# **计算机网络课程设计报告**

**学院：**计算机学院

**专业：**软件工程

**年级班别：**

**学号：**

**姓名：**

## **Ping 程序设计与实现 (Java)**

### **摘要**

本项目旨在设计并实现一个基于 Java 语言的 Ping 模拟程序，以理解网络层协议（特别是 ICMP 协议）的工作机制。程序将模拟 ping 命令的核心功能，包括向指定 IP 地址或主机名发送探测请求，测量网络延迟（往返时间 RTT），并报告主机的可达性。此外，程序将统计发送和接收到的 Ping 报文数量。鉴于 Java 标准库对原始套接字（Raw Sockets）的限制，本项目将探讨并采用 InetAddress.isReachable() 方法实现可达性检测，并在此基础上模拟 Ping 命令的关键参数，同时明确指出纯 Java 实现中对 ICMP 报头（如 TTL）直接解析的局限性。

### **1. 绪论**

#### **1.1 项目背景**

随着互联网的飞速发展，网络已成为我们日常生活和工作中不可或缺的一部分。在网络通信中，了解远程主机的可达性、网络延迟等信息对于网络故障诊断、性能评估至关重要。ping 命令作为网络诊断工具的基石，广泛应用于各种操作系统中，其背后是 ICMP 协议在支撑。

#### **1.2 项目目标**

本项目旨在通过 Java 语言实现一个简化的 Ping 程序，以加深对以下概念的理解：

* **ICMP 协议**：理解其报文格式和回显请求/应答机制。
* **网络编程基础**：掌握 Java 中 InetAddress 等网络相关类的使用。
* **网络可达性测试**：实现向目标主机发送探测包并接收响应的功能。
* **性能参数测量**：计算往返时间（RTT）并展示生存时间（TTL）等信息。
* **统计功能**：记录成功与失败的 Ping 次数。

### **2. 相关技术**

#### **2.1 Ping 原理**

ping 命令利用 **Internet 控制消息协议 (ICMP)** 的 **回显请求 (Echo Request)** 和 **回显应答 (Echo Reply)** 消息来测试两台主机之间的连通性。当用户执行 ping 命令时，源主机向目标主机发送一个 ICMP 回显请求报文。如果目标主机可达，它将回复一个 ICMP 回显应答报文。通过测量发送请求到接收应答之间的时间间隔，ping 命令可以估算网络延迟（往返时间 RTT）。同时，IP 报头中的 **生存时间 (TTL)** 字段可以指示报文在网络中可以经过的最大路由器跳数，每次经过路由器时 TTL 值减一，当 TTL 减为零时报文被丢弃。

#### **2.2 ICMP 协议**

ICMP 协议是 TCP/IP 协议族中的一个核心协议，它主要用于在 IP 网络中发送控制消息，报告网络状态、错误或查询信息。ICMP 报文封装在 IP 数据报中。

**ICMP 回显请求/应答报文格式：**

|  |  |  |
| --- | --- | --- |
| **字段** | **长度（字节）** | **描述** |
| 类型 (Type) | 1 | 对于回显请求为 8，回显应答为 0 |
| 代码 (Code) | 1 | 对于回显请求/应答为 0 |
| 校验和 (Checksum) | 2 | 整个 ICMP 报文的校验和 |
| 标识符 (Identifier) | 2 | 进程 ID 或序列号，用于匹配请求与应答 |
| 序列号 (Sequence Number) | 2 | 报文序列号，用于匹配请求与应答及乱序判断 |
| 数据 (Data) | 变长 | 任意数据，通常用于填充报文，可包含时间戳 |

#### **2.3 Java 网络编程中的 InetAddress**

在 Java 中，java.net 包提供了进行网络编程的基础类。InetAddress 类是其中非常重要的一个，它用于表示 IP 地址。

* **InetAddress.getByName(String host)**：根据主机名或 IP 地址字符串获取 InetAddress 实例。
* **InetAddress.getByAddress(byte[] addr)**：根据字节数组获取 InetAddress 实例。
* **InetAddress.getHostName()**：获取主机名。
* **InetAddress.getHostAddress()**：获取 IP 地址字符串。
* **InetAddress.isReachable(int timeout)**：尝试检测主机在指定时间内是否可达。**此方法在内部可能使用 ICMP Echo Request 或 TCP/UDP 连接，具体取决于操作系统和 JVM 的实现。它的优点是简单易用，缺点是无法直接获取 ICMP 报文的详细信息（如 TTL）和原始报文的往返时间，通常只返回布尔值。**

#### **2.4 原始套接字 (Raw Sockets)**

原始套接字允许程序直接访问网络层协议（如 IP、ICMP），从而可以手动构造和解析网络数据包的头部。这是 ping 命令能够自定义 IP 和 ICMP 报文的关键。

**在 Java 中实现原始套接字存在显著的局限性：**

* **安全限制：** 出于安全考虑，Java 标准库通常不允许直接创建和操作原始套接字。这意味着你无法直接在 Java 中构造一个 IP 包，然后将 ICMP 报文放入其中并发送。
* **平台依赖性：** 即使通过 JNI (Java Native Interface) 调用 C/C++ 代码来访问操作系统的原始套接字接口，这种方法也高度依赖于特定的操作系统和权限。
* **课程项目中的折衷：** 鉴于上述限制，对于通常的 Java 课程项目，如果明确要求“原始套接字”，通常有以下几种处理方式：
  1. **理论层面讨论：** 解释原始套接字的概念，并指出 Java 标准库的局限性，然后采用 InetAddress.isReachable() 进行实现，并注明其与真实 ping 的区别。
  2. **使用第三方库：** 引入如 Jpcap 或 Pcap4J 这样的第三方库，它们通过 JNI 封装了底层的数据包捕获和发送功能。但这会增加项目复杂性，需要额外的库配置和依赖。
  3. **模拟而非真实实现：** 像本报告中将要做的，我们用 InetAddress.isReachable() 来模拟可达性，通过测量方法执行时间来模拟 RTT，并说明 TTL 无法直接获取，以满足课程设计的核心要求。

本项目将主要依赖 InetAddress.isReachable() 实现可达性检测和 RTT 测量，并在此基础上进行结果统计和展示。对于 TTL，我们将在输出中占位并解释其无法直接获取的原因。

### **3. 系统设计**

#### **3.1 总体架构**

本 Ping 程序将采用模块化设计，主要分为以下几个部分：

1. **命令行参数解析模块**：负责解析用户输入的 IP 地址或主机名，以及 Ping 的次数。
2. **Ping 核心逻辑模块**：包含发送 Ping 请求、接收响应、计算 RTT 和统计结果的功能。
3. **结果展示模块**：负责将 Ping 结果（可达性、RTT、TTL、统计信息）格式化输出到控制台。

graph TD  
 A[程序启动] --> B{解析命令行参数};  
 B -- 有效参数 --> C[获取目标IP地址/范围];  
 C --> D{循环执行Ping请求};  
 D -- 每次请求 --> E[发送Ping探测];  
 E --> F[接收响应并测量时间];  
 F --> G{判断是否可达?};  
 G -- 可达 --> H[记录成功, 计算RTT, 显示TTL];  
 G -- 不可达 --> I[记录失败, 显示超时];  
 H --> J[更新统计数据];  
 I --> J;  
 J -- 循环结束 --> K[显示最终统计结果];  
 K --> L[程序结束];  
 B -- 无效参数 --> M[显示用法信息];  
 M --> L;

#### **3.2 类设计**

为了保持代码的清晰和可维护性，我们将设计一个主类 PingUtility 来封装 Ping 程序的全部功能。

* **PingUtility 类**：
  + **职责**：程序的入口点，负责命令行参数解析、Ping 流程的控制、结果的统计和输出。
  + **主要方法**：
    - main(String[] args)：程序主方法，处理命令行输入并调用 Ping 逻辑。
    - pingHost(String host, int count, int timeout)：核心 Ping 逻辑，对单个主机进行多次 Ping。
    - parseIpRange(String ipRange)：解析 IP 地址范围字符串（可选功能）。
    - displayUsage()：显示程序使用说明。

#### **3.3 程序流程**

1. **程序启动**：main 方法开始执行。
2. **参数解析**：解析 args 数组，获取目标主机/IP、Ping 次数和超时时间。如果参数无效，显示使用说明并退出。
3. **目标识别**：根据输入判断是单个 IP/主机名还是一个 IP 范围。
4. **循环 Ping**：
   * 对于每个目标 IP 地址，执行指定次数的 Ping 操作。
   * **发送探测**：利用 InetAddress.isReachable() 方法探测目标主机的可达性。
   * **时间测量**：在调用 isReachable() 前后记录系统时间，计算 RTT。
   * **结果判断**：根据 isReachable() 的返回值判断是否可达。
   * **信息输出**：
     + 如果可达，输出目标 IP、RTT 和一个关于 TTL 的说明（因为无法直接获取）。
     + 如果不可达，输出目标 IP 和超时信息。
   * **统计更新**：累加成功或失败的 Ping 次数。
5. **最终统计**：所有 Ping 操作完成后，输出总发送、总接收、成功率等统计信息。

### **4. 详细实现**

本节将提供 PingUtility 类的主体代码，展示如何利用 Java 的 InetAddress 类实现 Ping 功能。

#### **4.1 PingUtility.java 核心代码**

```{java}  
import java.net.InetAddress;  
import java.net.UnknownHostException;  
import java.io.IOException;  
import java.util.regex.Matcher;  
import java.util.regex.Pattern;  
import java.util.ArrayList;  
import java.util.List;  
  
/\*\*  
 \* 计算机网络课程设计：Ping程序设计与实现 (Java)  
 \*  
 \* 本程序模拟Ping命令的基本功能，通过Java的InetAddress.isReachable()方法  
 \* 检测主机可达性，并测量往返时间（RTT）。  
 \* 注意：由于Java标准API的限制，无法直接获取ICMP报头中的TTL字段。  
 \*  
 \* 用法:  
 \* java PingUtility <主机名/IP地址> [次数] [超时时间(ms)]  
 \* java PingUtility <起始IP-结束IP> [次数] [超时时间(ms)]  
 \*  
 \* 示例:  
 \* java PingUtility [www.google.com](https://www.google.com) 4 2000  
 \* java PingUtility 192.168.1.1 5  
 \* java PingUtility 192.168.1.100-192.168.1.105 3  
 \*/  
public class PingUtility {  
  
 private static final int DEFAULT\_PING\_COUNT = 4; // 默认Ping次数  
 private static final int DEFAULT\_TIMEOUT\_MS = 2000; // 默认超时时间（毫秒）  
  
 // 用于解析IP范围的正则表达式  
 private static final Pattern IP\_RANGE\_PATTERN = Pattern.compile(  
 "(\\d{1,3}\\.\\d{1,3}\\.\\d{1,3}\\.)(\\d{1,3})-(\\d{1,3})"  
 );  
  
 public static void main(String[] args) {  
 // --- 1. 命令行参数解析 ---  
 if (args.length == 0) {  
 displayUsage(); // 显示使用说明  
 return;  
 }  
  
 String target = args[0]; // 目标主机名、IP地址或IP范围  
 int count = DEFAULT\_PING\_COUNT;  
 int timeout = DEFAULT\_TIMEOUT\_MS;  
  
 // 解析Ping次数  
 if (args.length >= 2) {  
 try {  
 count = Integer.parseInt(args[1]);  
 if (count <= 0) {  
 System.out.println("错误: Ping次数必须是正整数。");  
 displayUsage();  
 return;  
 }  
 } catch (NumberFormatException e) {  
 System.out.println("错误: Ping次数格式不正确。");  
 displayUsage();  
 return;  
 }  
 }  
  
 // 解析超时时间  
 if (args.length >= 3) {  
 try {  
 timeout = Integer.parseInt(args[2]);  
 if (timeout <= 0) {  
 System.out.println("错误: 超时时间必须是正整数。");  
 displayUsage();  
 return;  
 }  
 } catch (NumberFormatException e) {  
 System.out.println("错误: 超时时间格式不正确。");  
 displayUsage();  
 return;  
 }  
 }  
  
 List<String> targetHosts = new ArrayList<>();  
 Matcher matcher = IP\_RANGE\_PATTERN.matcher(target);  
  
 // --- 2. 判断是单个IP/主机名还是IP范围 ---  
 if (matcher.matches()) {  
 // 是IP范围  
 String baseIp = matcher.group(1); // 提取IP地址前三段 (e.g., "192.168.1.")  
 int start = Integer.parseInt(matcher.group(2)); // 起始段 (e.g., 100)  
 int end = Integer.parseInt(matcher.group(3)); // 结束段 (e.g., 105)  
  
 if (start > end || start < 0 || end > 255) {  
 System.out.println("错误: IP范围不合法。");  
 displayUsage();  
 return;  
 }  
  
 for (int i = start; i <= end; i++) {  
 targetHosts.add(baseIp + i);  
 }  
 } else {  
 // 是单个主机名或IP地址  
 targetHosts.add(target);  
 }  
  
 // --- 3. 循环对每个目标进行Ping操作 ---  
 System.out.println("\n正在Ping " + target + "，请稍候...");  
 for (String host : targetHosts) {  
 System.out.println("\n----------------------------------------");  
 System.out.println("目标主机: " + host);  
 pingHost(host, count, timeout);  
 }  
 System.out.println("\n----------------------------------------");  
 System.out.println("Ping操作完成。");  
 }  
  
 /\*\*  
 \* 对单个主机执行Ping操作并显示结果。  
 \* @param host 要Ping的主机名或IP地址。  
 \* @param count Ping的次数。  
 \* @param timeout 超时时间（毫秒）。  
 \*/  
 private static void pingHost(String host, int count, int timeout) {  
 int sent = 0;  
 int received = 0;  
 long totalRtt = 0;  
 long minRtt = Long.MAX\_VALUE;  
 long maxRtt = Long.MIN\_VALUE;  
  
 try {  
 InetAddress address = InetAddress.getByName(host); // 获取目标IP地址对象  
 System.out.println("解析IP地址: " + address.getHostAddress());  
  
 for (int i = 0; i < count; i++) {  
 sent++; // 发送计数器递增  
 long startTime = System.nanoTime(); // 记录开始时间  
 boolean reachable = address.isReachable(timeout); // 调用isReachable进行探测  
 long endTime = System.nanoTime(); // 记录结束时间  
  
 if (reachable) {  
 received++; // 接收计数器递增  
 long rttMs = (endTime - startTime) / 1\_000\_000; // 计算RTT（毫秒）  
 totalRtt += rttMs;  
 minRtt = Math.min(minRtt, rttMs);  
 maxRtt = Math.max(maxRtt, rttMs);  
  
 // 模拟Ping输出格式  
 // 注意：Java的isReachable()无法直接获取IP包头中的TTL，这里仅作占位说明。  
 System.out.printf("来自 %s 的回复: 时间=%dms TTL=无法获取(Java API限制)\n",  
 address.getHostAddress(), rttMs);  
 } else {  
 System.out.printf("来自 %s 的请求超时。\n", address.getHostAddress());  
 }  
  
 // 每次Ping之间暂停一小段时间，避免请求过于频繁  
 try {  
 Thread.sleep(500); // 暂停500毫秒  
 } catch (InterruptedException e) {  
 Thread.currentThread().interrupt(); // 恢复中断状态  
 System.out.println("Ping过程被中断。");  
 return;  
 }  
 }  
  
 // --- 4. 显示统计结果 ---  
 System.out.println("\n" + address.getHostAddress() + " 的 Ping 统计信息:");  
 System.out.printf(" 数据包: 已发送 = %d，已接收 = %d，丢失 = %d (%.0f%% 丢失)\n",  
 sent, received, (sent - received), ((double)(sent - received) / sent) \* 100);  
  
 if (received > 0) {  
 System.out.printf("往返行程的估计时间(以毫秒为单位):\n");  
 System.out.printf(" 最短 = %dms，最长 = %dms，平均 = %.0fms\n",  
 minRtt, maxRtt, (double)totalRtt / received);  
 }  
  
 } catch (UnknownHostException e) {  
 System.out.println("错误: 未知主机 " + host + "。请检查主机名或IP地址。");  
 } catch (IOException e) {  
 // isReachable可能抛出IOException (例如，网络不可达或权限问题)  
 System.out.println("错误: Ping过程中发生I/O错误: " + e.getMessage());  
 System.out.println("请检查网络连接或程序运行权限。");  
 }  
 }  
  
 /\*\*  
 \* 显示程序的使用说明。  
 \*/  
 private static void displayUsage() {  
 System.out.println("用法:");  
 System.out.println(" java PingUtility <主机名/IP地址> [次数] [超时时间(ms)]");  
 System.out.println(" java PingUtility <起始IP-结束IP> [次数] [超时时间(ms)]");  
 System.out.println("\n示例:");  
 System.out.println(" java PingUtility [www.google.com](https://www.google.com) 4 2000");  
 System.out.println(" java PingUtility 192.168.1.1 5");  
 System.out.println(" java PingUtility 192.168.1.100-192.168.1.105 3 1500");  
 System.out.println("\n参数说明:");  
 System.out.println(" <主机名/IP地址> : 必填，要Ping的目标主机名或IP地址。");  
 System.out.println(" <起始IP-结束IP> : 必填，Ping一个IP地址范围（仅支持IPv4最后一段）。");  
 System.out.println(" [次数] : 可选，Ping的次数，默认 " + DEFAULT\_PING\_COUNT + " 次。");  
 System.out.println(" [超时时间(ms)] : 可选，每次Ping的超时时间（毫秒），默认 " + DEFAULT\_TIMEOUT\_MS + " ms。");  
 }  
}