网络编程

作者：少林之巅

目录

TCP/IP协议介绍

1. 互联网起源

   A. 起源于美国五角大楼，它的前身是美国国防部高级研究计划局主持研制的ARPAnet。

   B. 互联网的基础是TCP/IP协议

   C. TCP/IP 是供已连接因特网的计算机进行通信的通信协议。

TCP/IP协议介绍

2. TCP/IP协议

A. TCP (传输控制协议) - 应用程序之间通信。

B. UDP (用户数据包协议) - 应用程序之间的简单通信

C. IP (网际协议) - 计算机之间的通信。

D. DHCP (动态主机配置协议) - 针对动态寻址。

TCP协议

3. TCP协议

A. 面向连接的协议

B. 可靠传输，发送的数据保证对方能够收到。

C. 保证时序，发送的数据按发送的顺序到达。

D. 全双工的。

IP协议

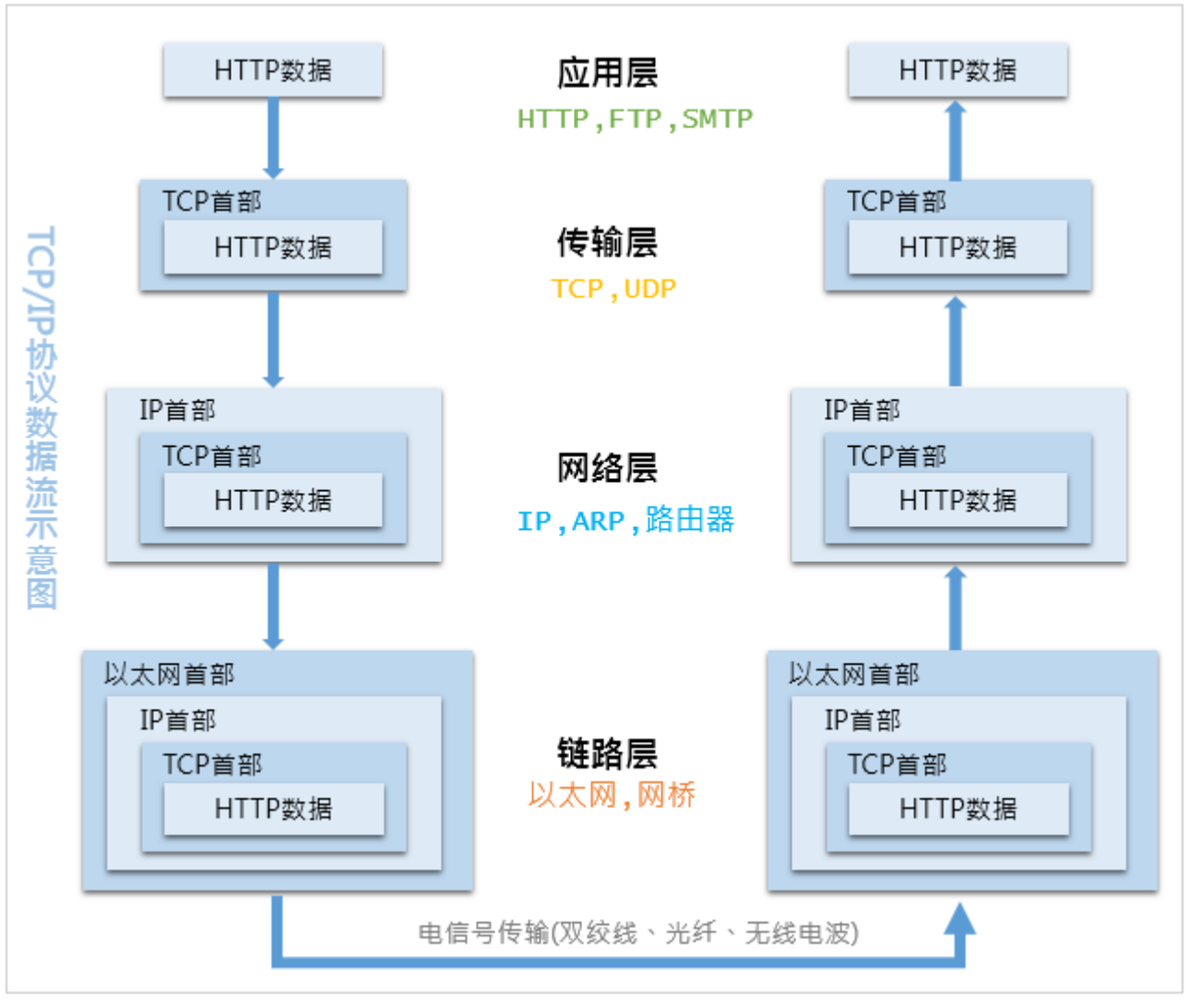
4. IP协议

A. 用于计算机之间进行通信，是TCP/UDP协议的底层

B. IP是无连接的，负责把数据包路由到目的地。

TCP/IP协议

5. TCP/IP协议图解



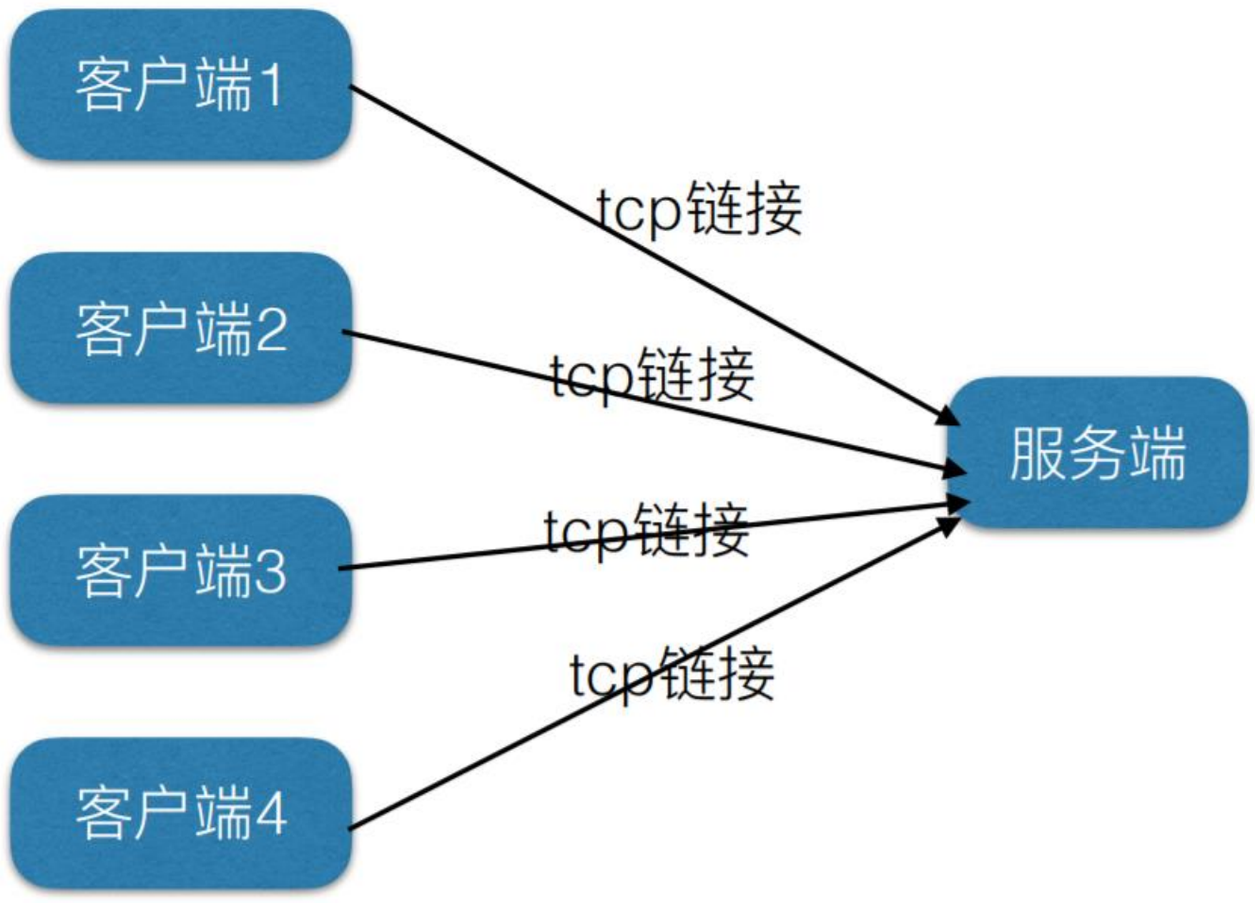
6. TCP协议基础

A. 通过IP和端口进行计算机之间进行访问

B. 域名和知名端口，http:80，https:443，ssl：22端口等

TCP服务端编程

7. Tcp服务端编程模型



TCP服务端编程

8. 服务端处理流程

a. 监听端口

b. 接收客户端的链接

c. 创建goroutine，处理该链接

```
package main

import (
	"fmt"
	"net"
)
func main() {
	fmt.Println("start server...")
	listen, err := net.Listen("tcp", "0.0.0.0:50000")
	if err != nil {
		fmt.Println("listen failed, err:", err)
		return
	}
	for {
		conn, err := listen.Accept()
		if err != nil {
			fmt.Println("accept failed, err:", err)
			continue
		}
		go process(conn)
	}
}
func process(conn net.Conn) {
	defer conn.Close()
	for {
		buf := make([]byte, 512)
		_, err := conn.Read(buf)
		if err != nil {
			fmt.Println("read err:", err)
			return
		}
		fmt.Println("read: ", string(buf))
	}
}
```

TCP编程

10. 客户端编程模式

a. 建立与服务端的链接

b. 进行数据收发

c. 关闭链接

```
package main

import (
	"bufio"
	"fmt"
	"net"
	"os"
	"strings"
)

func main() {

	conn, err := net.Dial("tcp", "localhost:50000")
	if err != nil {
		fmt.Println("Error dialing", err.Error())
		return
	}

	defer conn.Close()
	inputReader := bufio.NewReader(os.Stdin)
	for {
		input, _ := inputReader.ReadString('\n')
		trimmedInput := strings.Trim(input, "\r\n")
		if trimmedInput == "Q" {
			return
		}
		_, err = conn.Write([]byte(trimmedInput))
		if err != nil {
			return
		}
	}
}
```

12.发送HTTP请求

A. HTTP协议是基于TCP协议之上的文本协议。

B. 每行文本使用\r\n结尾，当连续两个\r\n时，表示整个数据包结束。

```
package main

import (
	"fmt"
	"io"
	"net"
)
func main() {

	conn, err := net.Dial("tcp", "www.baidu.com:80")
	if err != nil {
		fmt.Println("Error dialing", err.Error())
		return
	}
	defer conn.Close()
	msg := "GET / HTTP/1.1\r\n"
	msg += "Host: www.baidu.com\r\n"
	msg += "Connection: close\r\n"
	msg += "\r\n\r\n"

	_, err = io.WriteString(conn, msg)
	if err != nil {
		fmt.Println("write string failed, ", err)
		return
	}
	buf := make([]byte, 4096)
	for {
		count, err := conn.Read(buf)
		if err != nil {
			break
		}
		fmt.Println(string(buf[0:count]))
	}
}
```

UDP协议

13. UDP协议

A. 用户数据库报协议

B. 无连接，直接进行数据发送

C. 不可靠、没有时序

D. 实时性比较好，通常用于视频直播相关领域。

UDP服务端代码

```
package main
import (
    "fmt"
    "net"
)
func main() {
    // 创建监听
    socket, err := net.ListenUDP("udp4", &net.UDPAddr{
        IP:   net.IPv4(0, 0, 0, 0),
        Port: 8080,
    })
    if err != nil {
        fmt.Println("监听失败!", err)
        return
    }
    defer socket.Close()
    for {
        // 读取数据
        data := make([]byte, 4096)
        read, remoteAddr, err := socket.ReadFromUDP(data)
        if err != nil {
            fmt.Println("读取数据失败!", err)
            continue
        }
        fmt.Println(read, remoteAddr)
        fmt.Printf("%s\n\n", data)
        // 发送数据
        senddata := []byte("hello client!")
        _, err = socket.WriteToUDP(senddata, remoteAddr)
        if err != nil {
            return
            fmt.Println("发送数据失败!", err)
        }
    }
}
```

UDP客户端代码

```
package main
import (
    "fmt"
    "net"
)

func main() {
    // 创建连接
    socket, err := net.DialUDP("udp4", nil, &net.UDPAddr{
        IP:   net.IPv4(192, 168, 1, 103),
        Port: 8080,
    })
    if err != nil {
        fmt.Println("连接失败!", err)
        return
    }
    defer socket.Close()

    // 发送数据
    senddata := []byte("hello server!")
    _, err = socket.Write(senddata)
    if err != nil {
        fmt.Println("发送数据失败!", err)
        return
    }
    // 接收数据
    data := make([]byte, 4096)
    read, remoteAddr, err := socket.ReadFromUDP(data)
    if err != nil {
        fmt.Println("读取数据失败!", err)
        return
    }
    fmt.Println(read, remoteAddr)
    fmt.Printf("%s\n", data)
}
```