Socket 编程实践在计算机网络探究式课程教学中的应用*

毕宿志 林晓辉 侯华炜 苏恭超 顾一帆 全智

深圳大学电子与信息工程学院，深圳 518060

摘 要 在高校课堂运用探究式教学方法来提升学生自主学习能力、培养创新型人才是近年来一个备受关注的课题。本文提出在计算机网络探究式教学过程中融入网络 Socket 编程实践环节，针对 Socket 编程层次抽象化的特点设计了三个典型的 Socket 编程实验，并进一步提出了基于 Socket 编程的探究式教学拓展设计思路。经课堂教学实践，所提方法有助于提升计算机网络探究式课程教学效果，加深学生对于计算机网络层次化知识体系的理解，提升对具体工程问题自主分析和解决的能力。

关键词 计算机网络，Socket 编程，探索式教学

Socket Programming Experiments in the Exploratory Teaching of Computer Network Courses

BI Suzhi LIN Xiaohui HOU Huawei SU Gongchao, GU Yifan, QUAN Zhi

College of Information and Electronics Engineering
Shenzhen University
Shenzhen 518060, China

Abstract—The use of exploratory teaching methods in university education to improve students' learning ability and creativity is a topic that has attracted heated attention in recent years. This paper proposes to integrate network Socket programming into exploratory teaching of computer network, designs three Socket programming experiments according to the characteristics of Socket programming, and further proposes advanced design methods for extending Socket programming in the process of exploratory teaching. The proposed method can improve the effect of computer network exploratory teaching, deepen students' understanding of the layered structure of computer network, and improve the ability to independently analyze and solve specific engineering problems.

Keywords—Computer networks, Socket Programming, Exploratory teaching

1 引 言

《计算机网络》是电子信息、通信、计算机类学科共通的专业基础课程，课程内容面向下一代 6G 蜂窝网、物联网、卫星网、移动互联网等国民经济重大需求中的关键网络技术，涉及到网际互联原理、通信协议架构、网络性能分析等基础知识能力的培养。计算机网络的核心知识体系是 TCP/IP 通信协议的分层结构，通过将网络功能抽象为由上至下的应用层、传输层、网络层、数据链路层和物理层的层次化服务模型，上下层之间通过特定接口提供/接受服务，各自保持一定的独立性，有利于全局通信问题的简化、支持计算机网络软硬件技术的持续演进和发展[1]。计算机网络课程的讲解一般也遵循分层结构的思路，按照从下至上或相反的层次顺序逐层讲解。分层讲解模式对于初始接触网络协议架构的学生而言是最自然易懂的方法，然而由于各层内部知识繁复，导致课程教学内容往往“广”而“不深”，容易造成知识的快速遗忘。此外复杂的课程内容也导致教学注重定性原理讲解而忽略定量性能分析，学生缺乏协议性能分析能力的训练。在课程中学生大部分时间处于被动接受课程讲解的状态，缺乏对具体工程问题的判断和探索。

为了提升学生自主学习能力、培养创新型人才，近年来在高校课堂运用探究式教学方法成为一个备受关注的话题[2][3][4]。具体而言，探究式教学以学生为中心设置理论讲解、课程实验、文献查阅和项目实践等教学环节，不以传统意义上通过课堂讲授快速获取知识为目的，而是强调学习的自主性，注重培养个人在课题探究的过程中发现、分析、解释和解决问题的能力[3]。在多种探究式教学环节设置中，课程实验的设置

* **基金资助**：广东省高等教育教学改革建设项目（839/0000027228、839/0000027307，深圳大学教学改革研究项目（JG2021051，JG2022049）。

* * **通讯作者**：林晓辉（xhlin@szu.edu.cn），全智（zquan@szu.edu.cn）。

至关重要，对于计算机网络这样一个面向工程应用的课程更是必不可缺。然而当前计算机网络实验往往更注重网络设备的硬件配置，实验过程较为程序化，对于计算机网络的层次化结构和服务模型这一关键知识点的覆盖较少[5][6]。学生在实验中难以发挥创新设计能力，无法匹配探究式教学对学生自主学习过程的基础要求。

对此，本文提出在计算机网络探究式教学过程中融入网络 Socket 编程环节[7]。Socket 翻译为套接字，是应用层与 TCP/IP 协议族通信的中间软件抽象层，是应用程序操作网络协议栈的接口，反应了应用程序的数据进出过程。通过利用 Socket 编程层次抽象化的特点可以帮助学生理解计算机网络服务器、网络设备和终端之间的交互过程，分析各种不同应用程序的网络通信特点，掌握因特网层次化服务模型的技术内涵。在学生具备 Socket 编程基础上，进一步布置更为具体化的网络通信课题研究，让学生可以运用所掌握的编程技巧解决具体工程问题，自主观察分析实际网络性能并优化方案，达到探究式教学的目的。以此为课程目标，几个基础 Socket 编程实验教学内容的设置尤为重要，本文将以下进行详解。

2 计算机网络 Socket 编程简介

Socket 上联应用进程、下联网络协议栈，是应用层进程利用 TCP/IP 网络协议交换数据的接口。在 TCP/IP 协议栈中，针对应用程序的服务可靠性需求，一般分为 TCP Socket 和 UDP Socket，分别对应面向连接的 TCP 数据流服务和无连接的 UDP 数据报服务。本文 Socket 编程实验课程内容设计基于 Python 编程语言，是目前数据挖掘、算法推荐、控制脚本和人工智能等领域最热门的语言之一。

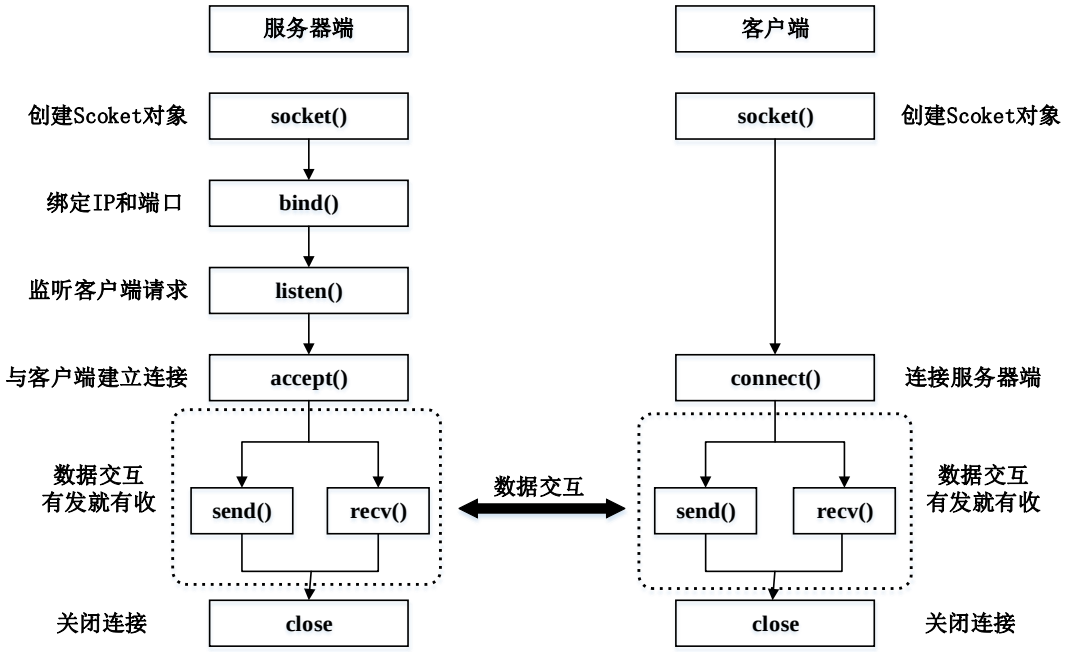

图 1 Socket 通信逻辑图

相比于使用各种成熟的仿真软件，如 eNSP[8]、OPNET[9]等，使用 Python 编程的实验更加接近于网络编程原理的底层逻辑，能更好的提升学生对于计算机网络各层次结构的核心算法和逻辑的理解和掌握。另外，Python 语言具有极高的自由度，本文设计的实验除了规定的要求内容之外还指明了技术拓展方向，学生使用基于 Python 的 Socket 编程有助于提升他们的自主探索学习能力。

基于 Python 的 Socket 编程允许使用者通过创建 Python 脚本的方式来实现服务器端（Server）和客户端（Client）之间的通信。通信逻辑图如图 1 所示。在 Python 中可以通过导入 socket 包，然后使用 socket.socket()的方法创建 socket 对象。下面以一段 Python 代码展示了创建基于 TCP 协议的 Socket。该代码主要实现的功能是客户端将用户输入的一串小写字母上传到服务器端；服务器端会将上传的小写字母全部转换为大写字母，并返回给客户端；客户端再将服务器发回来的信息在本地输出打印。

- 客户端代码

```
import socket
# 初始化服务器地址和端口号
serverName, serverPort = '127.0.0.1', 12000
# 创建 Socket 并连接服务器
clientSocket=socket.socket(
            socket.AF_INET,socket.SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
```

```
# 用户输入一串小写字母
sentence = input('Input lowercase sentence:')
# 将用户输入的内容编码为比特流发送至服务器
clientSocket.send(sentence.encode())
# 接收服务器返回的内容
modifiedSentence = clientSocket.recv(1024)
# 将接收的比特流解码为字符串并输出
print('From Server:', modifiedSentence.decode())
clientSocket.close()
```

- 服务器端代码

```
import socket
# 初始化端口号
serverPort = 12000
# 创建 TCP Socket
serverSocket = socket.socket(
               socket.AF_INET,socket.SOCK_STREAM)
# 绑定端口号
serverSocket.bind(('', serverPort))
# 开始监听
serverSocket.listen(1)
print('The server is ready to receive')
while True:
    # 建立连接
    connectionSocket, addr =
serverSocket.accept()
    # 接收客户端的数据
    sentence =
connectionSocket.recv(1024).decode()
    # 将接收到的字符串转换为大写
    capitalizedSentence = sentence.upper()
    connectionSocket.send(
                capitalizedSentence.encode())
    # 将字符串编码并发送回客户端
    connectionSocket.close()
    # 关闭 Socket
    clientSocket.close()
```

3 编程实践教学内容设计

本文根据计算机网络的分层结构以及实验的编程难度梯度，依次设计了三个实验，分别是：应用层的 Web 服务器搭建实验；传输层的 ARQ 重传协议；网络层的距离向量算法。三个实验均要求使用 Python 编程语言和基于 Python 的 Socket 编程来完成，对于初接触 Python 编程的学生，需要结合设置 Python 基础编程能力教学课时。

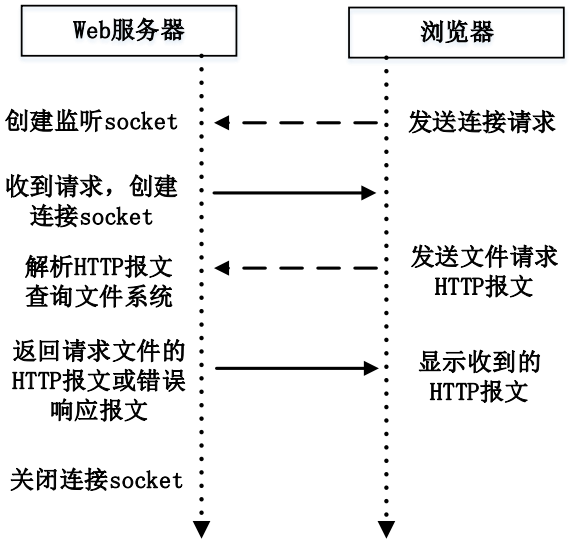

图 2 Web 服务器通信逻辑图

3.1 搭建简易 Web 服务器

本文根据计算机网络的分层结构以及实验的编程难度梯度，依次设计了三个实验，分别是：应用层的 Web 服务器搭建实验；传输层的 ARQ 重传协议；网络层的距离向量算法。三个实验均要求使用 Python 编程语言和基于 Python 的 Socket 编程来完成，对于初接触 Python 编程的学生，需要结合设置 Python 基础编程能力教学课时。

本实验将基于 Python 的 Socket 编程实现基本的 Web 服务器功能，帮助学生理解 HTTP 报文的结构和 HTML 网页文件的组成。要求自主探索 Python 语言中 Socket 工具包里面丰富的函数和功能，加深 Python 对于 HTTP 报文中字符串的处理功能。具体而言，实验要求 Web 服务器实现以下功能：

(1) 当有浏览器连接时创建一个 Socket 连接；
(2) 使用创建的连接接收 HTTP 请求；
(3) 解析请求从而确定客户端请求的文件；
(4) 从服务器端的文件系统获得该文件；
(5) 创建一个由请求的文件组成的 HTTP 响应报文；
(6) 经 TCP 连接向浏览器发送报文；

(7) 如果被请求一个不存在的文件，应当返回“404 Not Found”的响应。

该实验的通信逻辑图如图 2 所示。

部分核心代码如下：

```
# 与浏览器建立 socket 连接
connectionSocket, addr = serverSocket.accept()
message = connectionSocket.recv(2048)
  filename = message.split()[1].decode()
  if filename == "/":
      filename = "index.html"
  else:
      filename = filename[1:]
  # 解析浏览器报文，获得请求文本
  print("filename: {}".format(filename))
  try:
      with open(filename, "r") as f:
          outputdata = f.read()
      response = ("HTTP/1.1 200 OK\n"
                  "Server: Python 3.7.2\n"
                  "Content-Type: text/html;
                   charset=utf-8\r\n\n")
      # 发送 HTTP 报头和文件内容
      connectionSocket.send(response.encode())
      for i in range(0, len(outputdata)):
          connectionSocket.send(
                  outputdata[i].encode())
      connectionSocket.send("\r\n".encode())
      connectionSocket.close()
  except FileNotFoundError:
        # 发送错误响应报文
      response = ("HTTP/1.1 404 Not Found\n"
                  "Server: Python 3.7.2\n"
                  "Content-Type: text/html;
                  charset=utf-8\r\n\n")
     connectionSocket.sendall(response.encode())
      connectionSocket.close()
```

访问存在的文件和不存在的文件的实验结果预览如下图 3。

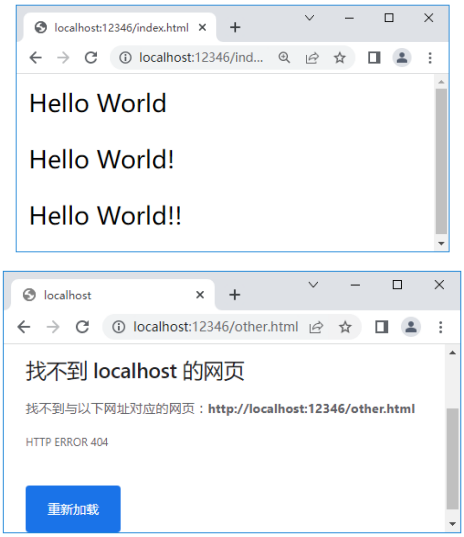

图 3 服务器响应结果。上：浏览器接收到请求文件；下：所请求文件不存在

3．2 自动重传请求协议(ARQ)

本实验基于Python的Socket编程实现传输层的基本停等传输协议（ARQ）[1]。实现的过程中，不仅需要考虑不同的丢包情况(发送分组丢失还是确认帧丢失)，还需要思考如何用编程实现手动设置的丢包机制。在技术拓展方面，可鼓励动手能力较强的学生尝试实现更高级的回退N步协议或者选择重传协议。实验内容上，学生需要编写一个服务器端和一个客户端的Python脚本，具体可以使用基于UDP或者TCP协议的Socket实现，具体要求如下：

(1) 客户端给服务器端发送一个数据包，并接收来自服务器端返回的确认帧，如果在规定的时间内没收到确认帧，客户端应该再次发送该数据包；

(2) 服务器端需要接收客户端的数据包，然后给客户端返回确认帧；

(3) 因现实的网络通信中，丢包率发生概率较低，所以需要手动设计一个虚拟的丢包机制，模仿实际网络中的丢包事件触发重传；

(4) 发包的数量可以自由定义；

(5) 在所有的数据包发送完毕后，客户端需要计算丢包率和总耗时。

该实验的通信逻辑图如图4所示。

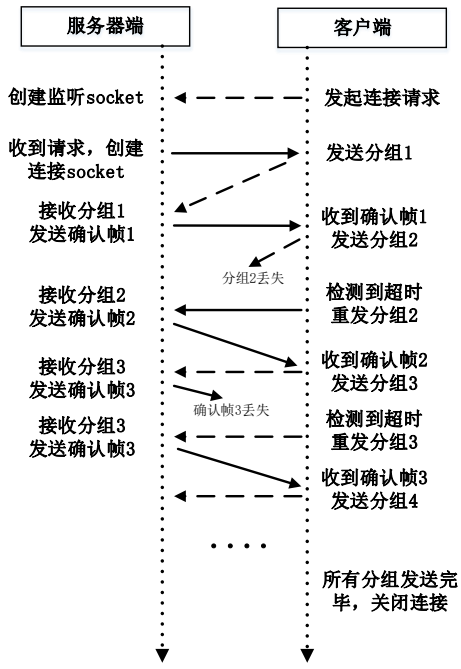

图 4 停等协议通信逻辑图，图中包含了分组丢失或者确认帧丢失的情况

由于版面受限，本文仅仅展示服务器部分的核心代码，其中涉及到手动设置10%丢包率的操作方法，模拟网络中的实际丢包场景。

```
import socket
import random
# 初始化地址和端口号
host, port = '', 10086
# 手动设置 10%的丢包率
rate = 10
server_socket = socket.socket(
        socket.AF_INET, socket.SOCK_STREAM)
# 创建 socket，绑定地址和端口号
server_socket.bind((host, port))
# 开始监听
server_socket.listen(1)
while True:
    # 开始监听
    connectSocket, addr = server_socket.accept()
    while True:
        # 接收数据包
        message = connectSocket.recv(1024)
        if message != b'':
            packet = 'receive the ' +
            message.decode()[-1] + ' packet'
            rand = random.randint(1, 100)
            if rand >= rate:
               # 以预设丢包率返回确认帧

connectSocket.sendall(packet.encode())
        else:
            break
```

```
运行: SAW_server × SAW_client ×
Send the 97 packet
ack received!
Send the 98 packet
ack received!
Send the 99 packet
ack received!
Send the 100 packet
ack received!
Sending 100 packet in total, packet loss rate = 0.11
Time spent 11019.592 microsecond

进程已结束，退出代码为 0
```

图 5　自动重传请求协议实验结果

图 5 展示了当手动设置的丢包率为 10%、发送 100 个数据包时，实际发生了 13 次重传的实验结果，其中统计了丢包率以及通信时延。

3．3 路由选择算法仿真

本实验将基于Python编程实现开放式最短路径优先路由选择算法（OSPF）[1]，要求学生掌握全局链路状态信息的交互获取方法，并对全局拓扑结构实现Dijkstra算法计算到网络其他节点的最小链路开销和最短路径。虽然本实验仅需要网络层的数据报交换不涉及到传输层Socket功能，但同样可以通过模拟网络节点间的交互过程，帮助学生理解和分析网络层路由表生成机制。在技术拓展方面，鼓励同学自主设计更加复杂的网络拓扑结构，或尝试实现其他路由选择算法。

如图6所示，实验中将给定网络拓扑图，其中每一个节点可以理解为一个子网，节点之间的路径消耗表示通信的开销（比如时延），要找到任意点到其他所有点的最小开销路径。实验具体要求如下：

(1) 所有通信节点生成本地链路状态数据报（LSP），通过洪泛法向其他网络节点广播自己本地的LSP；

(2) 经过多次迭代，所有节点汇总收集到的LSP，整合得到完整的全局网络拓扑图信息；

(3) 根据全局网络拓扑信息，输出某个指定节点到其它节点的最佳路径以及最小开销；

(4) 遍历任意节点到其他所有节点的最佳路径以及最小开销。

```
运行: Dijkstra1 ×
0号节点到其它节点的距离和最短开销路径
{0: 0, 1: 1.0, 2: 4.0, 3: 5.0, 5: 7.0, 4: 10.0}
{0: {0: [], 1: [1], 2: [1, 2], 3: [1, 3], 5: [1, 3, 5], 4: [1, 2, 4]}}
1号节点到其它节点的距离和最近路径
{1: 0, 0: 1.0, 2: 3.0, 3: 4.0, 5: 6.0, 4: 9.0}
{1: {1: [], 0: [0], 2: [2], 3: [3], 5: [3, 5], 4: [2, 4]}}
2号节点到其它节点的距离和最近路径
{2: 0, 3: 2.0, 1: 3.0, 0: 4.0, 5: 4.0, 4: 6.0}
{2: {2: [], 3: [3], 1: [1], 0: [1, 0], 5: [3, 5], 4: [4]}}
3号节点到其它节点的距离和最近路径
{3: 0, 5: 2.0, 2: 2.0, 1: 4.0, 0: 5.0, 4: 8.0}
{3: {3: [], 5: [5], 2: [2], 1: [1], 0: [1, 0], 4: [2, 4]}}
4号节点到其它节点的距离和最近路径
{4: 0, 2: 6.0, 3: 8.0, 1: 9.0, 0: 10.0, 5: 10.0}
{4: {4: [], 2: [2], 3: [2, 3], 1: [2, 1], 0: [2, 1, 0], 5: [2, 3, 5]}}
5号节点到其它节点的距离和最近路径
{5: 0, 3: 2.0, 2: 4.0, 1: 6.0, 0: 7.0, 4: 10.0}
{5: {5: [], 3: [3], 2: [3, 2], 1: [3, 1], 0: [3, 1, 0], 4: [3, 2, 4]}}
```

图 7　各节点间的最短距离和路径结果

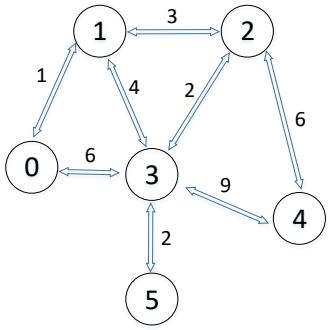

图 6　网络拓扑结构示意图

以图6中的拓扑结构为例，所有6个节点的最短路径计算结果如图7所示，受篇幅限制具体的Python代码

不在本文展示。

4 基于 Socket 编程的探究式教学拓展

本文根据计算机网络的分层结构以及实验的编程难度梯度，依次设计了三个实验，分别是：应用层的 Web 服务器搭建实验；传输层的 ARQ 重传协议；网络层的距离向量算法。三个实验均要求使用 Python 编程语言和基于 Python 的 Socket 编程来完成，对于初接触 Python 编程的学生，需要结合设置 Python 基础编程能力教学课时。

通过以上 3 个实验可以让学生理解应用层、传输层和网络层的典型应用运行流程和基础原理，并初步掌握网络性能指标分析的能力。在此基础上，可以引导学生去通过 Socket 编程简化实现更前沿的网络应用功能，比如 DNS（域名解析系统）、P2P、CDS（内容分发网络）等，编程实现传输层更复杂的控制原理，如拥塞控制、流量控制等，或者仿真网络层更复杂的路由寻址方法，如多播路由算法等，注重其中的节点交互过程和网络性能分析。

除了课本上已有的一些典型应用，还可以通过布置学生查阅近年计算机网络领域科技论文，编程实现尚未形成标准的通信协议，分析网络性能提升和额外系统开销，并鼓励对方法进行自主改进，达到发现问题、分析问题和解决问题的探究式学习目的。以建立和优化网络缓存服务器应用为例，可以基于 Socket 编程实践如下设置探究式教学环节：

（1）布置分组专题学习环节，让小组内的学生熟悉计算机网络缓存的基本运行机制，掌握计算机网络缓存部署对降低通信时延性能的定量分析方法；

（2） 组织调研近年来最新的网络缓存技术，关注网络缓存技术在移动互联网、蜂窝通信网络、内容分发网络、流媒体传输、边缘计算等前沿网络架构和应用上的研究进展[8]，总结分析网络缓存技术的发展趋势；

（3） 学生选取其中一个最近的研究工作，比如面向流媒体传输的蜂窝网络缓存部署课题[9]，建模小型网络中的流媒体数据文件的流行度、无线网络误码率、无线用户需求等参数，尝试理论分析现有缓存策略下系统的平均通信时延等性能指标；

（4） 基于 Socket 编程，仿真复现网络中用户与基站的数据交互过程，观察在网络缓存帮助下的系统通信时延提升，对比分析和观测数据，并尝试优化缓存部署策略，提升网络性能；

（5） 由学生针对研究课题进行分组课堂汇报，一部分介绍计算机网络缓存技术的前沿发展，另一方面展示针对专门课题研究下基于 Socket 编程得到的数值分析结果，总结研究所得。

以上教学进程可以以学生分组的形式组织开展，在教学过程中，教师需要定期关注学生进展，指导学生文献阅读和初步的问题建模分析，鼓励小组讨论，并引导学生关注课题的核心问题，根据学生的能力及时调整目标。通过抽象化的 Socket 编程实践，可以帮助学生理解网络通信协议的设计核心理念，探究科研课题中的工作难点和解决方法，培养学生自主探究和分析动手能力，提升计算机网络探究式课程教学效果。

以上教学方法于 2021-2022 学年在深圳大学“电子信息拔尖创新实验班”的《计算机网络》双语课程进行实施。在第 2 章节所设置的三个教学实验的基础上，由教师辅助划定范围、学生自主分组选题的形式，设立专题研究课题 15 项，课题名称包括“Caching strategies in computer networks”，“Video streaming optimization through SDN”等。引导学生对重点论文中所提出的前沿通信网络技术（比如内容驱动网络缓存、协作式数据流转发等）进行深入理解，与课本知识相结合分析性能提升的主要技术因素，并指导学生基于 Socket 编程对核心算法进行抽象简化和编程实现。在课题研究过程中学生展示出了很强的主动探索意识，与教师和助教频繁就网络技术和编程实践问题进行咨询讨论。每个小组通过约 6 周的专题研究，最终通过分组课堂汇报的形式汇报领域前沿技术概况、发展趋势和数值分析结果。

大部分课题研究结果达到了预设要求，能够在编程工具的辅助下自主分析一个细分课题的核心技术要点，部分学生展示出了对技术改进点和领域技术发展的深刻见解。经期末问卷调查显示：81.5%的同学经过课题研究训练初步掌握了总结文献创新点、技术贡献，判断论文的价值的能力；所有的同学都能够掌握 Socket 编程的服务器-用户基础通信过程，理解因特网网际互联基本原理和层次化协议栈体系，并应用于课题研究分析；85.2%的同学表示在基于 Socket 编程的课题探究中找到了研究的乐趣，有成就感。以上结果基本验证所实践的方法能够帮助学生加深对理论知识点的理解，在探究式教学中增强对工程技术问题的分析和解决能力，并且提升对技术研究工作的热情。

5 结束语

针对计算机网络探究式课程教学方法特点，本文提出了在计算机网络探究式教学过程中融入网络 Socket 编程环节。根据 TCP/IP 层次化架构的跨层交互特性，本文针对性地设计了应用层、传输层和网络层的三个 Socket 编程实验，并进一步提出了基于 Socket 编程的探究式教学拓展设计思路。经课堂教学实践，所提出的方法将有助于提升计算机网络探究式课程教学效果，加深学生对于计算机网络知识系统的理解，提升对具体工程问题自主分析和解决的能力。

参 考 文 献

[1] Kurose J F., Ross K. Computer networking a top-down approach [M], 8th ed, Pearson, Jan. 2022

[2] 罗玉川, 柳林, 徐明. 基于问题导向和深度学习的计算机网络课堂教学实践研究[J]. 计算机技术与教育学报, 2021(1):70-74

[3] 刘健西, 林炜. 大学探究式课堂教学的实践与探索: 内涵, 形式与评价[J]. 中国大学教学, 2018 (4): 30-33

[4] 王剑, 黄国辉. 以项目为依托开展探究式课堂教学模式的研究[J]. 教育教学论坛, 2017 (50): 172-173

[5] 杨波, 张菁, 王丽丹. 计算机网络技术实验课程改革研究[J]. 实验室科学, 2017, 20(6): 98-101

[6] 吴永辉，基于“编程解决问题”的程序设计语言实验——以程序设计方法的综合应用为例[J]. 计算机技术与教育报，2022(4)，56-60

[7] 苟英, 张晓华, 高博. Python 网络编程从入门到精通[M]. BEIJING BOOK CO. INC., 2021

[8] 许平, 胡曦. 基于华为 eNSP 工具在网络工程设计实验教学中的应用[J]. 中文科技期刊数据库 (文摘版) 工程技术, 2017 (06): 00106-00106

[9] 刘宴涛, 秦娜. 基于 OPNET 的按需组播路由协议仿真[J]. 计算机应用, 2021, 41(S01): 175-179

[10] Jedari B, Premsankar G, Illahi G, et al. Video caching, analytics, and delivery at the wireless edge: a survey and future directions [J]. IEEE Communications Surveys & Tutorials, 2020(1): 431-471

[11] Li L, Zhao G, Blum R S. A survey of caching techniques in cellular networks: Research issues and challenges in content placement and delivery strategies[J]. IEEE Communications Surveys & Tutorials, 2018(3): 1710-1732