Net-Centric Lab 4

Student: Nguyen Duc Toan

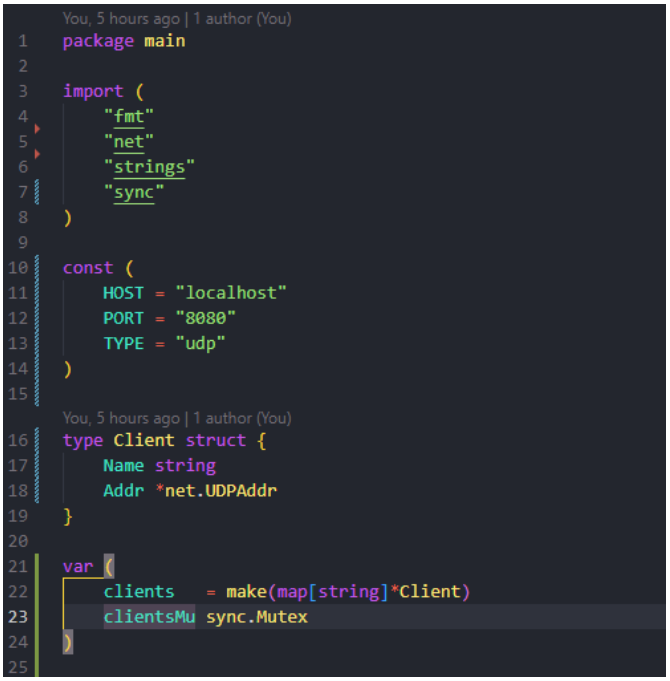
ID: ITCSIU21112

Client.go

```
You, 5 hours ago | 1 author (You)
package main

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

const (
    HOST = "localhost"
    PORT = "8080"
    TYPE = "udp"
)

func main() {
    serverAddr := HOST + ":" + PORT

    // Resolve server address
    s, err := net.ResolveUDPAddr("udp4", serverAddr)
    if err != nil {
        fmt.Println("Error resolving address:", err)
        return
    }

    // Dial UDP connection
    conn, err := net.DialUDP("udp4", nil, s)
    if err != nil {
        fmt.Println("Error connecting to server:", err)
        return
    }
    defer conn.Close()

    reader := bufio.NewReader(os.Stdin)
    fmt.Print("Enter your name: ")
    username, _ := reader.ReadString('\n')
    username = strings.TrimSpace(username)

    // Register the username with the server
    _, err = conn.Write([]byte("LOGIN " + username))
    if err != nil {
        fmt.Println("Error sending username to server:", err)
        return
    }

    fmt.Printf("Connected to UDP server at %s\n", conn.RemoteAddr().String())

```

```
    // Start a goroutine to listen for incoming messages from the server
    go func() {
        for {
            buffer := make([]byte, 1024)
            n, _, err := conn.ReadFromUDP(buffer)
            if err != nil {
                fmt.Println("Error reading from server:", err)
                return
            }
            fmt.Printf("\nMessage: %s\n>> ", string(buffer[:n]))
        }
    }()

    // Handle client input
    for {
        fmt.Print(">> ")
        text, _ := reader.ReadString('\n')
        text = strings.TrimSpace(text)

        // Check for exit command
        if text == "STOP" {
            _, err = conn.Write([]byte("LOGOUT " + username))
            if err != nil {
                fmt.Println("Error logging out:", err)
            }
            fmt.Println("Exiting UDP client!")
            return
        }

        // Handle private and broadcast messaging
        if strings.HasPrefix(text, "@") {
            _, err = conn.Write([]byte("MSG " + text))
        } else {
            _, err = conn.Write([]byte("MSG @all " + text))
        }

        if err != nil {
            fmt.Println("Error sending message:", err)
            continue
        }
    }
}

```

Server.go



```
You, 5 hours ago | 1 author (You)
package main

import (
    "fmt"
    "net"
    "strings"
    "sync"
)

const (
    HOST = "localhost"
    PORT = "8080"
    TYPE = "udp"
)

You, 5 hours ago | 1 author (You)
type Client struct {
    Name string
    Addr *net.UDPAddr
}

var (
    clients   = make(map[string]*Client)
    clientsMu sync.Mutex
)
```

```
func main() {
    serverAddr := HOST + ":" + PORT

    // Resolve server address
    addr, err := net.ResolveUDPAddr("udp4", serverAddr)
    if err != nil {
        fmt.Println("Error resolving address:", err)
        return
    }

    // Listen on UDP
    conn, err := net.ListenUDP("udp4", addr)
    if err != nil {
        fmt.Println("Error listening:", err)
        return
    }
    defer conn.Close()

    buffer := make([]byte, 1024)
    fmt.Println("Server listening on", serverAddr)

    for {
        n, clientAddr, err := conn.ReadFromUDP(buffer)
        if err != nil {
            fmt.Println("Error reading from UDP:", err)
            continue
        }
        message := strings.TrimSpace(string(buffer[:n]))
        fmt.Println("Received:", message, "from", clientAddr)

        // Process commands
        go handleCommand(conn, clientAddr, message)
    }
}
```

```go
func handleCommand(conn *net.UDPConn, addr *net.UDPAddr, message string) {
    parts := strings.SplitN(message, " ", 2)
    command := parts[0]
    switch command {
    case "LOGIN":
        if len(parts) < 2 {
            return
        }
        username := parts[1]
        registerClient(username, addr)
        conn.WriteToUDP([]byte("Welcome "+username), addr)
    case "LOGOUT":
        if len(parts) < 2 {
            return
        }
        username := parts[1]
        removeClient(username)
        conn.WriteToUDP([]byte("Goodbye "+username), addr)
    case "MSG":
        if len(parts) < 2 {
            return
        }
        handleMessage(conn, addr, parts[1])
    }
}

func registerClient(name string, addr *net.UDPAddr) {
    clientsMu.Lock()
    defer clientsMu.Unlock()
    clients[name] = &Client{Name: name, Addr: addr}
    fmt.Println("Registered client:", name, addr)
}

func removeClient(name string) {
    clientsMu.Lock()
    defer clientsMu.Unlock()
    delete(clients, name)
    fmt.Println("Removed client:", name)
}
```

```go

func handleMessage(conn *net.UDPConn, senderAddr *net.UDPAddr, message string) {
    clientsMu.Lock()
    defer clientsMu.Unlock()

    var senderName string
    for name, client := range clients {
        if client.Addr.String() == senderAddr.String() {
            senderName = name
            break
        }
    }

    // Check for @<username> or @all command
    if strings.HasPrefix(message, "@") {
        parts := strings.SplitN(message, " ", 2)
        if len(parts) < 2 {
            return
        }
        target := parts[0][1:] // Remove "@" prefix
        msg := parts[1]

        // Private message to specific user
        if target != "all" {
            if client, ok := clients[target]; ok {
                privateMsg := fmt.Sprintf("Private from %s: %s", senderName, msg)
                conn.WriteToUDP([]byte(privateMsg), client.Addr)
            } else {
                conn.WriteToUDP([]byte("User "+target+" not found"), senderAddr)
            }
            return
        }

        // Broadcast message to all users
        broadcastMsg := fmt.Sprintf("Broadcast from %s: %s", senderName, msg)
        for _, client := range clients {
            if client.Addr.String() != senderAddr.String() { // Exclude sender
                conn.WriteToUDP([]byte(broadcastMsg), client.Addr)
            }
        }
    }
}

```

Result:

```
PS E:\IU\Senior\Net-Centric Lab\Lab4\client> go run .\udp_client.go
Enter your name: Toan2
Connected to UDP server at 127.0.0.1:8080
>>
Message: Welcome Toan2
>> @Toan1 Hello
>>
Message: Private from Toan1: Hi
>>
Message: Broadcast from Toan1: Hello everyone
>> STOP
Exiting UDP client!
PS E:\IU\Senior\Net-Centric Lab\Lab4\client>
```

```
PS E:\IU\Senior\Net-Centric Lab\Lab4\client> go run .\udp_client.go
Enter your name: Toan1
Connected to UDP server at 127.0.0.1:8080
>>
Message: Welcome Toan1
>>
Message: Private from Toan2: Hello
>> @Toan2 Hi
>> @all Hello everyone
>> @Toan1 Are u there?
>>
Message: Private from Toan1: Are u there?
>> @Toan2 Are u there?
>>
Message: User Toan2 not found
>> STOP
Exiting UDP client!
PS E:\IU\Senior\Net-Centric Lab\Lab4\client>
```

```
PS E:\IU\Senior\Net-Centric Lab\Lab4\server> go run .\udp_server.go
Server listening on: localhost:8080
Received:  LOGIN Toan1  from  127.0.0.1:55439
Registered client:  Toan1 127.0.0.1:55439
Received:  LOGIN Toan2  from  127.0.0.1:50721
Registered client:  Toan2 127.0.0.1:50721
Received:  MSG @Toan1 Hello  from  127.0.0.1:50721
Received:  MSG @Toan2 Hi  from  127.0.0.1:55439
Received:  MSG @all Hello everyone  from  127.0.0.1:55439
Received:  LOGOUT Toan2  from  127.0.0.1:50721
Removed client:  Toan2
Received:  MSG @Toan1 Are u there?  from  127.0.0.1:55439
Received:  MSG @Toan2 Are u there?  from  127.0.0.1:55439
Received:  LOGOUT Toan1  from  127.0.0.1:55439
Removed client:  Toan1
```