DISTRIBUTED SYSTEM PROGRAMMING

Bit torrent implementation

Group name id

Feven Tolla----- UGR/4493/12
Mekdes Kebede -----UGR/3127/12
Bereket Demissie ------UGR/0587/12
Dagemawi Yesuf----- UGR/3493/12
Christina Solomon -----UGR/8231/12
Elshaday Kassu -----ATR/1831/11

FEBRUARY 22, 2023

TO MR WONDIMAGEN D

What is BitTorrent Protocol

BitTorrent peer-to-peer (P2P) protocol finds users with files other users want and then downloads pieces of the files from those users simultaneously. Consequently, transmission rates are faster than with http and ftp, which both download files sequentially from only one source.
By permitting direct peer-to-peer communication between file hosts and downloaders, BitTorrent is intended to alleviate both of these drawbacks. Participants in BitTorrent communicate using a peer-to-peer protocol where each member serves as both a server and client. The ability for many downloaders to work together to download a file from a far smaller number of file uploaders may be BitTorrent's most crucial feature. In order to maintain fairness, BitTorrent features additional built-in mechanisms that encourage users to share previously downloaded data as uploaders and penalize downloaders by capping their download speeds if they don't.

Once connected, a BitTorrent client gets all the data it can gather by downloading little portions of the torrent's contents. The BitTorrent client can start sending data to other BitTorrent clients in the swarm once it has some data. In this manner, everyone who downloads a torrent also uploads it. Each person's download speed is increased by this. A central server is not overworked if 10,000 individuals download the same content at once. Instead, to keep the torrent moving quickly, each downloader shares their upload bandwidth with other downloaders.

It's important to note that BitTorrent clients never really download files from the tracker. The only way the tracker takes part in the torrent is by monitoring the BitTorrent clients linked to the

Procedures we used to implement

First we start by setting up the listener for the connection. Here we are making the one computer to listen to the incoming connections.

```go
func main() {
    // Listen for incoming connections
    // 10.5.226.133 under aau staff
    ln, err := net.Listen("tcp", "192.168.43.37:1234")
    if err != nil {
        fmt.Println("Error starting server: ", err)
        return
    }
    fmt.Println("Server started", ln.Addr())
    defer ln.Close()

    // go downloadTorrent()
```

Then to accept connection and to forward to the handle connection method else we will notify that accepting connection has failed.

```go
    for {
        // Accept incoming connections
        conn, err := ln.Accept()
        if err != nil {
            fmt.Println("Error accepting connection: ", err)
            continue
        }
        fmt.Println("Accepted connection from", conn.RemoteAddr())

        go handleConnection(conn)
    }
}
```

Notifying that peer is connected and ready for file transfer..

```go
func handleConnection(conn net.Conn) {
  defer conn.Close()
  // Get peer address
  peerAddr := conn.RemoteAddr().String()
  fmt.Println("Peer connected:", peerAddr)
  // Add peer to peer list
  peerList = append(peerList, peerAddr)
  fmt.Println("Peer connected:", peerAddr)
  defer func() {
    // Remove peer from peer list
```

Next we will provide get ways to the peer file using web requests like shown below

```go
func handleWebServer() {
  http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
    fmt.Fprintf(w, "Welcome to the P2P File Sharing Application\n")
    fmt.Fprintf(w, "You can use '/files' to see available files\n ")
    fmt.Fprintf(w, "You can use '/dowunload?file=filename'...the file with the specifed name will be downloaded if
    found\n ")
  })

  http.HandleFunc("/files", func(w http.ResponseWriter, r *http.Request) {
    // Display a list of available files
    files, _ := filepath.Glob("*.*")
    fmt.Fprintf(w, "Available Files: %s", files)
  })
  http.HandleFunc("/download", func(w http.ResponseWriter, r *http.Request) {
    // Get the requested file name
    fileName := r.URL.Query().Get("file")
    fileRequests <- fileName

    fmt.Fprintf(w, "Downloading file: %s", fileName)
  })
  http.ListenAndServe("192.168.43.37:8080", nil)
}
```

From the peer connected side(from the one which requests the file) first it will dial for connection available like..

```go
func main() {
  fmt.Println("Starting the application...")
  // 10.5.226.133...mine ip undet staff wifi
  // Connect to this socket
  conn, err := net.Dial("tcp", "192.168.43.117:1234") //22
  if err != nil {
    log.Fatal(err)
  }

  defer conn.Close()
```

Then we will proceed if connection succeeded, and then we will accept the filename of the file requested

```go
  var fileName string
  // var fileContent []byte
  _, err = fmt.Fscanln(conn, &fileName)
  if err != nil {
    fmt.Println("Error receiving file name:", err.Error())
    conn.Close()
    continue
  }
```

Then we will receive the file and the contents

```go
    fmt.Println("Receiving file:", fileName)
    fileContent := make([]byte, 1024)
    n, err := conn.Read(fileContent)
    if err != nil {
      fmt.Println("Error receiving file content:", err.Error())
    }

    fmt.Println(string(fileContent[:n]))
```

The we will write the file contents

```
_, err = newFile.Write(data)
if err != nil {
  fmt.Println("Error writing file content:", err.Error())
  conn.Close()
  continue
}
newFile.Close()
fmt.Println("File content written")
```

Below code will check if the file give is torrent or normal file

```
fileExtension := filepath.Ext(fileName)
fmt.Println("file ext", fileExtension)

if fileExtension == ".torrent" {
  fmt.Println("file with", fileExtension, "Torrent file selected")
  go downloadTorrent(fileName)
} else {
  fmt.Println("file with", fileExtension, "Regular file selected")
  // downloadRegularFile(fileName)
}
```

If it is torrent file, we use the method below to download the torrent file…..like this

```
func downloadTorrent(fileName string) {
  client, err := torrent.NewClient(nil)
  if err != nil {
    fmt.Println(err)
    os.Exit(1)
  }
  defer client.Close()

  // Parse the torrent file
  t, err := client.AddTorrentFromFile(fileName)

  if err != nil {
    fmt.Println(err)
    os.Exit(1)
  }
```