面向网络靶场的自适应仿真流量生成技术

余　涛

院　（系）：计算机科学与技术学院　　专　　业：信息安全

学　　号：1180300829　　指导教师：詹东阳

2022 年 6 月

哈爾濱工業大學

毕业设计（论文）

题　　目　面向网络靶场的自适应应仿真流量生成技术

专　　业　信息安全

学　　号　1180300829

学　　生　余涛

指 导 教 师　詹东阳

答 辩 日 期　2022 年 6 月 9 日

摘　要

随着互联网和计算机技术的高速发展，网络安全事件发生日趋频繁。为了应对愈发严重的网络安全威胁，许多国家和大型的国际组织都开始了网络靶场的建设，仿真流量生成作为验证靶场网络设备的功能和网络环境的重要方式，众多软件厂商提供了成熟的流量生成方案。但这些方案对于网络流量仿真模拟的逼真度方面存在欠缺，并且无法在短时间内生成大量的仿真流量。

本课题提出了一种面向网络靶场的自适应仿真流量生成技术，能够精准构建网络靶场环境下的仿真流量，用于网络靶场中的各种网络行为测试。系统抓取了业务网络中原始数据流并建立对应的协议交互模型，然后根据协议交互模型构建HTTP、SMTP、POP3、IMAP、SSH、TELNET的流量生成器。用户通过基于SpringBoot的接口API和基于JavaFX的客户端操作两种方式与系统交互，从而动态调用流量生成器，并且能够获取流量生成结果的可视化信息。为了提升开发效率和其他协议的后续扩展，系统提取了开发过程的共有步骤，提供了一种自动化构建协议交互模型代码的工具。

本课题实现了一个试验性的仿真流量生成系统，并且在该系统上开展了功能性测试和性能测试。测试结果表明系统能够满足用户自定义构建各种协议仿真流量，并且能够高速并发产生仿真流量。系统也能够根据原始数据流自动构建相关流量生成工具代码。这说明仿真流量生成系统能够满足设计需求。

关键词：网络靶场；流量生成器；协议交互模型；自动化构建

Abstract

With the rapid development of Internet and computer technology, network security incidents occur more and more frequently. In order to deal with the increasingly serious network security threats, many countries and large international organizations have started the construction of network shooting range. Simulation traffic generation is an important way to verify the function of network equipment and network environment in the range. Many software manufacturers provide mature traffic generation schemes. However, these schemes are deficient in the fidelity of network traffic simulation and cannot generate a large amount of simulated traffic in a short time.

This paper proposes a system-level implementation scheme of adaptive simulation flow generation technology for network shooting range, which can accurately construct simulation flow in network shooting range environment and be used for various network behavior testing in network shooting range. The system captures the original data flow in the service network and establishes the corresponding protocol interaction model. Then, according to the protocol interaction model, the flow generators of HTTP, SMTP, POP3, IMAP, SSH and TELNET are constructed. The user interacts with the system through the SpringBoot-based API and JavaFX-based client operation, so as to dynamically invoke the flow generator and obtain the visual information of the flow generation results. In order to improve the development efficiency and the subsequent extension of other protocols, the system extracts the common steps of the development process and provides a tool to build protocol interaction model code automatically.

This paper implements an experimental simulation flow generation system and carries out functional and performance test on the system. The results of the test show that the system can meet the requirements of user-defined simulation traffic and generate simulation traffic concurrently at high speed. The system can also automatically build the relevant traffic generation tool code from the raw data stream. This shows that the simulation flow generation system can meet the design requirements.

Keywords: Network Shooting Range, Flow Generator, Protocol Interaction Model, Automated Build

目　录

第 1 章 绪　论

1.1 课题背景及研究的目的和意义

伴随着互联网和计算机技术的高速发展，各种信息以数据包的形式在计算机网络中发生着传输和交互，而由于网络的公开性和易破解性导致整个网络系统中存在着各式各样的网络漏洞，因而网络中每天都发生着大量的网络攻击。根据国家互联网应急中心发布的《2020 年我国互联网网络安全态势综述》[1]，2020 年发生了超过 10 万件网络安全事件，并且 CNVD 在本年新收录了 20704 个漏洞，这些漏洞可能造成非常严重的网络威胁。因此在网络攻防逐渐走向常态化的今天，网络安全事件也日趋增多。

为了应对愈发复杂和愈发严重的网络安全威胁，很多国家和大型的国际组织都先行开始了网络靶场的建设[2]，网络靶场基于虚拟化技术，通过对现实网络的架构、设备、工作业务的运行状态及环境进行仿真模拟和复现，逐渐成长为用于网络安全新技术验证、网络武器装备测试、网络攻防行为对抗模拟和网络安全风险评估和高校网络空间安全课程教学的重要支撑平台。

在网络靶场结构体系中，一个非常重要的组成部分就是仿真流量生成模块。现阶段众多科研机构和软件厂商在流量仿真生成方面取得了很多成果，提供了多种功能强大的流量生成工具，比如 Netperf、iPerf 等流量生成器、商用仿真软件 OPNET 以及科研用仿真软件 NS2 及 NS3 等，这些网络流量生成技术在目前的网络靶场中设备的功能测试和流量仿真中已经发挥了重要的作用。但是以上技术的主要目的是为了验证靶场网络设备的功能和网络环境的测试，侧重点在高速传输、大流量和高带宽[3]。已有的流量生成技术存在以下不足，第一就是对于网络流量仿真模拟的逼真度方面存在欠缺，无法在网络靶场中完成流量的精准构建，第二就是生成仿真流量的速度较慢，无法在短时间内生成大量的仿真流量。

为了解决以上问题，本文提出了一种面向网络靶场的自适应仿真流量生成技术，目的是构建网络靶场环境下逼真的仿真流量，从而模拟出一个真实的网络环境来进行网络靶场测试。通过抓取业务网络中原始流量，分别提取对应的数据流，并且按数据流建立对应的协议交互模型，协议包括 HTTP、SMTP、POP3、IMAP 等多种应用层协议。建立基于这些协议交互模型自适应的网络流量生成器，并且能够根据真实的业务需求调用相关 API 来动态调整不同种类流量的生成比例。

1.2 研究现状

目前的网络行为模拟技术主要通过流量产生和流量回放来实现。宏观上来看，可以通过流量生成方法和模拟的粒度两个维度来对流量生成技术进行分类。

1.流量生成方法的维度

（1）基于模型：这种流量生成方法使用网络流量中的数学模型，通过监测真实的网络使用环境，获取相应的网络流量特征，然后设置模型参数进而生成对应的网络流量。常用的模型有泊松模型、马尔科夫模型、M/G/∞排队模型[4]等，这种流量生成方法由于参数可选择性多，可以满足用户自定义的需求。但这些模型很多需要满足相关的强假设，且都需要较理想的网络环境，受网络环境影响较大。

（2）基于跟踪：该流量生成方法使用先捕获后回放的方式进行网络流量重现，通过 Wireshark[5]、TCPDump[6]等流量捕获工具捕获网络中已经存在的真实流量，然后使用 TCPivo[7]等回放工具将前面捕获的流量进行复现。这种流量生成方法实质上是对于一个已经存在的网络流量交互环境进行复现，能够真实的还原一个流量行为。

（3）基于预测：这种流量生成方法基于统计学的规律，分析前一阶段的网络流量特性，利用相关的预测算法来预测生成新的网络流量。Katris 等人[8]通过对各种预测算法的比较得出了非线性算法对于网络流量的预测更加准确的结论。Cortez 等人[9]通过对互联网服务提供商提供的流量分别使用 Holt-Winters、ARIMA 和人工神经网络方法进行了五分钟、一小时和一天的流量预测。实验结果显示，在时长为五分钟和一小时时，人工神经网络方法的流量预测结果最准确，而时长为一天时，Holt-Winters 方法的流量预测最准确。

总的来说，基于模型的流量生成方法的模型都需要满足一定的强假设，所以普通的流量模型并不能准确地描述真实网络流量。基于跟踪的流量生成方法虽然复现了网络流量的真实特征，但是只能针对单机到单机的流量还原。基于预测的流量生成方法虽然在理论上能够比较真实的预测网络流量，但却与统计模型的选择有关，不能预测实际的模拟效果。

2.模拟的粒度

（1）包模拟：Swing[10]是一种自动化的流量生成工具，具有自动化获取网络中各主体之间的分布，并生成与原始流量相似的包流量。Clegg 等人[11]设计了一种流量模型来进行包级流量模拟的方法，该方法使用了马氏调制方法来产生 ON/OFF 序列，进而生成原始流量的仿真流量。

（2）流模拟：Harpoon[12]是一种部署在网络各节点的流级流量模拟器，其通

过客户端与服务器之间的点到点连接产生大量的网络流量。Dainotti 等人[13]对流量包的大小和包到达的间隔时间建立了隐马尔科夫模型，并且使用了 MEA 来得到模型需要的相关参数。

（3）应用模拟：Garg 等人[14]使用了一个独立的网络模拟环境，在该环境中通过对用户进行行为统计，对用户行为进行了建模，调用自动化工具来执行用户的网络行为，模拟用户使用相关的应用程序，从而间接模拟了用户的操作产生了网络流量。Klemm 等人[15]通过 BMAP 对用户的 IP 流量建模，这种建模的困难在于模型的参数估计，在实际操作中经常使用 EM 算法进行参数估计。

总的来说，包模拟和流模拟是通过构建包的长度和包的到达间隔时间来在网络的某个流量出口进行流量模拟，不能完成在网络中所有节点的流量模拟，应用模拟相当于模拟了用户对于应用程序的真实，这样的方式完美还原了用户的网络行为。

综合来看，以上所有的流量生成技术大部分都是基于流量捕获和流量回放的方式来生成仿真流量。这种方式能够准确地还原原始流量，但却不能精准地构建自己想要的流量，并且在流量生成的速度上有所欠缺，无法在短时间内生成大量的流量。因此，本文提出了一种仿真流量生成技术，在保证流量仿真度的同时，完成仿真流量的精准构建和高性能生成。

1.3 课题的主要研究内容

在一个虚拟的网络靶场环境下，为了诱骗攻击者进行攻击和进行一些网络工具的测试，需要生成足够逼真的背景流量。本课题设计并实现了面向网络靶场的自适应仿真流量生成技术，与传统的流量生成方式相比，做到了精准构建各种流量和高速生成各种流量，即构建了一个精准的高性能仿真流量生成系统，具体研究内容分为以下几个方面。

（1）设计仿真流量生成系统。本课题设计一个可以大量生成各种仿真流量的服务器系统，对系统进行需求分析、架构设计、代码目录设计、调用方式设计、细化模块设计。需求分析工作需要明确课题的需求，并对需求功能进行子模块化，用于后续系统架构的设计。架构设计工作需要明确并细化各模块的功能，设计各模块的优先级，并确立系统的基本工作过程。代码目录设计工作需要根据各框架的基本目录要求，采用规范化的命名方式，设计组织明确、结构清晰的代码目录空间，用于后续代码的编写。调用方式设计需要确定系统与用户的交互方法，设计用户调用各模块的方式和各模块之间的调用方式。细化模块设计需要根据需求

确立系统需要的各个模块，并完成相应的代码编写工作，需要根据各种语言的规范，编写可读性高、复用性强、冗杂度低的高质量代码。

（2）实现仿真流量生成系统。根据仿真流量生成系统的设计，实现一个在开发机上运行的仿真流量生成系统，主要包括应用层协议流量的捕获和优化、流量生成器的设计和实现、仿真流量生成接口的设计与实现、可视化生成流量结果、用户客户端的设计和实现、自动化构建协议交互模型工具的开发。应用层协议流量的捕获和优化会获取各种应用层协议优化后的 Pcap 文件。流量生成器的设计和实现会根据优化后的 Pcap 文件完成 HTTP、SMTP、POP3、IMAP、SSH、TELNET 这些应用层协议的仿真流量生成模块。仿真流量生成接口的设计与实现会给用户提供各种 API 接口，这些接口用来大量产生各种应用层协议的仿真流量。可视化生成流量结果能够为用户提供图表可视化展示生成仿真流量的结果。用户客户端的设计和实现为用户提供了一个简洁美观、操作性强的客户端界面，让用户调用 API 接口更加便捷。自动化构建协议交互模型工具的开发能够为开发者提供一种便捷的工具，该工具能够简化开发仿真流量模块繁琐的手动捕包分析过程，使用自动化的方式自动生成开发仿真流量模块的代码。

（3）开展功能测试和性能测试。完成系统代码编写后，依据系统需求在仿真流量生成系统上对系统的功能、性能进行详细的测试，主要包括各种流量仿真模块的测试、可视化测试、大流量生成速率测试等。

1.4 论文的组织架构

论文主要分为四部分。第 2 章将介绍本课题在系统设计和实现中所依赖的各种工具和技术，包括用于流量捕获分析的软件 Wireshark、用于构建模拟数据包的 Scapy[16]、用于实现后端接口和数据处理的 SpingBoot[17] + Mybatis[18]框架、用于可视化的 JfreeChart[19]、用于绘制客户端 UI 界面的 JavaFX[20]和用于包解析的 Dpkt；第 3 章将介绍本课题的系统设计，分析了系统的功能需求和性能需求后，提出系统的架构，给出了系统的代码目录和各模块之间的交互方式；第 4 章将介绍仿真流量生成系统的实现细节，首先介绍了系统的开发环境，然后分别对架构图中各模块的具体实现过程进行阐述；第 5 章将介绍系统的测试结果，对仿真流量系统进行了全面的功能测试和性能测试，涵盖了系统的全部设计需求，完成了系统的验证。

第 2 章 相关技术概述

2.1 引言

在本章中主要介绍本项目中用到的一些关键技术。使用 Wireshark 工具用作应用层协议流量的捕获和优化。Scapy 用作数据包的处理和构建，进而构建流量生成器。SpringBoot 框架作为仿真流量生成接口的实现，并且在接口中使用 Java 线程池 ThreadPoolExcutor 调用流量生成器脚本。MySQL 用作生成流量结果的数据存储，Mybatis 框架用作生成流量结果的数据处理，JfreeChart 用作生成流量结果的可视化。JavaFX 作为用户客户端的实现，并将 JavaFX 嵌入到 SpringBoot 框架中。在自动化构建协议交互模型工具的开发过程中，使用 Dpkt 进行数据包的分层处理。

2.2 Wireshark

在项目的前期准备工作中，需要捕获在开发机的真实网络环境中所产生的 HTTP、SMTP、POP3、IMAP、SSH、TELNET 这些应用层协议的流量。开发机会执行能够产生对应协议流量的网络行为，然后使用 Wireshark 作为捕包分析工具，捕获对应协议的数据流并进行优化后保存为 Pcap 文件，用于后续流量生成器的构建。

Wireshark 是一款专业的网络数据包处理软件，能够捕获原始网络数据，并显示数据包的各种信息。其作为一款开源软件，可以完美运行在 Windows 和 MacOS 上。Wireshark 可以用来检查网络问题，抓包分析自己测试的软件，并且用来进行 Socket 编程的调试。为了安全考虑，Wireshark 只能查看数据包，而不能修改数据包的内容，或者发送数据包。所以在本项目中，Wireshark 只能作为各种应用层协议捕包和优化数据流的软件。

Wireshark 为用户提供了多种功能，可以设置各种包条件过滤的显示过滤器(Display Filter)，显示捕获到的封包的编号、时间戳、源 IP、目标 IP、协议类型、长度、以及封包信息等的封包列表(Packet List Pane)，显示封包中的物理层的数据帧概况、数据链路层以太网帧头部信息、互联网层 IP 包头部信息、传输层的数据段头部信息、应用层的信息的封包详细信息(Packet Details Pane)。除此之外，Wireshark 还能将封包有选择性的以 Pcap、Pcapng 等各种格式保存。

2.3 Scapy

项目中仿真流量生成模块需要细粒度的模拟原始流量包发送过程，为了精准构建各种仿真流量生成模块的每一个数据包，项目引入了 Scapy 库。

Scapy 是一个轻量化的 Python 包处理程序，能够允许用户捕获、分析并伪造网络数据包。换言之，Scapy 是一个功能强大的交互式数据包处理程序。它能够解码和伪造各种网络协议的数据包，因此，Scapy 可以轻松处理大多数经典任务，如扫描、追踪路由、探查、单元测试或网络攻击。由于以上特性，Scapy 可以取代 hping、arpspoof、arp-sk、arping、p0f、 Nmap、tcpdump、tshark 等网络流量处理工具中的某些部分。

在本项目中，主要应用了 Scapy 来伪造各种网络协议的数据包，通过对 Scapy 伪造包的方法进行顶层封装，能够生成用户自定义的数据包。

2.4 SpringBoot + Mybatis 框架

本项目后端接口 API 基于目前 Java 社区非常先进和流行的 SpringBoot 框架实现，与数据库 MySQL 的交互使用了数据持久层框架 Mybatis。

SpringBoot 是一个基于 Spring 框架的 MVC 应用快速开发框架，相比于传统的 SSH(Struts-Spring-Hibernate)框架有很多优势。Spring 是 Java 社区一个轻量级开源框架，为 Web 服务器的开发提供了全套的解决方案。SpringBoot 依赖于 Spring，完美继承了 Spring 的控制反转和面向切面编程的特点，简化了传统基于 Spring 的 Web 服务器开发流程，可以看作传统 Spring 框架“约定大于配置的”思想的完美时间。SpringBoot 具有以下特性：快速搭建项目；无需繁琐的 XML 的配置；开箱即用，可直接以 json 形式或 yml 形式简化配置文件；内嵌 Tomcat 或 Jetty 等 Servlet 容器；Start 自动依赖和版本控制；使用 Spring Boot Actuator 组件提供了应用的系统监察，可以查看应用配置的详细信息；基于 Maven 或 Gradle 插件，可以创建可执行的 JARs 和 WARs。

Mybatis 是一款开源的 orm 类型的数据持久化框架，与传统的 jdbc 连接数据库操作不同，其将注册驱动、建立连接、获取 SQL 执行对象、释放连接等操作进行了自动化配置，开发人员只需要通过简单的配置就可以自动完成上述功能。Mybatis 实现了程序逻辑代码和 SQL 语句分离，开发人员只需要关注 SQL 语句的编写而不用过多的关注数据库连接问题。Mybatis 支持自定义 SQL、存储过程以及高级映射，可以通过映射 xml 文件实现 SQL 语句的编写。

由于 SpringBoot 易于配置的特性，直接在 Maven 配置文件中引入 Mabatis 框

架的依赖 jar 包，非常容易的就实现了在 SpringBoot 中嵌入 Mybatis 框架。进而完成项目的接口和持久层功能实现。

2.5 JfreeChart

为了给用户直观的展示流量生成情况，本项目选择使用 JfreeChart 生成仿真流量生成结果的柱状图和饼状图，进行结果的可视化。直接在 Maven 里引入 JfreeChart 的依赖 jar 包即可。

JFreeChart 是 Java 平台上的一个优质的开放图标构建库。使用 Java 语言编写，为 app、服务器和前端等设计，是一个相当不错的 Java 图形解决方案，能够解决大部分图形方面的需求。通过使用 JfreeChart，开发人员可以非常容易的用专业质量图表可视化应用程序的数据结果。JFreeChart 可生成柱状图、饼状图等多种图表，可以以各种格式图片输出，并且可以与可移植文档文件和电子表格关联。由于 API 处理简单、容易上手、免费的特性，JFreeChart 成为了很多开发人员 Java 可视化的第一选择。

2.6 JavaFX

鉴于客户端操作更加直观的特性，本项目除了提供了接口 API 供用户调用各模块之外，还为用户提供了一个客户端进行仿真流量的生成。客户端的编写使用了 JavaFX 技术。由于项目使用了 SpringBoot 框架，而 JavaFX 有针对 SpringBoot 的特殊实现形式，所以可以直接在 SpringBoot 中嵌入 JavaFX 技术，将客户端相关代码文件和后端代码文件进行合并。

JavaFX 是 Java 平台用于构建富互联网应用程序的库。使用 JavaFX 开发的应用程序可以在各种设备上运行，如台式计算机、手机、电脑、平板电脑等。JavaFX 技术具有良好的发展远景，包括可以直接调用 Java API 的能力。由于 JavaFX 是一种声明性的、静态类型的脚本语言，其也具有代码结构化、可重用性和高封装性的特征，如包、类直接继承和单独发布组件，这样就使得使用 JavaFX 技术创建和操作大型项目成为可能。

JavaFX 具有以下特性：是使用 Java 语言编写的、可以用于 JVM 执行的语言；使用 FXML 作为一个类似声明式标记语言 HTML，唯一目的是定义用户界面；可单独或集成到 IDE 中，用户可访问拖放设计界面，用于开发 FXML。可以使用 Swing Node 类嵌入 Swing 内容；提供类似样式 CSS；提供了一组丰富的 API 来开发 GUI 应用程序。以上特性使得使用 JavaFX 开发客户端界面变得简洁直观，并且界面代

码和逻辑代码能够分开，减少代码冗杂性。

2.7 Dpkt

项目的自动化构建协议交互模型工具的开发过程中，需要对原始 Pcap 文件进行每个包的分层拆分，Python 的 Dpkt 包提供了完美的解决方案。

Dpkt 是一个用于快速、简单数据包解析的 Python 模块，定义了基本 TCP/IP 协议。Dpkt 可以解析任意一个网络数据包，可以将数据包的各个组件的内容按照顺序依次解析出来，从下往上进行解析，每一层的 data 字段为上一层数据包的内容，依次为 ETHERNET 层，IP 层、TCP 层、RAW 层，通过获取每一层数据包对应字段的值，即可完成数据包内容的获取和分析。

2.8 本章小结

本章主要对开发面向网络靶场的自适应仿真流量生成技术中用到的关键工具和技术进行了简要的介绍，包括 Wireshark、Scapy、SpringBoot + Mybatis 框架、JfreeChart、JavaFX、Dpkt 等。这些工具和技术对本系统的设计和实现起到了基础性作用，具体应用和实现将在之后的章节进行更加详细的阐述。

第 3 章 系统设计

3.1 引言

本章全面分析了系统的需求，从系统的功能需求和功能需求两个方面进行总结。在需求的基础上阐述了系统的设计思路和技术创新，并给出了系统的架构设计和调用方式设计，从宏观上分析该系统的设计科学性和可行性。

3.2 需求分析

3.2.1 系统功能需求

面向网络靶场的自适应仿真流量生成技术需要构建一个这样的系统。该系统能够精准构建网络靶场环境下逼真的仿真流量，模拟出一个真实的网络环境来进行网络靶场测试。具体需求如下。

（1）流量仿真接口。本系统需要为网络靶场用户提供 HTTP、SMTP、POP3、IMAP、SSH、TELNET 协议的仿真流量生成接口 API，用户可以通过自定义协议类型、源 IP 地址、目的 IP 地址、产生流量数目来精准构建仿真流量。

（2）结果可视化。本系统需要为用户提供可视化展示仿真流量的生成结果。可视化的方式包括柱状图和饼状图，内容为生成的各种协议仿真流量数目。

（3）客户端界面。本系统需要为用户提供一个美观易操作的客户端 UI 界面，增强了系统的操作性和易交互性。

（4）代码构建自动化。本系统需要为仿真流量生成器开发者提供一种自动化构建协议交互模型的工具，用于后续其他协议仿真流量生成器的扩展。

3.2.2 系统性能需求

面向网络靶场的自适应仿真流量生成技术需要满足大量流量并发生成的性能需求。仿真流量生成接口需要提供一个产生仿真流量数目的参数，用户在调用接口时，可以设置该参数，并且能够在短时间内高并发的产生大量仿真流量。

3.3 设计思路

针对以上系统需求，本文对于系统有以下设计思路。

（1）首先需要捕获足够真实的网络流量，因此开发过程需要模拟用户真实的网络环境，从中提取出各种协议的原始数据流。数据流提取完成后，需要考虑选用相应的包模拟工具，完整的模拟出原始流量的交互过程，将这个过程生成仿真流量生成的模板，所有协议的流量生成工具的构建都可以用该过程实现。并且在这个实现过程中，需要充分考虑到系统的性能需求。

（2）流量生成工具构建完成后，就需要考虑到用户与系统的交互。针对开发者和普通用户，分别提供不同的系统调用方法，然后使用可视化工具展示流量的生成结果。

（3）完成系统的基本功能后，需要考虑到后续系统的协议扩展。将开发过程中开发者所进行的重复操作进行总结，编写相应的算法完成自动化构建协议交互模型代码的功能，以后对于任意应用层协议，开发者都能快速构建新的协议流量生成工具。

3.4 技术创新

（1）协议交互模型的构建。面向网络靶场的自适应仿真流量生成技术针对各种应用层协议，对其真实网络环境中产生的数据流都建立了协议交互模型，并且去除了失败的交互过程。所有协议的仿真流量都是根据该协议的模型构建生成。与传统的流量生成方式对比，本文中的仿真流量生成技术能够最大化的保证原始流量的特征，做到生成流量的足够仿真。

（2）协议自动机的设计。考虑到超大带宽的情况，以 1000Mb 的带宽为例，此时某个端的包解析速度可能远远大于端到端包传输速度。这种情况下为了保证仿真流量工具构建的包仍然能够按时发送，选择对所以应用层协议都建立有限状态自动机(Finite State Machine)，以下简称 FSM。仿真流量生成器在设计好的时间完成发包操作后，FSM 进行相应的状态转换。

（3）仿真流量并发生成。考虑到系统对流量生成速率的要求，系统使用了线程池并发调用仿真流量生成模块。在多个流量生成请求发送到系统后，系统将为所有请求创建一个线程，将其放入线程池中并发执行，从而实现多条仿真流量的并发生成。

（4）自动化构建。系统实现了很强的扩展性，为开发者提供了一种自动分析 Pcap 文件并构建协议交互模型代码的工具，开发者可以针对新的应用层协议，快速完成仿真流量生成器的开发。

3.5 系统架构

面向网络靶场的自适应仿真流量生成技术从整体来看一共包括五个模块，按照各模块完成顺序依次是流量获取模块、流量仿真模块、数据展示模块、用户模块和自动化模块。系统架构图如图 3-1。

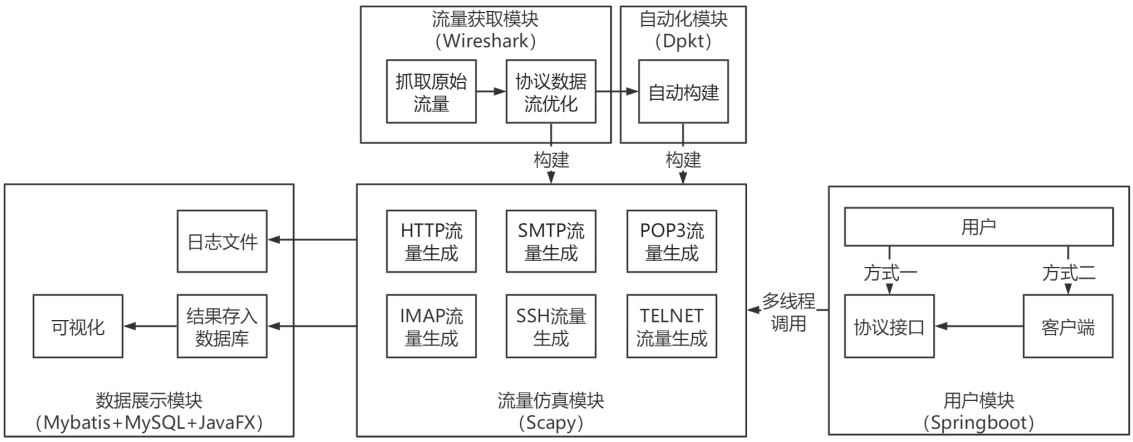

图 3-1 系统架构图

流量获取模块承担系统原始流量获取的功能。在开发机上捕获用户能够产生各种协议的网络流量的行为，使用 Wireshark 进行抓包获取原始的数据流，并用 Wireshark 的 Filter 过滤器进行手动包过滤，去掉一下丢失的包和错误的包，将优化后的数据流保存为 Pcap 文件，用于后续各协议仿真流量生成工具的开发。

流量仿真模块包含了 HTTP、SMTP、POP3、IMAP、SSH、TELNET 这些应用层协议的仿真流量生成工具。

用户模块为用户提供了两种调用流量仿真模块的方式。第一种方式是通过 SpringBoot 配置各种协议接口并发调用各协议的流量仿真模块，第二种方式是为用户提供一个 UI 客户端，通过客户端的输入调用相应的协议接口。

数据展示模块包括两部分。第一部分是在日志里存储调用流量仿真接口的记录，第二部分是在 MySQL 数据库里面储存各种协议流量产生的总数量，用柱状图和饼状图可视化展示生成流量的结果。

自动化模块设计了一种自动构建代码的方法，这种方法能够自动分析 Pcap 文件，分析里面的数据流的各字段，然后自动编写相应的流量交互代码，开发者以后扩展新的协议流量仿真模块可以通过此模块自动化的完成。

3.6 调用方式

3.6.1 系统与用户的交互

对于网络靶场中的用户来说，系统提供了两种方式来使用面向网络靶场的自适应仿真流量生成技术，从而满足开发者和使用者不同群体的需求。

第一种是通过接口 API 进行系统调用，主要面向开发者和部分使用者。系统一共给用户提供了 13 个可以外部访问的接口，用户通过 http 请求并携带相关的参数即可直接访问这些接口。接口一共包括 6 个仿真流量生成接口如表 3-1，4 个数据库操作接口如表 3-2，2 个可视化接口如表 3-3，1 个自动构建代码接口如表 3-4。

表 3-1 仿真流量生成接口表

接口名称	调用方法	参数	功能
/http	GET	sourceIp,targetIp,num	生成 num 条 http 仿真流量
/smtp	GET	sourceIp,targetIp,num	生成 num 条 smtp 仿真流量
/pop3	GET	sourceIp,targetIp,num	生成 num 条 pop3 仿真流量
/imap	GET	sourceIp,targetIp,num	生成 num 条 imap 仿真流量
/ssh	GET	sourceIp,targetIp,num	生成 num 条 ssh 仿真流量
/telnet	GET	sourceIp,targetIp,num	生成 num 条 telnet 仿真流量

表 3-2 数据库操作接口表

接口名称	调用方法	参数	功能
/flow-generation-num	PUT	--	将 flow_generation 表的所有 num 置零
/flow-generations	GET	--	查询所有 flow_generation
/flow-generation/{id}	GET	id	根据 id 查询 flow_generation 某条记录的 num
/flow-generation-num/{id}	PUT	num,id	根据 id 增加 flow_generation 某条记录的 num,增加的量为参数 num

表 3-3 可视化接口表

接口名称	调用方法	参数	功能
/visualization-bar	GET	--	可视化流量生成结果柱状图
/visualization-pie	GET	--	可视化流量生成结果饼状图

表 3-4 自动构建代码接口表

接口名称	调用方法	参数	功能
/automata	GET	pcapName、automataName	根据原始 Pcap 文件 pcapName 自动编写仿真流量生成代码，代码储存在 automataName 中

第二种是通过 UI 客户端直接进行操作，主要面向系统使用者。在客户端中选择协议类型，输入需要构建的仿真流量的参数，仿真流量生成后，可以选择用柱状图或饼状图可视化展示流量生成结果。由于 UI 客户端主要面向普通使用者用户，因而部分接口 API 的功能不会在 UI 客户端中实现。UI 客户端界面设计如图 3-3。

图 3-2 UI客户端界面

3.6.2 接口 API 与 Python 文件的交互

系统中的 6 个仿真流量生成接口 API 为了实现根据用户输入生成自定义的仿真流量，需要调用已经编写好的 Python 仿真流量生成工具脚本。并且为了满足短时间内生成大量仿真流量的性能需求，系统设计了一种接口 API 和 Python 文件交互的逻辑。设计了一种 CallPython 线程类，该线程类会创建一个线程，这个线程会指定一个 Python 文件，给其传递相应的参数。所有的仿真流量生成接口 API 都会根据参数创建对应数量的 CallPython 线程类，然后将所有线程类放入一个全局的线程池 ThreadPoolExecutor 中执行。

以/HTTP 接口为例，用户在调用该接口后，接口会创建用户指定的数量的线程类CallPython，并且给其赋好参数，指定需要调用的HTTP仿真流量生成的Python文件。然后将这些线程类放入线程池中运行，这样就保证了所有的产生仿真流量的过程是并发执行的，从而实现在短时间内产生大量的仿真流量。接口 API 调用 Python 文件的逻辑如图 3-4。

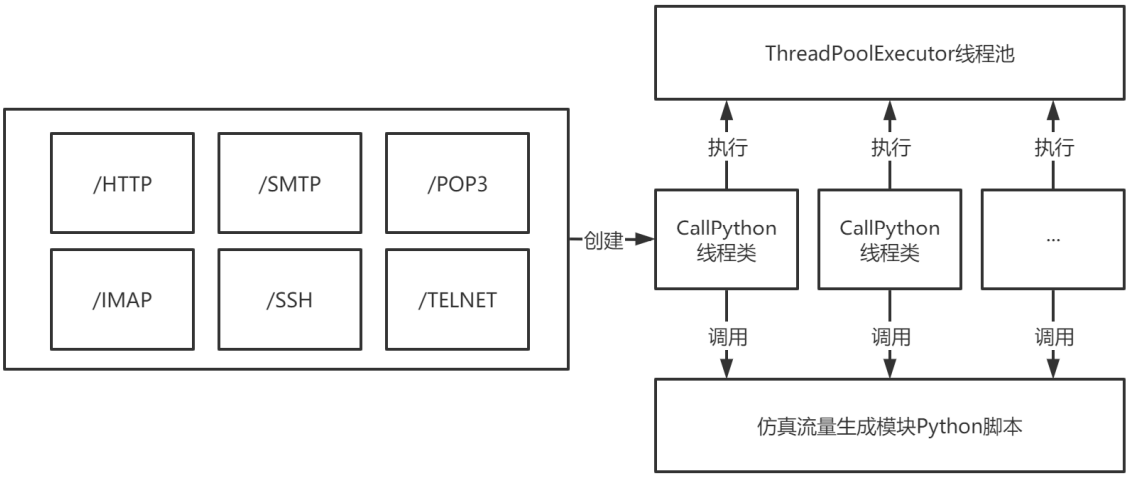

图 3-3 接口API调用Python文件逻辑

3.6.3 接口 API 与 MySQL 的交互

系统的 4 个数据库操作接口需要操作数据库 MySQL，本系统选择使用 Mybatis 作为数据库操作框架。根据 MVC 架构的基本格式，选择使用常规的 controller、service、mapper 层作为目录结构，通过 SpringBoot 的依赖注入和这三层自上而下的调用，然后在 mapper 层指定其所绑定的 Mybatis 的 xml 数据库操作配置文件，实现了每个数据库操作接口都能操作数据库 MySQL 进行对应的改变，进而完成接口 API 和 MySQL 的交互。

3.7 本章小结

本章详细分析了面向网络靶场的自适应仿真流量生成技术的需求，归纳总结了流量仿真接口、结果可视化、客户端界面、代码构建自动化四个方面的功能需求，并分析了大量流量生成性能需求。介绍了该仿真流量生成技术的技术创新和流量生成的设计思路。同时从宏观上介绍了系统架构、调用方式，并且在调用方式中详细设计了系统与用户的交互方式、接口 API 和 Python 文件的交互方式、接口 API 和 MySQL 的交互方式。为后续系统的具体实现起了基础性作用。

第 4 章 系统实现

4.1 引言

本章介绍了仿真流量生成系统开发的具体实现。该系统严格按照系统设计，完成了系统架构中所有部分的实现。主要包括系统的开发环境、各功能模块的具体实现。功能模块包括应用层协议流量的捕获和优化、流量生成器的设计和实现、仿真流量生成接口的设计与实现、可视化生成流量结果、用户客户端的设计和实现、自动化构建协议交互模型工具的开发。

4.2 系统开发环境

系统环境：16G 内存，100GB 硬盘容量，AMD R7 4800H 处理器

软件环境：Windows 11 专业版，MySQL，JDK1.8，IntelliJ IDEA，JetBrains Pycharm，Wireshark，DBeaver，Postman，Maven，Git，JavaFX SceneBuilder

开发语言：Java，Python，XML

开发框架：SpringBoot，Mybatis，JavaFX

4.3 功能模块实现

4.3.1 应用层协议流量的捕获和优化

此模块通过在开发机上进行能够产生不同网络协议流量的网络行为，使用 Wireshark 捕获了 HTTP、SMTP、POP3、IMAP、SSH、TELNET 协议原始流量并优化，然后将优化后数据流以 Pcap 文件保存，用于后续流量生成器的设计和实现。下面将详细介绍每种应用层协议流量的捕获和优化过程。

4.3.1.1 HTTP 流量

对于一些常用的网站如 https://www.baidu.com，使用浏览器访问后然后抓包分析，会发现这类网站使用了 HTTP 的长连接，浏览器和服务器一直处于 keep-alive 状态，这样就导致了需要很长时间浏览器才会与服务器断开连接，不利用分析保存 HTTP 的原始流量。所以需要选择一个浏览器访问之后不会与服务器建立 HTTP 长连接的网站进行捕包。对比很多网站之后，本系统最终决定选择哈工大本科生

院 http://hituc.hit.edu.cn 进行访问，过滤出浏览器访问时产生的 HTTP 数据流，选取其中一条数据流进行保存如图 4-1。可以看出该 HTTP 交互过程在 Tcp 连接和断开之间，经过了 HTTP 协议的请求和响应交互，并且没有丢包和重传现象。

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.164.80	219.217.226.25	TCP	66	63315 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.004961	219.217.226.25	172.20.164.80	TCP	66	80 → 63315 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1360 SACK_PERM=1 WS=128
3	0.005048	172.20.164.80	219.217.226.25	TCP	54	63315 → 80 [ACK] Seq=1 Ack=1 Win=131840 Len=0
4	0.005295	172.20.164.80	219.217.226.25	HTTP	749	GET / HTTP/1.1
5	0.008678	219.217.226.25	172.20.164.80	TCP	56	80 → 63315 [ACK] Seq=1 Ack=696 Win=30592 Len=0
6	0.017544	219.217.226.25	172.20.164.80	TCP	1414	80 → 63315 [ACK] Seq=1 Ack=696 Win=30592 Len=1360 [TCP segment of a reassembled PDU]
7	0.017544	219.217.226.25	172.20.164.80	TCP	1414	80 → 63315 [ACK] Seq=1361 Ack=696 Win=30592 Len=1360 [TCP segment of a reassembled PDU]
8	0.017544	219.217.226.25	172.20.164.80	TCP	1414	80 → 63315 [ACK] Seq=2721 Ack=696 Win=30592 Len=1360 [TCP segment of a reassembled PDU]
9	0.017544	219.217.226.25	172.20.164.80	TCP	1414	80 → 63315 [ACK] Seq=4081 Ack=696 Win=30592 Len=1360 [TCP segment of a reassembled PDU]
10	0.017544	219.217.226.25	172.20.164.80	HTTP	1125	HTTP/1.1 200 OK (text/html)
11	0.017544	219.217.226.25	172.20.164.80	TCP	56	80 → 63315 [FIN, ACK] Seq=6512 Ack=696 Win=30592 Len=0
12	0.017644	172.20.164.80	219.217.226.25	TCP	54	63315 → 80 [ACK] Seq=696 Ack=6513 Win=131840 Len=0
13	0.018647	172.20.164.80	219.217.226.25	TCP	54	63315 → 80 [FIN, ACK] Seq=696 Ack=6513 Win=131840 Len=0
14	0.022936	219.217.226.25	172.20.164.80	TCP	56	80 → 63315 [ACK] Seq=6513 Ack=697 Win=30592 Len=0

图 4-1 HTTP数据流

4.3.1.2 SMTP 流量

在一封电子邮件从发件方发送到收件方收到的过程中，首先会在发件方到 SMTP 服务器之间产生 SMTP 协议的交互过程。基于以上特点，为了获取原始的 SMTP 网络流量，选择使用两个邮箱进行邮件的发送和接受。使用 Foxmail 登录两个自己的邮箱，然后用 1063695334@qq.com 发送一封邮件给 3268130899@qq.com，在这个过程中，使用 Wireshark 捕获产生的 SMTP 数据流，过滤后保存如图 4-2。可以看出该 SMTP 交互过程在 Tcp 连接和断开之间，经过了 SMTP 协议需要的各种请求和响应交互，只在最后的 Tcp 四次挥手断开时出现了包不对应的情况，这个需要在后续仿真流量生成器的实现中进行优化。

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.84.115	120.241.186.196	TCP	66	53177 → 25 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.060829	120.241.186.196	172.20.84.115	TCP	66	25 → 53177 [SYN, ACK] Seq=0 Ack=1 Win=14400 Len=0 MSS=1360 SACK_PERM=1 WS=128
3	0.060903	172.20.84.115	120.241.186.196	TCP	54	53177 → 25 [ACK] Seq=1 Ack=1 Win=131840 Len=0
4	0.144824	120.241.186.196	172.20.84.115	SMTP	117	S: 220 newxmesmtplogicsvrszc6.qq.com XMail Esmtp QQ Mail Server.
5	0.150503	172.20.84.115	120.241.186.196	SMTP	76	C: EHLO LAPTOP-OUDFRK43
6	0.210826	120.241.186.196	172.20.84.115	TCP	56	25 → 53177 [ACK] Seq=64 Ack=23 Win=14464 Len=0
7	0.212838	120.241.186.196	172.20.84.115	SMTP	223	S: 250-newxmesmtplogicsvrszc6.qq.com | PIPELINING | SIZE 73400320 | STARTTLS | AUTH LOGI
8	0.213170	172.20.84.115	120.241.186.196	SMTP	401	C: AUTH XOAUTH2 dXNlcj0xMDYzNjk1MzM0QHFxLmNvbQFhdXRoPUJlYXJlciBBQVRKSlk0RUFBQUJBQUFBQUFB
9	0.310774	120.241.186.196	172.20.84.115	TCP	56	25 → 53177 [ACK] Seq=233 Ack=370 Win=15488 Len=0
10	0.462834	120.241.186.196	172.20.84.115	SMTP	74	S: 235 2.7.0 Accepted
11	0.466172	172.20.84.115	120.241.186.196	SMTP	96	C: MAIL FROM: <1063695334@qq.com> SIZE=1042
12	0.524845	120.241.186.196	172.20.84.115	TCP	56	25 → 53177 [ACK] Seq=253 Ack=412 Win=15488 Len=0
13	0.568790	120.241.186.196	172.20.84.115	SMTP	62	S: 250 OK
14	0.569033	172.20.84.115	120.241.186.196	SMTP	84	C: RCPT TO: <3268130899@qq.com>
15	0.628799	120.241.186.196	172.20.84.115	TCP	56	25 → 53177 [ACK] Seq=261 Ack=442 Win=15488 Len=0
16	0.716787	120.241.186.196	172.20.84.115	SMTP	62	S: 250 OK
17	0.717147	172.20.84.115	120.241.186.196	SMTP	60	C: DATA
18	0.776836	120.241.186.196	172.20.84.115	TCP	56	25 → 53177 [ACK] Seq=269 Ack=448 Win=15488 Len=0
19	0.784838	120.241.186.196	172.20.84.115	SMTP	92	S: 354 End data with <CR><LF>.<CR><LF>.
20	0.784991	172.20.84.115	120.241.186.196	SMTP	458	C: DATA fragment, 404 bytes
21	0.882842	120.241.186.196	172.20.84.115	TCP	56	25 → 53177 [ACK] Seq=307 Ack=852 Win=16640 Len=0
22	0.882900	172.20.84.115	120.241.186.196	SMTP/I…	697	from: "1063695334@qq.com" <1063695334@qq.com>, subject: theme, (text/plain) (text/html)
23	0.941415	120.241.186.196	172.20.84.115	TCP	56	25 → 53177 [ACK] Seq=307 Ack=1495 Win=17920 Len=0
24	1.294838	120.241.186.196	172.20.84.115	SMTP	74	S: 250 OK: queued as.
25	1.295560	172.20.84.115	120.241.186.196	SMTP	60	C: QUIT
26	1.352967	120.241.186.196	172.20.84.115	TCP	56	25 → 53177 [ACK] Seq=327 Ack=1501 Win=17920 Len=0
27	1.356837	120.241.186.196	172.20.84.115	SMTP	64	S: 221 Bye.
28	1.358860	120.241.186.196	172.20.84.115	TCP	56	25 → 53177 [FIN, ACK] Seq=337 Ack=1501 Win=17920 Len=0
29	1.358912	172.20.84.115	120.241.186.196	TCP	54	53177 → 25 [ACK] Seq=1501 Ack=338 Win=131584 Len=0
30	1.695293	172.20.84.115	120.241.186.196	TCP	54	53177 → 25 [RST, ACK] Seq=1501 Ack=338 Win=0 Len=0

图 4-2 SMTP数据流

4.3.1.3 POP3 流量

在一封电子邮件从发件方发送到收件方收到的过程中，经过 SMTP 交互之后，然后会在收件方到 POP3/IMAP 服务器之间产生 POP3/IMAP 协议的交互过程。在 Foxmail 配置收件服务器为 POP3，然后再使用 1063695334@qq.com 发送一封邮件给 3268130899@qq.com。在这个过程中，使用 Wireshark 捕获产生的 POP3 数据流，过滤后保存如图 4-3。可以看出该 POP3 交互过程在 Tcp 连接和断开之间，经过了 POP3 协议需要的各种请求和响应交互，只在最后的 Tcp 四次挥手断开时出现了包不对应的情况，这个需要在后续仿真流量生成器的实现中进行优化。

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.84.115	120.241.186.196	TCP	66	58999 → 110 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.061971	120.241.186.196	172.20.84.115	TCP	66	110 → 58999 [SYN, ACK] Seq=0 Ack=1 Win=14400 Len=0 MSS=1360 SACK_PERM=1 WS=128
3	0.062051	172.20.84.115	120.241.186.196	TCP	54	58999 → 110 [ACK] Seq=1 Ack=1 Win=131840 Len=0
4	0.123976	120.241.186.196	172.20.84.115	POP	108	S: +OK XMail POP3 Server v1.0 Service Ready(XMail v1.0)
5	0.124195	172.20.84.115	120.241.186.196	POP	78	C: USER 3268130899@qq.com
6	0.183978	120.241.186.196	172.20.84.115	TCP	56	110 → 58999 [ACK] Seq=55 Ack=25 Win=14464 Len=0
7	0.189933	120.241.186.196	172.20.84.115	POP	59	S: +OK
8	0.190165	172.20.84.115	120.241.186.196	POP	77	C: PASS bigyhlolsfzycjdb
9	0.290538	120.241.186.196	172.20.84.115	TCP	56	110 → 58999 [ACK] Seq=60 Ack=48 Win=14464 Len=0
10	0.796000	120.241.186.196	172.20.84.115	POP	59	S: +OK
11	0.796797	172.20.84.115	120.241.186.196	POP	60	C: STAT
12	0.857937	120.241.186.196	172.20.84.115	TCP	56	110 → 58999 [ACK] Seq=65 Ack=54 Win=14464 Len=0
13	0.895943	120.241.186.196	172.20.84.115	POP	67	S: +OK 6 26402
14	0.896113	172.20.84.115	120.241.186.196	POP	60	C: LIST
15	0.982000	120.241.186.196	172.20.84.115	POP	110	S: +OK
16	0.982355	172.20.84.115	120.241.186.196	POP	60	C: UIDL
17	1.064005	120.241.186.196	172.20.84.115	POP	266	S: +OK
18	1.067153	172.20.84.115	120.241.186.196	POP	62	C: RETR 6
19	1.164938	120.241.186.196	172.20.84.115	TCP	56	110 → 58999 [ACK] Seq=346 Ack=74 Win=14464 Len=0
20	1.194816	120.241.186.196	172.20.84.115	POP	1414	S: +OK 2395
21	1.195338	120.241.186.196	172.20.84.115	POP	126	S: DATA fragment, 72 bytes
22	1.195338	120.241.186.196	172.20.84.115	POP	1032	S: DATA fragment, 978 bytes
23	1.195406	172.20.84.115	120.241.186.196	TCP	54	58999 → 110 [ACK] Seq=74 Ack=2756 Win=131840 Len=0
24	1.217372	172.20.84.115	120.241.186.196	POP	60	C: QUIT
25	1.276789	120.241.186.196	172.20.84.115	TCP	56	110 → 58999 [ACK] Seq=2756 Ack=80 Win=14464 Len=0
26	1.342943	120.241.186.196	172.20.84.115	POP	63	S: +OK Bye
27	1.352964	120.241.186.196	172.20.84.115	TCP	56	110 → 58999 [FIN, ACK] Seq=2765 Ack=80 Win=14464 Len=0
28	1.353021	172.20.84.115	120.241.186.196	TCP	54	58999 → 110 [ACK] Seq=80 Ack=2766 Win=131840 Len=0
29	2.234830	172.20.84.115	120.241.186.196	TCP	54	58999 → 110 [FIN, ACK] Seq=80 Ack=2766 Win=131840 Len=0
30	2.295009	120.241.186.196	172.20.84.115	TCP	56	110 → 58999 [RST] Seq=2766 Win=0 Len=0

图 4-3 POP3数据流

4.3.1.4 IMAP 流量

获取 IMAP 原始流量的过程和上述类似，只是需要在 Foxmail 配置收件服务器为 IMAP，然后再使用 1063695334@qq.com 发送一封邮件给 3268130899@qq.com。在这个过程中，使用 Wireshark 捕获产生的 IMAP 数据流。分析之后发现该 POP3 交互过程，客户端一共使用了五个端口，第一个端口用于客户端和服务器的三次握手建立连接，然后紧接着有三个端口分别进行了完整的 IMAP 交互过程，最后又使用了一个端口用于四次挥手断开连接。为了实现只用一个端口的模拟，选择将三次握手和第一个 IMAP 交互过程结合保存如图 4-4。并且需要在后续仿真流量生成器的实现中补全四次挥手过程。

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	120.241.17.119	172.20.236.83	TCP	66	143 → 54120 [SYN, ACK] Seq=0 Ack=1 Win=14400 Len=0 MSS=1360 SACK_PERM=1 WS=128
2	0.000148	172.20.236.83	120.241.17.119	TCP	54	54120 → 143 [ACK] Seq=1 Ack=1 Win=515 Len=0
3	0.076167	120.241.17.119	172.20.236.83	IMAP	164	Response: * OK [CAPABILITY IMAP4 IMAP4rev1 ID AUTH=PLAIN AUTH=LOGIN AUTH=XOAUTH2 NAME
4	15.070233	172.20.236.83	120.241.25.80	IMAP	64	Request: C87 NOOP
5	15.134265	120.241.25.80	172.20.236.83	TCP	56	143 → 65012 [ACK] Seq=1 Ack=11 Win=197 Len=0
6	15.151924	120.241.25.80	172.20.236.83	IMAP	77	Response: C87 OK NOOP Completed
7	15.153809	172.20.236.83	120.241.25.80	IMAP	104	Request: C88 STATUS "INBOX" (MESSAGES RECENT UIDVALIDITY)
8	15.228545	120.241.25.80	172.20.236.83	IMAP	142	Response: C88 OK STATUS completed
9	15.231180	172.20.236.83	120.241.25.80	IMAP	64	Request: C89 NOOP
10	15.302991	120.241.25.80	172.20.236.83	IMAP	77	Response: C89 OK NOOP Completed
11	15.304931	172.20.236.83	120.241.25.80	IMAP	70	Request: C90 CAPABILITY
12	15.371927	120.241.25.80	172.20.236.83	IMAP	177	Response: C90 OK CAPABILITY Completed
13	15.373300	172.20.236.83	120.241.25.80	IMAP	209	Request: C91 ID ("name" "com.tencent.foxmail" "version" "7.2.23.116" "os" "windows" "
14	15.441954	120.241.25.80	172.20.236.83	IMAP	85	Response: C91 OK ID completed
15	15.442388	172.20.236.83	120.241.25.80	IMAP	74	Request: C92 SELECT "INBOX"
16	15.542293	120.241.25.80	172.20.236.83	TCP	56	143 → 65012 [ACK] Seq=289 Ack=262 Win=205 Len=0
17	15.578105	120.241.25.80	172.20.236.83	IMAP	358	Response: C92 OK [READ-WRITE] SELECT complete
18	15.579203	172.20.236.83	120.241.25.80	IMAP	75	Request: C93 FETCH 1:3 (UID)
19	15.643006	120.241.25.80	172.20.236.83	TCP	56	143 → 65012 [ACK] Seq=593 Ack=283 Win=205 Len=0
20	15.738944	120.241.25.80	172.20.236.83	IMAP	137	Response: C93 OK FETCH Completed
21	15.739594	172.20.236.83	120.241.25.80	IMAP	114	Request: C94 UID FETCH 19 (UID RFC822.SIZE FLAGS BODY.PEEK[HEADER])
22	15.802921	120.241.25.80	172.20.236.83	TCP	56	143 → 65012 [ACK] Seq=676 Ack=343 Win=205 Len=0
23	16.047062	120.241.25.80	172.20.236.83	TCP	1414	143 → 65012 [ACK] Seq=676 Ack=343 Win=205 Len=1360 [TCP segment of a reassembled PDU]
24	16.047279	120.241.25.80	172.20.236.83	IMAP/I…	510	from: "3268130899@qq.com" <3268130899@qq.com>, subject: ImapTheme,
25	16.047310	172.20.236.83	120.241.25.80	TCP	54	65012 → 143 [ACK] Seq=343 Ack=2492 Win=515 Len=0

图 4-4 IMAP数据流

4.3.1.5 SSH 流量

SSH 主要用于远程主机的加密登录，远程控制某主机，并且整个过程是完全加密的。为了产生 SSH 的网络流量，本系统选择在开发机上使用 SSH 远程连接自己租用的阿里云服务器，在开发机的命令行执行 ssh root@101.132.158.199，然后输入自己的密码远程登录阿里云服务器，登录成功后输入 exit 退出登录。在这个过程中使用 Wireshark 捕获产生的 SSH 数据流，过滤后保存，保存后的部分数据流如图 4-5。可以看出该 SSH 交互过程在 Tcp 连接和断开之间，经过了 SSH 协议需要的各种请求和响应交互，只在最后的 Tcp 四次挥手断开时出现了包不对应的情况，这个需要在后续仿真流量生成器的实现中进行优化。

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.236.83	101.132.158.199	TCP	66	62724 → 22 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.049546	101.132.158.199	172.20.236.83	TCP	66	22 → 62724 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1360 SACK_PERM=1 WS=128
3	0.049700	172.20.236.83	101.132.158.199	TCP	54	62724 → 22 [ACK] Seq=1 Ack=1 Win=131840 Len=0
4	0.055167	172.20.236.83	101.132.158.199	SSHv2	87	Client: Protocol (SSH-2.0-OpenSSH_for_Windows_8.1)
5	0.103584	101.132.158.199	172.20.236.83	TCP	56	22 → 62724 [ACK] Seq=1 Ack=34 Win=29312 Len=0
6	0.105563	101.132.158.199	172.20.236.83	SSHv2	75	Server: Protocol (SSH-2.0-OpenSSH_8.0)
7	0.108867	172.20.236.83	101.132.158.199	TCP	1414	62724 → 22 [ACK] Seq=34 Ack=22 Win=131840 Len=1360 [TCP segment of a reassembled PDU]
8	0.108867	172.20.236.83	101.132.158.199	SSHv2	86	Client: Key Exchange Init
9	0.157527	101.132.158.199	172.20.236.83	SSHv2	1102	Server: Key Exchange Init
10	0.159509	172.20.236.83	101.132.158.199	SSHv2	102	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
11	0.211623	101.132.158.199	172.20.236.83	SSHv2	506	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys, Encrypted packet
12	0.217472	172.20.236.83	101.132.158.199	SSHv2	70	Client: New Keys
13	0.307137	101.132.158.199	172.20.236.83	TCP	56	22 → 62724 [ACK] Seq=1522 Ack=1490 Win=32000 Len=0
14	0.307307	172.20.236.83	101.132.158.199	SSHv2	98	Client: Encrypted packet (len=44)
15	0.355598	101.132.158.199	172.20.236.83	TCP	56	22 → 62724 [ACK] Seq=1522 Ack=1534 Win=32000 Len=0
16	0.355851	101.132.158.199	172.20.236.83	SSHv2	98	Server: Encrypted packet (len=44)
17	0.356118	172.20.236.83	101.132.158.199	SSHv2	114	Client: Encrypted packet (len=60)
18	0.411585	101.132.158.199	172.20.236.83	SSHv2	138	Server: Encrypted packet (len=84)
19	0.411924	172.20.236.83	101.132.158.199	SSHv2	554	Client: Encrypted packet (len=500)
20	0.465777	101.132.158.199	172.20.236.83	SSHv2	138	Server: Encrypted packet (len=84)
21	0.509339	172.20.236.83	101.132.158.199	TCP	54	62724 → 22 [ACK] Seq=2094 Ack=1734 Win=131584 Len=0
22	5.451979	172.20.236.83	101.132.158.199	SSHv2	202	Client: Encrypted packet (len=148)
23	5.512572	101.132.158.199	172.20.236.83	SSHv2	82	Server: Encrypted packet (len=28)
24	5.514938	172.20.236.83	101.132.158.199	SSHv2	166	Client: Encrypted packet (len=112)
25	5.598031	101.132.158.199	172.20.236.83	SSHv2	682	Server: Encrypted packet (len=628)

图 4-5 部分SSH数据流

4.3.1.6 TELNET 流量

TELNET 与 SSH 类似，也是用于远程主机的加密登录，远程控制某主机，但是整个过程是完全的明文传输，所以 TELNET 是不安全的。为了产生 TELNET 的网络流量，本系统选择在开发机上使用 TELNET 远程连接自己租用的阿里云服务器。由于 TELNET 的不安全性，在几乎所有的主机上 TELNET 服务器并没有进行安装。所以首先需要在阿里云服务器上面安装 TELNET，在阿里云服务器上安装 TELNET 服务依赖的 xinetd 服务后，运行 TELNET 服务，并且打开阿里云服务器的 23 端口。服务器的 TELNET 服务配置完成后，在开发机的命令行执行 telnet 101.132.158.199，然后输入用户名和密码登录阿里云服务器，在这个过程中使用 Wireshark 捕获产生的 SSH 数据流，过滤后保存，保存后的部分数据流如图 4-6。可以看出该 TELNET 交互过程在 Tcp 连接和断开之间，经过了 TELNET 协议需要的各种请求和响应交互，并且没有丢包和重传现象。

	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.20.51.137	101.132.158.199	TCP	66	59442 → 23 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.362407	101.132.158.199	172.20.51.137	TCP	66	23 → 59442 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1360 SACK_PERM=1 WS=128
3	0.362535	172.20.51.137	101.132.158.199	TCP	54	59442 → 23 [ACK] Seq=1 Ack=1 Win=131840 Len=0
4	1.373436	101.132.158.199	172.20.51.137	TELNET	66	Telnet Data ...
5	1.373704	172.20.51.137	101.132.158.199	TELNET	60	Telnet Data ...
6	1.415440	101.132.158.199	172.20.51.137	TCP	56	23 → 59442 [ACK] Seq=13 Ack=7 Win=29312 Len=0
7	1.415440	101.132.158.199	172.20.51.137	TELNET	57	Telnet Data ...
8	1.415562	172.20.51.137	101.132.158.199	TELNET	63	Telnet Data ...
9	1.457772	101.132.158.199	172.20.51.137	TELNET	66	Telnet Data ...
10	1.457916	172.20.51.137	101.132.158.199	TELNET	63	Telnet Data ...
11	1.539576	101.132.158.199	172.20.51.137	TCP	56	23 → 59442 [ACK] Seq=28 Ack=25 Win=29312 Len=0
12	1.539634	172.20.51.137	101.132.158.199	TELNET	70	Telnet Data ...
13	1.581440	101.132.158.199	172.20.51.137	TCP	56	23 → 59442 [ACK] Seq=28 Ack=41 Win=29312 Len=0
14	1.581440	101.132.158.199	172.20.51.137	TELNET	66	Telnet Data ...
15	1.581633	172.20.51.137	101.132.158.199	TELNET	57	Telnet Data ...
16	1.664927	101.132.158.199	172.20.51.137	TCP	56	23 → 59442 [ACK] Seq=40 Ack=44 Win=29312 Len=0
17	1.664993	172.20.51.137	101.132.158.199	TELNET	63	Telnet Data ...
18	1.707447	101.132.158.199	172.20.51.137	TCP	56	23 → 59442 [ACK] Seq=40 Ack=53 Win=29312 Len=0
19	1.707657	101.132.158.199	172.20.51.137	TELNET	112	Telnet Data ...
20	1.707898	172.20.51.137	101.132.158.199	TELNET	57	Telnet Data ...
21	1.749682	101.132.158.199	172.20.51.137	TELNET	85	Telnet Data ...
22	1.749751	172.20.51.137	101.132.158.199	TELNET	57	Telnet Data ...
23	1.831594	101.132.158.199	172.20.51.137	TCP	56	23 → 59442 [ACK] Seq=129 Ack=59 Win=29312 Len=0
24	4.655063	172.20.51.137	101.132.158.199	TELNET	55	Telnet Data ...
25	4.705774	101.132.158.199	172.20.51.137	TCP	56	23 → 59442 [ACK] Seq=129 Ack=60 Win=29312 Len=0
26	4.706436	101.132.158.199	172.20.51.137	TELNET	56	Telnet Data ...
27	4.746646	172.20.51.137	101.132.158.199	TCP	54	59442 → 23 [ACK] Seq=60 Ack=130 Win=131584 Len=0

图 4-6 部分TELNET数据流

4.3.2 流量生成器的设计和实现

此模块主要实现了 HTTP、SMTP、POP3、IMAP、SSH、TELNET 这些协议的仿真流量生成工具，通过对 4.3.1 节得到的各种优化后的应用层协议数据流进行分析，逐包构建和原始数据流交互过程相同的仿真流量交互过程，这个过程能够实现用户的自定义构建。并且建立各种协议的状态自动机，在和原始数据流量相同的发包时间点进行仿真流量的发包，然后完成协议自动机的状态转换。下面将详细介绍仿真流量生成工具的构建过程。

4.3.2.1 封装发包方法

Scapy 可以根据用户需求自定义构建一个数据包并发送，在本系统中，通过对 HTTP、SMTP、POP3、IMAP、SSH、TELNET 这些应用层协议的交互过程进行分析，可以得出一共有 6 种标志位的 TCP 数据包被发送，分别为建立连接的 SYN 包、建立连接的回复 SYNACK 包、不带数据的 ACK 包、带数据的 ACK 包、断开连接的 FIN 包、带数据的 PSH 包。

Scapy 构建数据包的基本格式为 PKT = IP(可选字段)/TCP(可选字段)/(可选 DATA)，分别对应 OSI 五层网络模型的上三层，即网络层、传输层、应用层。因此根据系统需求，可以选择封装上述提到的 6 种 Scapy 发送数据包的方法，以封装带数据的 ACK 包为例，封装好的发包方法如下所示。

```
def send_tcp_ack_data(source_ip, target_ip, source_port, target_port, source_seq, target_seq, data):
    pkt = IP(src=source_ip, dst=target_ip) / TCP(dport=target_port, sport=source_port,
seq=source_seq, ack=target_seq, flags='A') / data
    send(pkt, verbose=False)
```

以上方法可以构建一个自定义 IP 地址、端口号和数据的 ACK 数据包，该方法内构建了 IP 层数据，其可选字段为源 IP 地址和目的 IP 地址，并且构建了 TCP 层数据，其可选字段为源端口号、目的端口号、seq 序号、ack 序号和标志位，同时加入了应用层数据。其他的 5 种封装好的发包方法原理类似，所有封装好的发包方法如表 4-1。

表 4-1 封装的发包方法

方法名称	参数	Flags	功能
send_tcp_syn	source_ip, target_ip, source_port, target_port, source_seq, target_seq	2	发送 SYN 包
send_tcp_synack	source_ip, target_ip, source_port, target_port, source_seq, target_seq	18	发送 SYNACK 包
send_tcp_ack	source_ip, target_ip, source_port, target_port, source_seq, target_seq	16	发送 ACK 包
send_tcp_ack_data	source_ip, target_ip, source_port, target_port, source_seq, target_seq, data	16	发送 ACK 数据包

表 4-1（续表）

方法名称	参数	Flags	功能
send_tcp_finack	source_ip, target_ip, source_port, target_port, source_seq, target_seq	17	发送 FINACK 包
send_tcp_pshack_data	source_ip, target_ip, source_port, target_port, source_seq, target_seq, data	24	发送 PSHACK 数据包

在后续仿真流量生成工具的开发中，都需要使用以上封装好的发包方法，对所有协议逐包进行原始数据流交互过程的模拟。

4.3.2.2 交互过程模拟

选取 4.3.1 节得到的所有 Pcap 文件，对 Pcap 文件里面的所有包，都需要逐包进行分析处理，对任意一个数据包，都需要按顺序依次执行表 4-2 过程中的所有操作，从而完成协议交互过程的模拟。

表 4-2 原始包处理步骤

步骤	动作
1.选取发包方法	（1）查看 TCP 段的 Flags，选取表 4-1 中与 Flags 对应的方法
2.方法参数构建	（1）构建 source_ip, target_ip, source_port, target_port：查看 TCP 段的 Destination Port，如果 Destination Port 为各种协议服务器的端口号，则构建从客户端到服务器的参数，反之则构建从服务器到客户端的参数。 （2）构建 source_seq, target_seq：查看 TCP 段的 Sequence Number 和 Acknowledgment Number，直接在参数里使用其 Number 即可。 （3）构建 data：查看 TCP 段中是否有数据，查看数据对应的 ASCII 码。若 ASCII 码为明文，则直接将参数设置为该 ASCII 码即可。若为密文，则选择查看原始数据十六进制流，使用 binascii.unhexlify()方法转化后设置为参数即可。

Pcap 文件中的原始数据包按照顺序经过表 4-2 中所有的步骤处理后，就能完成所有协议交互过程模拟的代码的构建。模拟交互过程与原始数据流相同。

4.3.2.3 协议自动机

针对所有已经建立好的协议交互过程，建立其协议对应的 FSM，并且在各种协议的自动机建立完成后，选择一个起始时间点，记录原始流量每次发包的时间点，对仿真流量工具在同样的时间点进行发包操作，并在包发送完成后进行 FSM 的状态转换。各种应用层协议的 FSM 如表 4-3 和表 4-4。

表 4-3 HTTP、SMTP、POP3协议FSM

HTTP	SMTP	POP3
0:BEGIN	0:BEGIN	0:BEGIN
1:SHAKEHAND1_SYN_SENT	1:SHAKEHAND1_SYN_SENT	1:SHAKEHAND1_SYN_SENT
2:SHAKEHAND2_SYN_ACK_SENT	2:SHAKEHAND2_SYN_ACK_SENT	2:SHAKEHAND2_SYN_ACK_SENT
3:SHAKEHAND3_ACK_SENT	3:SHAKEHAND3_ACK_SENT	3:SHAKEHAND3_ACK_SENT
4:HTTP_REQUEST_SENT	4:SERVER_220_SENT	4:SERVER_READY_SENT
5:HTTP_RESPONSE_SENT	5:CLIENT_EHLO_SENT	5:CLIENT_USER_SENT
6:SHAKEHAND1_FIN_SENT	6:SERVER_250_SENT1	6:SERVER_OK_SENT1
7:SHAKEHAND2_ACK_SENT	7:CLIENT_AUTH_SENT	7:CLIENT_PASS_SENT
8:SHAKEHAND3_FIN_SENT	8:SERVER_235_SENT	8:SERVER_OK_SENT2
9:SHAKEHAND4_ACK_SENT	9:CLIENT_MAIL_FROM_SENT	9:CLIENT_STAT_SENT
10:CLOSED	10:SERVER_250_SENT2	10.SERVER_OK_SENT3
--	11:CLIENT_RCPT_TO_SENT	11:CLIENT_LIST_SENT
--	12:SERVER_250_SENT3	12:SERVER_OK_SENT4
--	13:CLIENT_DATA_SENT	13:CLIENT_UIDL_SENT
--	14:SERVER_354_SENT	14:SERVER_OK_SENT5

表 4-3（续表）

HTTP	SMTP	POP3
--	15:CLIENT_MESSAGE_SENT	15:CLIENT_RETR_SENT
--	16:SERVER_250_SENT4	16:SERVER_DATA_SENT
--	17:CLIENT_QUIT_SENT	17:CLIENT_OK_SENT
--	18:SERVER_221_SENT	18:CLIENT_QUIT_SENT
--	19:SHAKEHAND1_FIN_SENT	19:SERVER_OK_SENT6
--	20:SHAKEHAND2_ACK_SENT	20:SHAKEHAND1_FIN_SENT
--	21:SHAKEHAND3_FIN_SENT	21:SHAKEHAND2_ACK_SENT
--	22:SHAKEHAND4_ACK_SENT	22:SHAKEHAND3_FIN_SENT
--	23:CLOSED	23:SHAKEHAND4_ACK_SENT
--	--	24:CLOSED

表 4-4 IMAP、SSH、TELNET协议FSM

IMAP	SSH	TELNET
0:BEGIN	0:BEGIN	0:BEGIN
1:SHAKEHAND1_SYN_SENT	1:SHAKEHAND1_SYN_SENT	1:SHAKEHAND1_SYN_SENT
2:SHAKEHAND2_SYN_ACK_SENT	2:SHAKEHAND2_SYN_ACK_SENT	2:SHAKEHAND2_SYN_ACK_SENT
3:SHAKEHAND3_ACK_SENT	3:SHAKEHAND3_ACK_SENT	3:SHAKEHAND3_ACK_SENT

表 4-4（续表）

IMAP	SSH	TELNET
4:SERVER_OK_SENT	4:CLIENT_PROTOCOL_SENT	4:SERVER_CLIENT_EXCHANGE_MESSAGE
5:CLIENT_NOOP_SENT	5:SERVER_PROTOCOL_SENT	5:SERVER_WAIT_INPUT_NAME
6:SERVER_OK_NOOP_SENT	6:CLIENT_KEY_EXCHANGE_INIT_SENT	6:CLIENT_INPUT_NAME_AND_SERVER_AUTH
7:CLIENT_STATUS_INBOX_SENT	7:SERVER_KEY_EXCHANGE_INIT_SENT	7:SERVER_WAIT_INPUT_PASSWORD
8:SERVER_OK_STATUS_SENT	8:CLIENT_DH_KEY_EXCHANGE_INIT_SENT	8:CLIENT_INPUT_PASSWORD
9:CLIENT_NOOP_SENT2	9:SERVER_DH_KEY_EXCHANGE_REPLY_NEW_KEYS_SENT	9:SERVER_LAST_LOGIN_AND_WELCOME
10.SERVER_OK_NOOP_SENT2	10:CLIENT_NEW_KEYS_SENT	10:SERVER_WAIT_COMMAND
11:CLIENT_CAPABILITY_SENT	11:CLIENT_SERVER_DATA_EXCHANGE	11:SHAKEHAND1_FIN_SENT
12:SERVER_OK_CAPABILITY_SENT	12:SHAKEHAND1_FIN_SENT	12:SHAKEHAND2_ACK_SENT
13:CLIENT_ID_SENT	13:SHAKEHAND2_ACK_SENT	13:SHAKEHAND3_FIN_SENT
14:SERVER_OK_ID_SENT	14:SHAKEHAND3_FIN_SENT	14:SHAKEHAND4_ACK_SENT
15:CLIENT_SELECT_INBOX_SENT	15:SHAKEHAND4_ACK_SENT	15:CLOSED
16:SERVER_OK_SELECT_SENT	16:CLOSED	--

表 4-4（续表）

IMAP	SSH	TELNET
17:CLIENT_FETCH_SENT	--	--
18:SERVER_OK_FETCH_SENT	--	--
19:CLIENT_UID_FETCH_SENT	--	--
20:SERVER_DATA_SENT	--	--
21:SHAKEHAND1_FIN_SENT	--	--
22:SHAKEHAND2_ACK_SENT	--	--
23:SHAKEHAND3_FIN_SENT	--	--
24:SHAKEHAND4_ACK_SENT	--	--
25:CLOSED	--	--

4.3.3 仿真流量生成接口的设计与实现

本模块基于 SpringBoot 框架，为网络靶场的用户提供了各种接口 API 用于生成各种协议的仿真流量，并可以根据用户需求控制各种流量生成的数目和比例。下面将详细介绍仿真流量生成接口的实现过程。

4.3.3.1 并发调用

在 4.3.2 节中使用了 Python 语言作为六种仿真流量生成工具的实现语言，而本模块使用了基于 Java 语言的 SpringBoot 框架作为服务器接口的实现，所以实现在 Java 文件中调用 Pyhon 文件的功能，并且在前面 3.2.2 节中有大量流量生成的系统性能需求，即需要让这些仿真流量生成接口能够在短时间内生成大量的仿真流量。基于以上要求，此模块使用了线程池来并发执行多个调用仿真流量生成工具的线程。

首先是调用仿真流量生成工具的线程类的开发。在项目的 utils 目录下创建一个线程类 CallPythonSimulate，该类需要 implements Runnable 接口，然后实现 run 方法，run 方法由于是无参方法，所以无法直接传递线程类需要的参数，所以定义了 CallPythonSimulate 的构造方法，构造方法中指定了 Python 文件名 agreement、构造的源 IP 地址 sourceIp、构造的目的 IP 地址 targetIp、构造的源端口号 sourcePort、构造的目的端口号 targetPort。构造方法如下。

```
public CallPythonSimulate(String agreement, String sourceIp, String targetIp, Integer sourcePort,
Integer targetPort) {
        this.agreement = agreement;
        this.sourceIp = sourceIp;
        this.targetIp = targetIp;
        this.sourcePort = sourcePort;
        this.targetPort = targetPort;
}
```

在 run()方法中根据上述参数，指定 Python 路径、被调用的 Python 文件路径、源 IP 地址、目的 IP 地址、源端口号和目的端口号后，调用仿真流量生成工具，并且在流量生成工具中读取这些参数生成对应的仿真流量。至此调用仿真流量生成工具的线程类开发完毕。

然后就是并发调用的开发。创建了一个 ThreadPoolExecutor 线程池作为全局的线程池，每次需要生成仿真流量时，系统都会创建一个 CallPythonSimulate 线程类，并根据接口的参数指定线程类构造方法里面的参数，将创建的线程类放入线程池中运行，这样便实现了流量的并发生成，满足了性能需求中的大量流量生成部分。

4.3.3.2 接口开发

表 3-1 设计了 6 中仿真流量生成接口，分别为/http、/smtp、/pop3、/imap、/ssh 和/telnet，选择使用 SpringBoot 的@GetMapping 注解作为这些接口的实现，以/http 接口为例，其接口的实现形式如下。

```
@GetMapping("/http")
public String httpSimulate(@RequestParam("sourceIp") String sourceIp,
@RequestParam("targetIp") String targetIp, @RequestParam("num") Integer num) {
   ……  // 多线程生成 HTTP 仿真流量
}
```

接口中只指定了源 IP 地址 sourceIp、目的 IP 地址 targetIp 和生成流量数目 num，而端口号的指定则不需要用户给出。系统设计的客户端端口号规则为“某起始值 + i + num”，其中 0 < i < num。据此可以得出仿真流量生成接口所生成的仿真流量的所有端口号如表 4-5。

表 4-5 仿真流量端口表

接口名称	客户端端口范围	服务器端口号
/http	5000~9999	80
/smtp	10000~14999	25
/pop3	15000~19999	110
/imap	20000~24999	143
/ssh	25000~29999	22
/telnet	30000~34999	23

经过以上步骤，便得到了 CallPythonSimulate 需要的所有参数，然后创建线程类并发在线程池中执行即可。

4.3.4 可视化生成流量结果

本模块用于仿真流量生成结果的可视化，在 MySQL 数据库中建立相关表储存用户产生的各种类型仿真流量的数量，使用 SpringBoot + Mybatis 框架完成对数据库的相关操作。通过 Mybatis 查询流量生成结果后，使用 JfreeChart 将结果以柱状图和饼状图的形式展示给用户，从而实现可视化生成流量结果的功能。下面将详细介绍可视化生成流量的实现过程。

4.3.4.1 流量结果数据表

为了能够实时更新产生的仿真流量的更新，在 MySQL 数据库中创建了一个记录流量生成结果的数据表，该表记录了协议 id、协议类型、生成的仿真流量数目。流量结果表 flow_generation 结构见表 4-6。

表 4-6 流量结果表

序号	名称	数据类型	非空	约束
1	id	int(32)	Yes	PK
2	agreement	varchar(100)	Yes	--
3	num	int(32)	Yes	--

4.3.4.2 表处理

使用 SpringBoot + Mybatis 框架建立控制层(Controller)、业务层(Service)、持久层(Mapper)三层结构，其调用逻辑如图 4-7，从而完成对上面建立的 flow_generation 的各种数据库操作。相关数据表操作已在表 3-2 中给出。

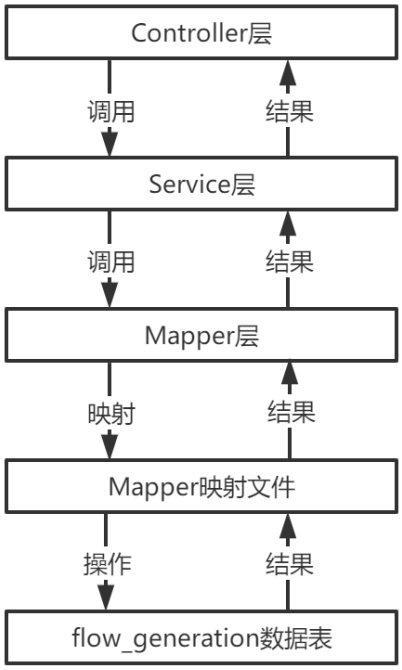

图 4-7 表处理逻辑图

通过 Controller、Service、Mapper 三层的层层调用，依赖于 SpringBoot 框架的@Autowired 注解自动注入下一层的单例对象，从而完成对 flow_generation 表的数据操作。并且在 4.3.3 中开发的仿真流量生成接口中增加以下功能，即每次生成仿真流量后，都对 flow_generation 表的 num 值进行更新。

4.3.4.3 流量结果可视化

在 Controller 层增加两个可视化接口，接口如表 3-3 所示，两个接口分别用于构建仿真流量生成结果的饼状图和柱状图。在 Controller 中依赖于自动注入的 flowGenerationService，查询 flow_generation 表中所有的信息储存为 List，然后将 List 传入已经构建好的可视化方法中，进而完成仿真流量生成结果的可视化。用户调用相关可视化接口即可生成美观、详细的图形流量生成结果。

4.3.5 用户客户端的设计和实现

本模块主要基于 JavaFX，为用户提供了一个美观直接的客户端界面。通过设计相应的 JavaFX 客户端组件，并在组件上执行相应的动作，然后调用相关接口 API 进行功能的实现。下面将详细介绍用户客户端的设计和实现过程。

创建一个 fxml 文件，使用 xml 语言在该文件中进行客户端界面各种组件的设计，分析系统需求后，一共设计了 14 个 JavaFX 组件，包括 6 个 Label 组件、1 个 JFXComboBox 组件、3 个 TextField 组件、3 个 JFXButton 组件和 1 个 TextArea 组件。Label 组件用于界面的一些文字展示，JFXComboBox 组件用于协议的选择，TextField 组件用于用户的输入，JFXButton 组件用户点击按钮执行动作，TextArea 组件用于展示动作执行后的结果。部分组件功能如表 4-7。

表 4-7 JavaFX部分组件表

组件类型	功能
JFXComboBox	提供一个协议类型选择的下拉栏，用于用户选择协议类型
TextField	用于用户输入源 IP 地址、目的 IP 地址和生成流量数目
JFXButton	用于用户执行生成流量、生成饼状图、生成柱状图的操作
TextArea	用于展示操作结果

然后创建一个 Controller 文件，与@Autowired 类似，JavaFX 在 SpringBoot 框架中也可以通过注解@FXML 注入各组件，对需要执行动作的 JavaFX 组件，调用相关 API 执行设定的动作即可。

4.3.6 自动化构建协议交互模型工具

本模块设计了一种算法，该算法能够自动分析 Pcap 包文件，自动化的构建协议交互过程的代码。在之前的流量生成器开发过程中，对于 Pcap 文件中的每一个数据包，都需要经过表 4.2 中所有的操作，这样才能完成一个数据包的发送代码构建，这个过程是繁琐费时且重复的。因此本系统提供了一种算法，这种算法总结了表 4.2 中的包构建过程，实现自动编写整个协议交互过程代码的功能，并且能够剔除掉一些发送出错的包，主要用于开发者后续对流量生成工具的扩展。

用于自动化的构建协议交互过程的如算法 1。

Algorithm 1 Automatic Construction of Protocol Interaction Model

Input：*pcap_path, protocol_map, server_port_list, method_list*

Output：Simulation traffic generation code *all_command*

```
function generate_automata (pcap_path, protocol_map, server_port_list, method_list)
    all_command ← ''
    pcap ← open(pcap_path)
    for pkt in pcap do
        command ← 'send_tcp_'
        get source_port, source_seq, target_seq, flags, data from pkt
        for protocol_key, protocol_value in protocol_map do
            if protocol_value = flags then
                command ← command + protocol_key   # 加上 protocol
                if len(data) ≠ 0 and protocol_key = 'ack'   # ack 包判断是否需要加上 data
                    command ← command + '_data'
                end if
            end if
        end for
        command ← command + '('
        if source_port in server_port_list then   # 加上 ip 和 port
            command ← command + 'target_ip, source_ip, target_port, source_port, '
        else
            command ← command + ' source_ip, target_ip, source_port, target_port, '
        end if
```

```
21.          command ← command + sources_seq + ',' + target_seq   # 加上 seq 和 ackseq
22.          if len(data) ≠ 0   # 加上 data
23.              command ← command + ',' + data + ')'
24.          else
25.              command ← command + ')'
26.          end if
27.          for method in method_list   # 优化：去掉原始 pcap 文件中失败的包
28.              if method in command
29.                  all_command ← all_command + command + '\n'
30.                  break
31.              end if
32.          end for
33.      end for
34.      return all_command
35.  end function
```

Algorithm 1 中需要用到的常量参数如下。

```
protocol_map = {'syn': 2, 'synack': 18, 'ack': 16, 'pshack_data': 24, 'finack': 17}
server_port_list = [80, 25, 110, 143, 22, 23]
method_list = ['send_tcp_syn', 'send_tcp_synack', 'send_tcp_ack', 'send_tcp_ack_data',
'send_tcp_finack', 'send_tcp_pshack_data']
```

在 Controller 层增加自动化构建协议交互模型的接口，接口如表 3-4 所示，开发者可以指定接口需要分析的源 Pcap 文件 pcapName，然后将生成的代码保存在 automataName 文件中。后续开发者只需要完成代码协议自动机的补充即可。

4.4 本章小结

本章分模块对面向网络靶场的自适应仿真流量生成技术的系统实现进行了详细阐述。按照系统的开发顺序，主要包括应用层协议流量的捕获和优化、流量生成器的设计和实现、仿真流量生成接口的设计与实现、可视化生成流量结果、用户客户端的设计和实现、自动化构建协议交互模型工具的开发，并结合表和伪代码进行了辅助说明。

第 5 章 系统测试

5.1 引言

本章完成对该系统的功能性测试和非功能性测试。软件测试是软件开发步骤中非常重要的一个部分，能够验证软件的正确性和完整性，只有通过了完备的软件测试步骤，才能发现系统中忽视的 bug 和问题，经过完备的软件测试后系统才能正常上线。本章分别从功能测试和性能测试两个方面介绍。

5.2 功能测试

系统的功能测试内容主要采用黑盒测试的方式进行，黑盒测试是在已知系统功能基础上，对系统实现的各种功能进行测试，在测试时不需要了解系统内部实现的具体逻辑，侧重于系统的外部功能表现。针对仿真流量生成系统，所有的系统功能都通过接口 API 或客户端 UI 提供给了用户，因此只需要完成接口和客户端的功能性测试即可。对于接口 API，使用 Postman 工具构建满足各接口参数的 http 请求进行测试，接口测试结果表如表 5-1。

表 5-1 接口测试结果表

接口	请求类型	测试用例	结果验证
/http	GET	http://localhost/http?sourceIp=101.132.158.80&targetIp=101.132.158.100&num=10	Wireshark 成功捕获到 10 条 HTTP 仿真流量
/smtp	GET	http://localhost/smtp?sourceIp=101.132.158.80&targetIp=101.132.158.100&num=10	Wireshark 成功捕获到 10 条 SMTP 仿真流量
/pop3	GET	http://localhost/pop3?sourceIp=101.132.158.80&targetIp=101.132.158.100&num=10	Wireshark 成功捕获到 10 条 POP3 仿真流量

表 5-1（续表）

接口	请求类型	测试用例	验证结果
/imap	GET	http://localhost/imap?sourceIp=101.132.158.80&targetIp=101.132.158.100&num=10	Wireshark 成功捕获到 10 条 IMAP 仿真流量
/ssh	GET	http://localhost/ssh?sourceIp=101.132.158.80&targetIp=101.132.158.100&num=10	Wireshark 成功捕获到 10 条 SSH 仿真流量
/telnet	GET	http://localhost/telnet?sourceIp=101.132.158.80&targetIp=101.132.158.100&num=10	Wireshark 成功捕获到 10 条 TELNET 仿真流量
/flow-generation-num	PUT	http://localhost/flow-generation-num	flow_generation 表的所有元组的 num 成功置 0
/flow-generations	GET	http://localhost/flow-generations	成功获取 flow_generation 表的所有元组
/flow-generation/{id}	GET	http://localhost/flow-generation/1	成功获取 flow_generation 表 id 为 1 的元组的 num
/flow-generation-num/{id}	PUT	http://localhost/flow-generation-num/1?num=10	成功给 flow_generation 表 id 为 1 的元组的 num 加 10
/visualization-bar	GET	http://localhost/visualization-bar	成功绘制流量生成结果柱状图
/visualization-pie	GET	http://localhost/visualization-pie	成功绘制流量生成结果饼状图
/automata	GET	http://localhost/automata?pcapName=http.pcap&automataName=httpAutomata.txt	成功根据“http.pcap”文件生成自动机代码

对于客户端 UI，模拟用户的操作界面的各种组件进行测试，客户端 UI 测试结果表如表 5-2。

表 5-2 客户端UI测试结果表

测试项目	测试目的	操作步骤	结果验证
用户无输入提示	用户输入验证	用户不选择协议类型、不输入 SOURCE_IP、TARGET_IP 或 NUM	弹出提示用户选择协议类型、输入相关内容的提示框
协议流量生成	验证各种协议的流量生成功能	用户选择协议类型，输入 SOURCE_IP、TARGET_IP 和 NUM	结果栏显示成功信息，Wireshark 成功捕获到相关流量
柱状图流量统计显示	验证柱状图可视化功能	用户点击“柱状图”按钮	结果栏显示结果信息，弹出柱状图
饼状图流量统计显示	验证饼状图可视化功能	用户点击“饼状图”按钮	结果栏显示结果信息，弹出饼状图

5.3 性能测试

在需求分析中提出了系统的性能需求，即系统需要在短时间内产生多条仿真流量，在开发机上进行性能测试，针对每种协议生成 300 条仿真流量，性能时间开销如表 5-3。

表 5-3 性能时间开销表

协议类型	流量数目	时间	速率
HTTP	300	22 秒	15 条/秒
SMTP	300	30 秒	10 条/秒
POP3	300	35 秒	9 条/秒
IMAP	300	32 秒	9 条/秒

表 5-3（续表）

协议类型	流量数目	时间	速率
SSH	300	39 秒	8 条/秒
TELNET	300	45 秒	7 条/秒

对于每种协议的流量生成模块，不断增加系统生成的仿真流量数目，测试每个模块能达到的最大发包速率。用流量回放工具 TcpReplay 对原始流量进行流量回放，测试 TcpReplay 的最大发包速率，与本系统的发包速率进行对比。对比测试如图 5-1。

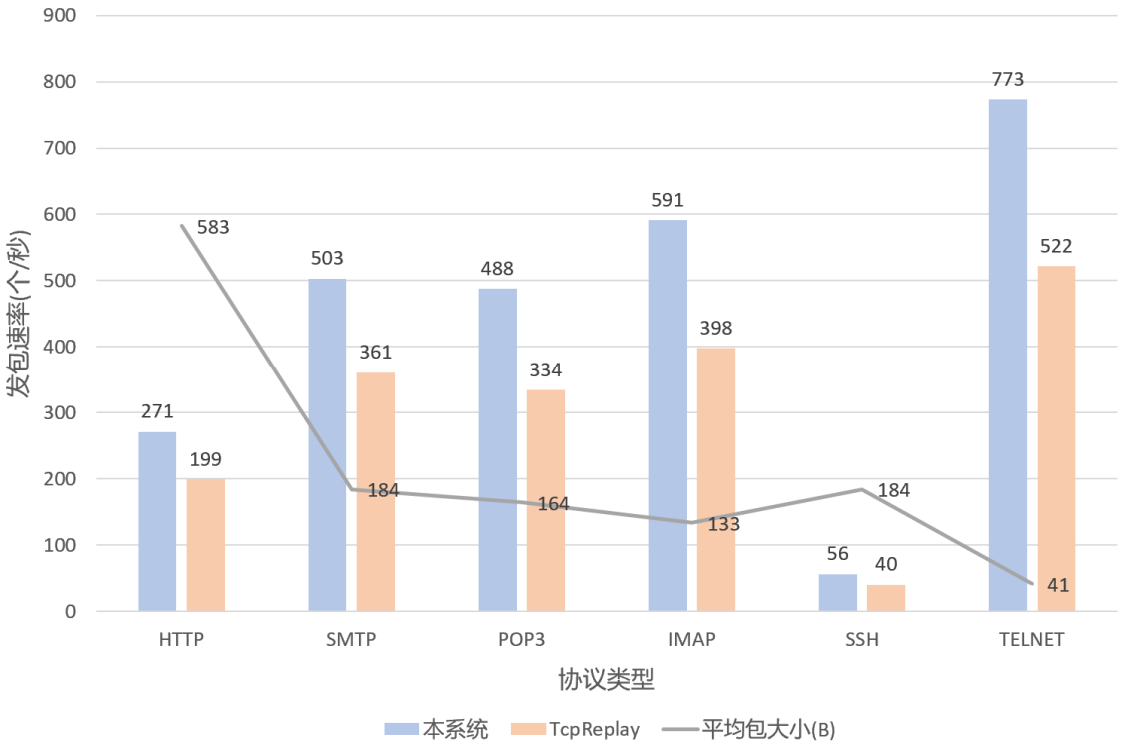

图 5-1 发包速率对比图

由于构建了协议自动机完成状态在规定时间自动转化，所以每一条仿真流量发包的时间都和原始数据流相同，因为对于不同协议的发包速率可能存在较大差距。分析图 5-1 可知，系统的发包速率主要受到包大小和自动机转化时间的影响，本仿真流量生成技术具有较高的性能。

5.4 本章小结

本章分别从功能和性能两个角度，设计和执行了对仿真流量生成系统的测试用例，验证了所有的系统功能，并测试了流量生成速率和和每个流量生成模块能达到对的最大带宽。测试结果表明面向网络靶场的自适应仿真流量生成技术能够满足所有的系统设计需求，系统设计合理。

结　论

本课题面向网络靶场的用户，针对网络靶场中自适应的仿真流量的生成，设计并实现了一种仿真流量生成系统，用于网络靶场的用户完成各种应用层协议仿真流量的精准构建，用户可以根据需求控制各种协议仿真流量的生成数量和比例，从而诱骗攻击者进行攻击和进行一些网络工具的测试。本文做了以下工作。

（1）设计实现了一种仿真流量生成系统。与传统的流量生成工具流量直接回放的实现方案不同，本文设计了新的一种流量生成方案。该方案完全基于原始数据流，通过对原始数据流的流量交互过程进行建模，实现了一个用户可以自定义的流量生成工具，这个工具所产生的仿真流量与原始数据流的交互过程几乎一致，因而拥有极高的仿真度，并且该仿真流量可以根据用户的需求进行部分内容的自定义。确立了一个仿真流量生成系统的需求、架构、目录空间、模块交互方式和具体实现细节。

（2）验证了系统的功能和性能。为了保证系统能够为网络靶场用户提供稳定的服务，本文设计了针对仿真流量系统的功能测试和性能测试。测试结果表明系统能够精准地完成各种协议仿真流量的构建，用户可以自定义各种协议仿真流量的数量，也可以可视化获取仿真流量产生结果。测试结果说明系统能够满足所有系统需求，系统设计正确。

当然，本系统也存在一些不足和可以提升的空间，需要在后续的工作安排中进行完善，主要包括以下方面。

（1）强依赖于原始数据流。在本系统开发的准备工作中，需要捕获各种协议的原始数据流，部分网络环境中捕获的原始数据流交互过程会非常复杂，且丢包和出错会很频繁，极大增加开发难度。后续应当探索能够捕获到最易分析的原始数据流的方案。

（2）性能可进一步提升。系统目前部署在开发机上，开发机的 CPU 为 AMD R7 4800H，该 CPU 为 8 核 16 线程，同时最多支持 16 个流量生成模块并发运行，虽然目前产生流量速率已经很快，但为了进一步提高流量生成速率，后续应该探索更多提高并发量的方案。如布置服务器集群、负载均衡等方式进行性能进一步升级。

参考文献

[1] 王小群, 丁丽, 严寒冰,等. 2020 年我国互联网网络安全态势综述[J]. 保密科学技术, 2021(5):8.

[2] 方滨兴, 贾焰, 李爱平,等. 网络空间靶场技术研究[J]. 信息安全学报, 2016(3):9.

[3] 刘红日, 吕思才, 王佰玲. 面向网络空间靶场的网络行为模拟关键技术研究[J]. 智能计算机与应用, 2021, 11(4):4

[4] Krunz M M , Makowski A M . Modeling video traffic using M/G/∞ input processes: A compromise between Markovian and LRD models[J]. IEEE Journal on Selected Areas in Communications, 2015, 16(5):733-748.

[5] Orebaugh A . Wireshark & Ethereal Network Protocol Analyzer Toolkit[J]. Jay Beales Open Source Security, 2006:523 – 540.

[6] 姜庆民, 吴宁, 闫申友. 网络分析软件 Tcpdump 的研究[J]. 智能计算机与应用, 2007, 000(002):21-22.

[7] Feng W C , Goel A , Bezzaz A , et al. TCPivo: a high-performance packet replay engine[C]// the ACM SIGCOMM workshop. ACM, 2003.

[8] Katris C , Daskalaki S . Comparing forecasting approaches for Internet traffic[J]. Expert Systems with Applications: An International Journal, 2015.

[9] Cortez P , Rio M , Rocha M , et al. Multi-scale Internet traffic forecasting using neural networks and time series methods[J]. Expert Systems, 2012, 29(2):143-155.

[10] Vishwanath K V , Vahdat A . Swing: Realistic and Responsive Network Traffic Generation[C]// IEEE Press. IEEE Press, 2009:712-725.

[11] Clegg R G . Simulating internet traffic with Markov-modulated processes[J]. proceedings of uk performance engineering workshop, 2007.

[12] Sommers J , Kim H , Barford P . Harpoon: a flow-level traffic generator for router and network tests[M]. ACM, 2004.

[13] Dainotti A , Pescape A , Rossi P S , et al. Internet traffic modeling by means of Hidden Markov Models[J]. Computer Networks, 2008, 52(14):2645-2662.

[14] Garg A , Sankaranarayanan V , Upadhyaya S , et al. USim: A User Behavior Simulation Framework for Training and Testing IDSes[J]. IEEE Computer Society, 2006.

[15] Klemm A , Lindemann C , Lohmann M . Modeling IP traffic using the batch Markovian arrival process[J]. Performance Evaluation, 2003, 54(2):149-173.

[16]李树军. Scapy 在网络协议分析实验教学中的应用[J]. 实验科学与技术, 2014, 12(6):5.

[17]汪云飞. JavaEE 开发的颠覆者:Spring Boot 实战[M]. 电子工业出版社, 2016.

[18]Mudunuri, Srinivas. Mybatis in Practice : a step by step approach for learning mybatis framework[M]. CreateSpace Independent Publishing Platform, 2013.

[19]侯俣, 刘万军. JFreeChart 在 Java Web 项目中的应用[J]. 科学技术与工程, 2008, 8(10):3.

[20]J Weaver. JavaFX Script: Dynamic Java Scripting for Rich Internet/Client-side Applications[J]. Apress, 2007.

哈尔滨工业大学本科毕业设计（论文）原创性声明

本人郑重声明：在哈尔滨工业大学攻读学士学位期间，所提交的毕业设计（论文）《面向网络靶场的自适应仿真流量生成技术》，是本人在导师指导下独立进行研究工作所取得的成果。对本文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，其它未注明部分不包含他人已发表或撰写过的研究成果，不存在购买、由他人代写、剽窃和伪造数据等作假行为。

本人愿为此声明承担法律责任。

作者签名：　　日期：　　年　　月　　日

致　谢

落笔题字至此，四年大学生涯也是接近了尾声。从一个普普通通的南方小城，到哈尔滨这么一个陌生的城市。从第一次看到大雪纷飞的喜悦，到以后少见的白雪皑皑之景。从下午七点的晚霞，到凌晨三点的日出。学生生涯总是如此，充斥着各种意外之喜和遗憾。在过去的几个月时间内，我都无比期待着穿上学士服，在主楼为大学生涯划上句号的情形。而当这天真的即将到来之时，好像有很多话想说，又好像不知从何说起。

我衷心感谢詹东阳老师对我的指导。在过去的一年时间内，詹老师对我毕设的选题、项目实现和论文写作给予了很多指导，使我的毕业设计能够按时完成。在毕设工作的推进过程中，詹老师提出了很多宝贵的意见和建议，为我提供了很多新的方向和想法。詹老师渊博的学识和对待科研严谨的态度，使我获益匪浅。

感谢我的父母，在完成毕设的这段时间内，我经常会因为一些项目推进中遇到困难而心烦意乱。父母经常性的关心，虽然隔着屏幕，也能让我平心静气，抚平最近的疲惫。

四年时间，说短不短，说长不长。做出了很多选择，为一些选择后悔过，也庆幸过自己的选择。周围不断变化的一切，为我构建起了认知这个世界的桥梁，我也相信这四年所建立的世界观，会为指明以后前进的方向。从决定工作而不考研，前前后后为了找工作付出了很多努力，也希望自己能够工作顺利。

时间在历史的长河里刻下了大学四年的青春。我很庆幸，四年时光我遇到了可以交心的室友，有过最纯真的学生时代的爱情，找到了自己还算满意的工作。我很庆幸，我在一个陌生的城市，在校园里面，在街头巷角，看到了人世界最纯真的生活百态。我也很庆幸，这短暂的四年内游历过祖国的大好河山，看过世间最美丽的风景。

写到这里，百感交集，一字一句落笔，都意味着一个篇章的完结，意味着一段新的生活的开始。脱离学生身份，从校园到社会，又何尝不是令人惊喜的过渡呢？我会一直相信最美好的存在，从高高庙堂，到浅浅世间，从高楼大厦，到人间烟火，都有着生活最纯正的美好。最后一句话，送给这四年的大学时光，送给看到这句话的你，也送给即将离开校园的自己：

踏上征程，或对或错，或真或假，又何尝不是美好呢？

最后，下课。