CPS706 – Content Distribution Network Project System Documentation

GROUP 30
Lucas Silvestri - 500630169
Liran Fraiman - 500622868
Elliot Tio – 500740959

The source code for the Content Distribution Network is written in Java, using built-in libraries (java.io, java.net, java.util). The main system architecture is divided into six java classes, each representing one functional part of the network. In addition, a src directory is included, which contains files that the network uses, each in a separate directory. Control messages are printed to system output. The details of the architecture is outlined below.

SRC directory

The src directory contains four sub-directories: ClientFiles, HerCDNFiles, HisCinemaFiles, and LocalDnsFiles.

ClientFiles:

This is where files requested by the client will be stored – such as the index of www.hiscinema.com and any media files requested from HerCDN.

HerCDNFiles:

Contains media files stored in HerCDN such as gifs, mp4s, pngs, and txts that can be requested and sent to client through the network. As well, the records for HerCDNAuthoritativeDns are kept in this directory.

HisCinemaFiles:

Contains index for www.hiscinema.com and records for HisCinemaAuthoritativeDns.

LocalDnsFiles:

Contains records for Local DNS in the form of a txt file.

Client.java

Client.java contains the source code for the client application as well as client UI to initiate a request to the network. Client stores ports and IP addresses in static final variables, which can be altered depending on port choice and IP addresses used.

The main method of the Client class is runClient(), which initiates a request to contact www.hiscinema.com and utilizes a LinkedList data structure to collect and present content URL to the user for them to choose. The LinkedList is constructed by the method getIndexTCP().

getIndexTCP() uses a Java Socket object to initiate a TCP connection with HisCinemaServer.java, and uses a PrintWriter object to produce output at the server. This is how HTTP messages are implemented between the dummy web server and client. A new file is accepted into ClientFiles which is index.html from the web server. The HTML file is then parsed with htmlParserHTTPMessage() to obtain the HTTP response, and then again with htmlParser() to extract the content URLs into a LinkedList.

A BufferedReader() object is used to accept input from a client, from a range of 1-4 depending on which URL the client chooses. Once the client makes a choice, the URL is taken from the LinkedList using the get() method, and a new DatagramSocket object is created to send a UDP request into the network. Two byte arrays are instantiated, named sendData and recieveData, to contain the byte data to be sent and received, respectively. A DatagramPacket object is created which sends the URL request to the Local DNS IP address and port. Similarly, a second DatagramPacket object is created to accept the reply from the network. The expected return value is then stored into IPADDRESS_HER, and the socket is closed.

The method getCDNFileTCP() is then called to initiate a TCP connection to HerCDNServer to request the content file, given the IP address retrieved from the network, using a Java Socket object. The file is then received with an InputStream and stored into the ClientFiles directory.

LocalDns.java

LocalDns.java is a Java implementation of a local DNS accepting UDP packets. It stores port numbers in static final variables. A DatagramSocket object is used to initialize a connection socket, bound to PORT_LOCAL_DNS. A DatagramPacket object is created to accept UDP queries and packets, which is parsed with parseQuery() which returns a trimmed String representation of the initial query.

Based on the parsed query, LocalDns consults its records stored in localrecords.txt and determines which authoritative DNS to reroute the query to. When the return query is an IP address, the packet is returned to Client.

HisCinemaServer.java

HisCinemaServer.java is a dummy web server running on TCP using a ServerSocket object. Multithreading is implemented by extending Java Thread, and by creating a new ServerThread object every time a TCP connection is accepted. BufferedReader objects are used to parse client requests and to return files to the client, as well as returning HTTP replies.

HisCinemaAuthoritativeDns.java

HisCinemaAuthoritativeDns.java implements the authoritative DNS of HisCinema, accepting UDP packets from LocalDns, and parsing its records file with a Scanner object to return the proper record reply.

HerCDNServer.java

HerCDNServer.java is a dummy web server running on TCP using a ServerSocket object. Multithreading is implemented by implementing Java Runnable, and by creating a new Thread and new HerCDNServer object every time a TCP connection is accepted. HerCDNServer uses BufferedReaders to accept queries and requests, as well as returning files and HTTP replies to the client.

HerCDNAuthoritativeDns.java

HerCDNAuthoritativeDns.java implements the authoritative DNS of HerCDN, accepting UDP packets from LocalDns, and parsing its records file with a Scanner object to return the proper record reply. In addition, HerCDNAuthoritativeDns uses a second scanner to return the proper “A” type record.