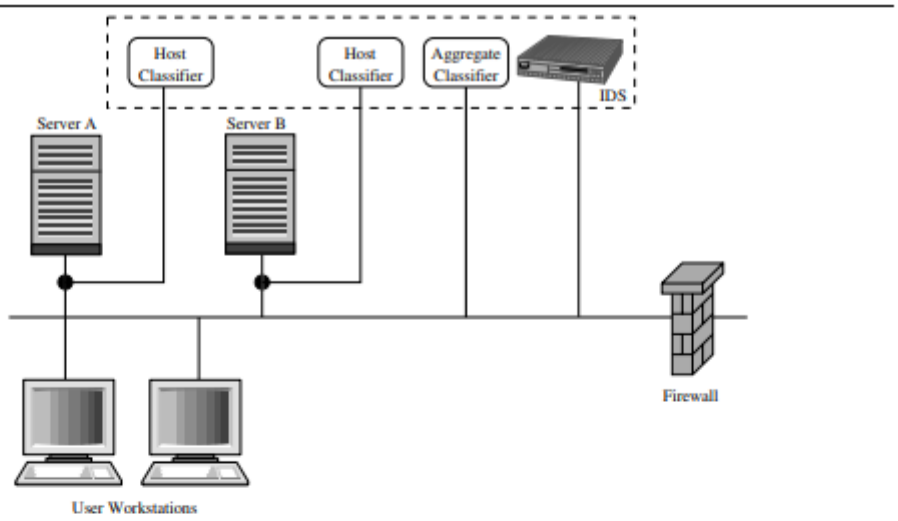


Figure 3. Network placement of the host and aggregate classifiers. Host classifiers monitor specific server flows for deviations from expected behavior. Aggregate classifiers monitor user traffic to determine if flow behavior matches generalized behavior of other flows of the same type.

The question the aggregate model tries to answer is: "What other flows does this flow resemble?" In contrast, host models are based on the previously observed behavior of flows for a specific host. Given an unseen flow, the host models try to answer the question: "Does this flow resemble previous server flows from this host?"

Intrusion/misuse detection systems and firewalls try to identify actions a priori as being harmful to the system or network. An IDS may passively monitor traffic and alert in the presence of some attack condition. Firewalls actively drop network packets that violate some network policy. Our classification method attempts to identify activities indicative of intrusion or misuse *after* such an event has occurred. Working in concert with a priori mechanisms, we can attempt to determine at any moment in time whether there is an impending attack or artifacts of a successful attack.

Figure 3 shows how our classification methods can be integrated into a network with an existing IDS. The organization uses servers to provide network services (internally, externally, or both) to some community of users. Our host-flow classification system monitors the output of these servers directly. The purpose is to determine if currently observed flows continue to behave as expected. If an attacker manages to take control of a particular service, he/she will need to interact with the server in such a way as to exactly match the previous behavior. A trojaned web server that behaves like a Telnet sever when communicating with a select group of host addresses would not match the expected host model, and thus be detected.

The network carries additional user traffic to servers that are external to the organization. This traffic is monitored with the aggregate model. Here, we classify the flow generally and compare this to the port label. Observation of traffic that resembles Telnet to some non-standard server port may be an indication of an installed backdoor. Traffic labeled as web traffic (with a server port of 80) that behaves more like Telnet traffic may indicate the presence of a proxy used to evade firewall rules. A peer-to-peer client operating at some user-defined port may be a violation of the network policy. In each of these cases, the aggregate classifier can indicate if a given flow behaves in a manner consistent with its port label. It may not be necessary to monitor every flow – the system could be configured to randomly select a flow and attempt to classify it. If this flow generally matches a flow that is unusual or undesirable for a port range, it can be identified and investigated.

Our method can operate on its own physical system, or it may be part of the IDS or firewall. This decision will be driven by the number of flows the system will be expected to monitor. Following the construction of the models, the system makes simple and rapid classifications.

6. Subverting Classification

Given the presence of a monitoring system described above, we examined ways in which an attacker could manipulate a session in order to affect classification of a server flow. We've previously seen one such example, albeit an innocuous one, in our discussion of the Water Cooler Effect. Here, the user suspends interaction thus causing variation in the arrival time of packets and hence a potentially large fluctuation in the mean inter-arrival time measured across the packet window for the server flow. An attacker can do the same thing. However, it is not clear that he/she would be able to cause a *particular* classification to be chosen. It is more likely that he/she will alter the observations as to cause some indeterminant class to be chosen. If a host classifier is being used, the deviation from the expected behavior would trigger an alarm.

Another method might involve the use of extraneous TCP flags in packets sent to the server. An example might be the use of the URG flag in HTTP packets. The distribution of TCP flags in the corresponding server flow may or may not be affected, based on the implementation of the server on that host. As with the effects of timing, we are investigating the sensitivity of classifiers to this manipulation in future work.

7. Related Work

Previous work in flow identification has employed a variety of techniques and feature sets. Dunigan and Ostrouchov used Principle Component Analysis (PCA) on two features (packet inter-arrival time and length) to create signatures for a variety of flow types [10]. Their reported classification accuracies are comparable to our method. However, their method requires off-line analysis. In contrast, our method performs classification in real time.

Tan and Collie used a modified neural network built on a single feature (total number of bytes transmitted) [35]. They confined their analysis to the classification of Telnet and FTP protocols. Their reported classification accuracies were generally lower than our method for these protocols.

Daniels [9] reports using a decision tree built with a single feature (first one hundred bytes of a packet) to classify flows. However, classification accuracy was found to be inadequate for practical use.

There are a number of commercial products that attempt to identify flow type [23–25, 34]. These are primarily used in the context of bandwidth allocation. For example, a network administrator creates a policy stating that web traffic must not exceed a certain percentage of total bandwidth and uses one of these products to selectively drop traffic when that policy is violated. Many details of the classification methods used by these products are not publicly available because they are proprietary. Thus, we are unable to compare our method to those used in these products. It is unclear whether any of these products can correctly classify flows in the presence of malicious activity (as described in Section 1). One company, Packeteer [25], reports that their product uses information "from all seven layers of the protocol stack" to create an application signature that is used to classify flows. As stated previously, such a system may or may not be appropriate in an environment where payload encryption is used or where there are concerns about user privacy.

We have also identified a component of the Snort IDS [29] that is used to classify server flows. However, this system relies on the port number and detection of the TCP 3-way handshake. As stated previously, in the presence of a proxy or compromised service, this system is unlikely to classify a flow correctly.

With respect to our feature set, the NATE (Network Analysis of Anomalous Traffic Events) [36] system is also based on TCP flags. NATE uses principle component analysis to identify that the TCP state flags can detect certain types of attacks. Our method differs in two respects. First, the NATE system attempts to model differences between normal traffic and attack traffic. They do not attempt to model differences in behavior between protocols. The second difference involves NATE's use of clustering to identify anomalies. This method must be done off-line, thus limiting the usefulness of the system in environments requiring near real-time detection. In contrast, once a decision tree has been created, our system can monitor packets in real-time.

8. Conclusions

We have presented a novel approach for defining a set of features to model operational behavior of server flow traffic. We demonstrated through the use of the C5.0 decision tree algorithm that our features can differentiate the behavior of server protocols with an accuracy of 82% to 100%. We illustrate empirically that aggregate models can classify an unseen server flow as belonging to a family of previously seen flows, and that host models can determine whether flows from a given server match the behavior of previously seen flows from that server. These classifiers can augment traditional intrusion detection systems to detect artifacts of successful attacks. Our techniques of classification are independent of packet labellings and are thus immune to techniques that modify port numbers to conceal activity.

The decision tree classifiers can be sensitive to fluctuations in the inter-arrival time of packets. This was exemplified in what we call the Water Cooler Effect. We plan to investigate how this sensitivity can be mitigated to increase the classification accuracy for certain protocols.

We intend to further examine the use and augmentation of our feature set to model additional types of server flows. Application of our technique will be expanded to other transport protocols, particularly those used by peer-to-peer file sharing systems. Other feature sets are in development to model UDP traffic, ICMP traffic, and selected routing protocols.

9. Acknowledgments

This research is supported by AFRL grant number F30602-02-2-0217.

References

[1] 1999 darpa intrusion detection evaluation data set. URL http://www.ll.mit.edu/IST/ideval/data/1999/1999_data_index.html.

[2] RFC 1700: Assigned Port Numbers. URL ftp://ftp.internic.net/rfc/rfc1700.txt.

[3] RFC 793 - Transmission Control protocol. URL ftp://ftp.internic.net/rfc/rfc0793.txt.

[4] Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):262–294, 2000. ISSN 1094-9224.

[5] S. Barbara and S. Jajodia. *Applications of Data Mining in Computer Security*. Kluwer Academic Publishers, 2002.

[6] K. Bennett and C. Campbell. Support Vector Machines: Hype or Hallelujah? *SIGKDD Explorations*, 2:1–13, 2000.

[7] J. Boxmeyer. ONCTec - List of Possible Trojan/Backdoor Port Activity. ONCTek, llc. URL http://www.onctek.com/trojanports.html.

[8] Cornell University Student Assembly Committee on Information and Technologies and ResNet. Cornell Internet Usage Statistics. URL http://www.cit.cornell.edu/computer/students/bandwidth/charts.html.

[9] Thomas E. Daniels. personal communication, September 2003.

[10] Tom Dunigan and George Ostrouchov. Flow Characterization for Intrusion Detection. Technical report, Oak Ridge National Laboratory, November 2000.

[11] Y. Freund. Boosting a Weak Learning Algorithm by Majority. *Information and Computation*, 121(2):256–285, 1995.

[12] Saul Hansell. E-mail's Backdoor Open to Spammers. *New York Times*, May 20 2003.

[13] iNetPrivacy Software Inc. Antifirewall. URL http://www.antifirewall.com/intro.htm.

[14] Internet Technical Resources. Traffic Statistics. URL http://www.cs.columbia.edu/~hgs/internet/traffic.html.

[15] G. H. Kim and G. Spafford. The Design and Implementation of Tripwire: A File System Integrity Checker. In *ACM Conference on Computer and Communications Security*, pages 18–29, 1994. URL citeseer.nj.nec.com/article/kim94design.html.

[16] T. Lane and C. E. Brodley. Temporal Sequence Learning and Data Reduction for Anomaly Detection. In *Proceedings of the Fifth ACM Conference on Computer and Communications Security*, pages 150–158. Association for Computing Machinery, Nov 1998.

[17] T. Lane and C. E. Brodley. Temporal Sequence Learning and Data Reduction for Anomaly Detection. *ACM Transactions on Computer Security*, 2(3):295–331, 1999.

[18] W. Lee and S. Stolfo. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security*, 3(4):227–261, November 2000.

[19] M. Mahoney and P. K Chan. PHAD: Packet Header Anomaly Detection for Indentifying Hostile Network Traffic. Technical Report CS-2001-4, Florida Tech, 2001.

[20] Evangelos P. Markatos. Tracing a Large-scale Peer to Peer System : an Hour in the Life of Gnutella. Technical Report 298, 2001. URL citeseer.nj.nec.com/markatos01tracing.html.

[21] T. Miller. Detecting Loadable Kernel Modules (LKM). URL http://www.incident-response.org/LKM.htm.

[22] N. Murilo and K. Steding-Jessen. chkrootkit: A Tool that Locally Checks for Signs of a Rootkit. URL http://www.chkrootkit.org/.

[23] NetScreen Technologies, Inc. NetScreen-5000 Series. URL http://www.netscreen.com/products/datasheets/ds_ns_5000.jsp.

[24] Captus Networks. Captus ips 4000 series. URL http://www.captusnetworks.com/.

[25] Packeteer, Inc. Packeteer PacketShaper. URL http://www.packeteer.com/resources/prod-sol/PSDS.pdf.

[26] P. A. Porras and P. G. Neumann. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *Proc. 20th NIST-NCSC National Information Systems Security Conference*, pages 353–365, 1997. URL http://citeseer.nj.nec.com/porras97emerald.html.

[27] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[28] Ross Quinlan. Data Mining Tools See5 and C5.0. URL http://www.rulequest.com/see5-info.html.

[29] M. Roesch and C. Green. Snort - The Open Source Network Intrusion Detection System. URL http://www.snort.org/.

[30] Robert E. Schapire. A Brief Introduction to Boosting. In *IJCAI*, pages 1401–1406, 1999. URL citeseer.nj.nec.com/schapire99brief.html.

[31] Sharman Networks Ltd. . Kazaa Media. URL http://www.kazaa.com/us/.

[32] E. Skoudis. *Counter Hack: A Step-by-Step Guide to Computer Attacks and Effective Defenses*. Prentice Hall, 2002.

[33] J. St. Sauver. Percentage of Total Internet2 Traffic Consisting of Kazaa/Morpheus/FastTrack - University of Oregon. In *Collaborative Computing in Higher Education: Peer-to-Peer and Beyond Workshop*. URL http://darkwing.uoregon.edu/~joe/kazaa.html.

[34] Stampede Technologies, Inc. Turbogold enterprise edition. URL http://www.stampede.com/productsclienttoserverfeaturesTraffic.html.

[35] K. M. C. Tan and B. S. Collie. Detection and Classification of TCP/IP Network Services. In *Proceedings of the Thirteenth Annual Computer Security Applications Conference*, pages 99–107, San Diego, California, December 1997.

[36] Carol Taylor and Jim Alves-Foss. NATE: Network Analysis of Anomalous Traffic Events, a low-cost approach. In *Proceedings of the 2001 Workshop on New Security Paradigms*, pages 89–96. ACM Press, 2001. ISBN 1-58113-457-6.