```java
import java.util.Random;
import java.net.*;
import java.io. *;

public class LoadBalancer extends Thread {
   static String hosts[] = {"localhost", "localhost"};
   static int ports[] = {8081,8082};
   static int nbHosts = 2;
   static Random rand = new Random();
   Socket sClient;

   public LoadBalancer(Socket s) {
      this.sClient = s;
   }

   public void run() {
      try {
         System.out.println("Debut Load Blancing");

         // Recuperer la requete du Client
         OutputStream os = sClient.getOutputStream();
         InputStream is = sClient.getInputStream();
         byte[] rq = new byte[1024];
         is.read(rq);

         // Transmettre la requete vers une Comanche
         int i = rand.nextInt(nbHosts);
         Socket sComanche = new Socket(hosts[i], ports[i]);
         OutputStream osc = sComanche.getOutputStream();
         InputStream isc = sComanche.getInputStream();
         osc.write(rq);

         // Recuperer la reponse de la Comanche et la transmettre au client
         byte[] rep = new byte[1024];
         isc.read(rep);
         os.write(rep);

         os.close();
         is.close();
         osc.close();
         isc.close();
         sClient.close();
         sComanche.close();

         System.out.println("Fin Load Blancing");
      } catch (Exception e) {
         e.printStackTrace();
      }
   }

   public static void main(String args[]) {

       try {
          ServerSocket ss = new ServerSocket(8080);
          while (true) {
             Socket s = ss.accept();
             System.out.println("Demande de connection");

             LoadBalancer t = new LoadBalancer(s);
             t.start();
          }
       } catch (Exception e) {
          e.printStackTrace();
       }
   }
}
```

```java
import java.net.*;
import java.io.*;

public class Comanche implements Runnable {
    private Socket s;
    public Comanche (Socket s) { this.s = s; }

    public static void main (String[] args) throws IOException {
        ServerSocket s = new ServerSocket(Integer.parseInt(args[0]));
        while (true) { new Thread(new Comanche(s.accept())).start(); }
    }

    public void run () {
        try {
            InputStreamReader in = new InputStreamReader(s.getInputStream());
            PrintStream out = new PrintStream(s.getOutputStream());
            String rq = new LineNumberReader(in).readLine();
            System.out.println(rq);
            if (rq.startsWith("GET ")) {
                File f = new File(rq.substring(5, rq.indexOf(' ', 4)));
                if (f.exists() && !f.isDirectory()) {
                    InputStream is = new FileInputStream(f);
                    byte[] data = new byte[is.available()];
                    is.read(data);
                    is.close();
                    String s = new String(data);
                    out.print("HTTP/1.0 200 OK\n\n"+s);
                } else {
                    out.print("HTTP/1.0 404 Not Found\n\n <html>Document not found.</html>");
                }
            }
            out.close();
            s.close();
        } catch (IOException ex) {
            ex.printStackTrace();
        }
    }
}


    public HdfsClient() {
        try {
            Socket socket = new Socket(nameNode, portsNameNode);
            OutputStream os = socket.getOutputStream();
            InputStream is = socket.getInputStream();

        ObjectOutputStream oos = new ObjectOutputStream (os);
            ObjectInputStream ois = new ObjectInputStream (is);

            oos.writeObject(CommandeNameNode.NM_ADD);
            oos.writeObject("tmp");
            oos.writeObject("KV");
            boolean endAdd = (boolean) ois.readObject();

            oos.close();
            os.close();
            ois.close();
            is.close();
            socket.close();
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }




   public static void HdfsDelete(String hdfsFname) {
        try {
            Socket socketN = new Socket(nameNode, portsNameNode);
            OutputStream osN = socketN.getOutputStream();
            InputStream isN = socketN.getInputStream();
            ObjectOutputStream oosN = new ObjectOutputStream (osN);
            ObjectInputStream oisN = new ObjectInputStream (isN);

            oosN.writeObject(CommandeNameNode.NM_CONTAINS);
            oosN.writeObject(hdfsFname);

            if ((boolean) oisN.readObject()) {
                boolean endContains = (boolean) oisN.readObject();
                for (int i = 0; i < nbNodes; i++) {
                    oosN.close();
                    osN.close();
                    oisN.close();
                    isN.close();
                    socketN.close();
                    // Envoi de l'ordre de supression au serveur
                    Socket socket = new Socket(nodes[i], ports[i]);
                    OutputStream os = socket.getOutputStream();
                    InputStream is = socket.getInputStream();

                    ObjectOutputStream oos = new ObjectOutputStream (os);
                    ObjectInputStream ois = new ObjectInputStream (is);
                    oos.writeObject(CommandeCmd.CMD_DELETE);
                    oos.writeObject(hdfsFname);

                    boolean answer = (boolean) ois.readObject();
                    if (answer) { //suppression réussie

                    } else { //échec de la suppression
                        System.out.print("Le fichier correspondant n'a pas pu être supprimé sur le serv
eur ");

                        System.out.print(i);

                        oos.close();
                        os.close();
                        ois.close();
                        is.close();
                        socket.close();

                        throw(new Exception());
                    }

                    oos.close();
                    os.close();
                    ois.close();
                    is.close();
                    socket.close();

                    socketN = new Socket(nameNode, portsNameNode);
                    osN = socketN.getOutputStream();
                    isN = socketN.getInputStream();
                    oosN = new ObjectOutputStream (osN);
                    oisN = new ObjectInputStream (isN);

                    oosN.writeObject(CommandeNameNode.NM_DELETE);
                    oosN.writeObject(hdfsFname);
                    boolean endDelete = (boolean) oisN.readObject();

                    oosN.close();
                    osN.close();
                    oisN.close();
                    isN.close();
                    socketN.close();
                }
            } else {
                System.out.println("Le fichier que vous essayez de supprimer n'existe pas");
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
```