 未标题-1

**软件体系结构**

**实验报告**

**实验题目: C/S架构的在线聊天客户端体系结构设计**

**C/S架构的在线聊天客户端体系结构设计**

**院系名称： 信息科学与工程学院 专业班级：软件1702**

**学生姓名： 刘文博 学 号： 201716040224**

**指导教师： 刘 灿 日 期： 2020/3/27**

**一．相关知识**

客户端-服务器模式(Client–server model)简称C/S结构，是一种网络架构，它把客户端 (Client) 与服务器 (Server) 区分开来。每一个客户端软件的实例都可以向一个服务器或应用程序服务器发出请求。

服务器进程:

它接受一个来自客户端的请求的过程。获得来自客户端的请求后会处理所需的收集所需的信息，将其发送到请求客户端。一旦这样做完成后，就又变成准备为另一个客户端。服务器进程始终等待准备用于处理传入请求。

实例: Web服务器一直等待来自互联网浏览器的请求，并尽快得到任何请求从浏览器，它拿起一个请求的HTML页面，并把它发送回该浏览器。

注意，客户端需要知道的存在服务器的地址，但是服务器并不需要在建立的连接之前知道客户端的地址。一旦建立连接后，双方都可以发送和接收信息。

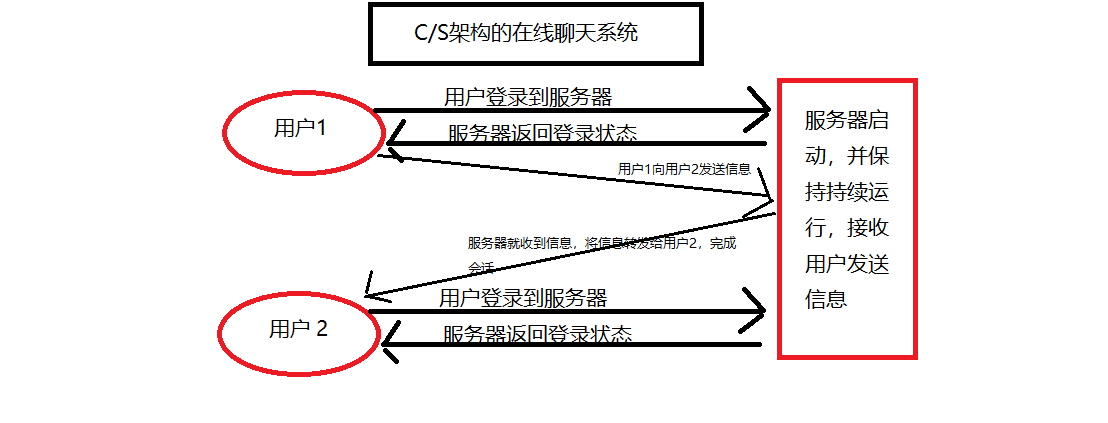
客户端进程:

这是一个过程，这通常使得信息请求。得到回应后，这一过程可能会终止或可能会做一些其他的处理。

例如: 互联网浏览器作为一个客户端应用程序，Web服务器发送一个请求到得到一个HTML网页。

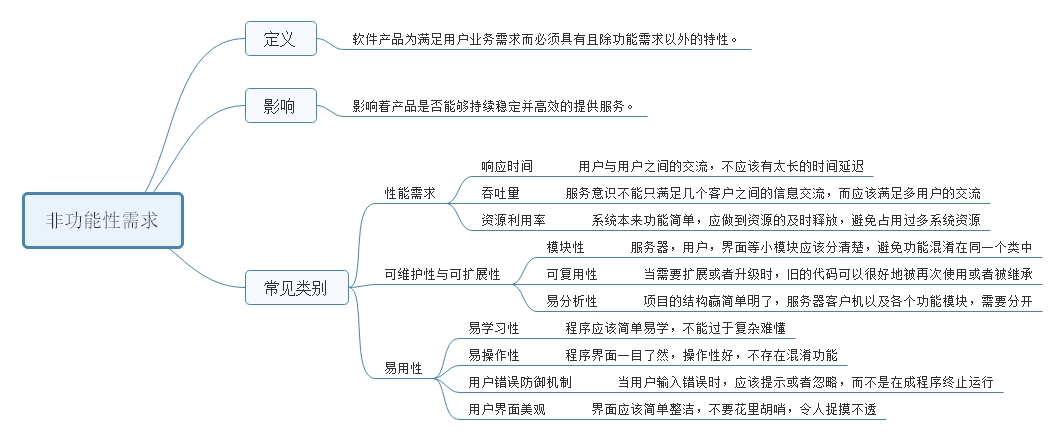
**二. 需求描述**

**功能性需求**

****

在线聊天系统需要服务器一直保持运行，时刻接受用户发起的登录以及用户间的会话信息，并正确的将用户的信息准确无误的转发出去。客户机启动，选择需要启动的端口，与服务器进行连接，并且可以向在线的用户发送信息，进行实时通信。

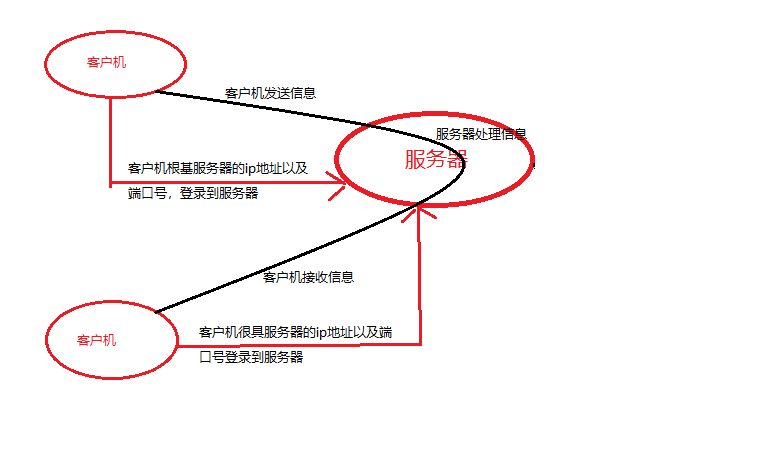
**非功能性需求**



在我看来，该系统最重要的是响应时间的问题，在线聊天系统，延迟不易过长，延迟过长会给用户极差的用户体验；还有资源利用率问题，简单的聊天系统，不能占用过多的系统资源，造成资源紧张；还有安全性问题，该系统紧紧适用于局域网内，外网访问不到..没有考虑安全问题；对于操作性与界面，该程序功能简单，不需要花里胡哨的界面，做到对话，聊天功能齐全就行了。

**三. 系统结构**

最初的设想是：启动服务器，保持服务器一直运行，然后再启动客户机，客户机连接上服务器，就可以进行在线聊天了，但是发现此时，系统的IP地址以及端口号都未被指定，所以进行了一个管理员登录的过程，由管理员指定服务器的端口号IP地址等信息，然后启动服务器，并可以进行对用户的强制下线操作



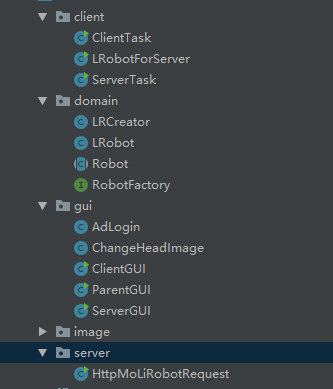
服务器直接启动

****

**由管理员指定端口启动服务器**

**四．架构质量属性**

### 1.代码结构应该清晰，从代码结构看出系统结构



从图中可以看出，该系统由客户端，实体类，GUI界面，以及服务类组成，结构清晰。

### 2.可扩展性

代码中运用了继承操作，实现了对统一代码，界面的复用，当需要增加实体或者功能的时候，同样继承父类ParentGUI就可以实现对功能的扩充。

|  |
| --- |
| import java.awt.\*;  import javax.swing.\*;  public class ParentGUI extends JPanel {  protected int jlab\_size;  protected JLabel jlbs[]; //服务器IP地址、服务器端口号标签  protected JTextField jtfs[];  protected JButton jb[]; //启动服务器、结束按钮  protected JLabel chatmess; //聊天信息标签  protected JTextArea jtchat; //聊天信息框  protected JPanel northPanel;  protected JPanel centerPanel;  protected JPanel southPanel;  public ParentGUI(int size){  jlab\_size=size;  setLayout(new BorderLayout());  jlbs=new JLabel[jlab\_size];  jtfs=new JTextField[jlab\_size];  jb=new JButton[2];  for(int i=0;i<jlbs.length;i++){  jlbs[i]=new JLabel("标签"+i);  jtfs[i]=new JTextField(12);    }  for(int i=0;i<2;i++)  jb[i]=new JButton("按钮"+i);  jlbs[0].setText("服务器IP ");  jlbs[1].setText("服务器端口号");  northPanel=new JPanel();  for(int j=0;j<jlbs.length;j++){  northPanel.add(jlbs[j]);  northPanel.add(jtfs[j]);  }  for(int j=0;j<2;j++)  northPanel.add(jb[j]);  chatmess=new JLabel("聊天信息");  jtchat=new JTextArea(30,35);  jtchat.setLineWrap(true);  jtchat.setEditable(false);    JScrollPane jscro = new JScrollPane(jtchat);  jscro.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL\_SCROLLBAR\_AS\_NEEDED);  jscro.setVerticalScrollBarPolicy(JScrollPane.VERTICAL\_SCROLLBAR\_AS\_NEEDED);  centerPanel=new JPanel();  southPanel=new JPanel();  JPanel innerPanel=new JPanel();  innerPanel.setLayout(new BorderLayout());  innerPanel.add(chatmess,BorderLayout.NORTH);  innerPanel.add(jscro,BorderLayout.CENTER);  centerPanel.add(innerPanel);    add(northPanel,BorderLayout.NORTH);  add(centerPanel,BorderLayout.WEST);  add(southPanel,BorderLayout.SOUTH);  } |

|  |
| --- |
| import java.awt.\*;  import javax.swing.\*;  public class ClientGUI extends ParentGUI {  //private JLabel lblUser ;  // private JTextField txtUser;  protected JTextField inputmess;  protected JLabel chatlist;  protected JComboBox chatobj;  protected JButton sent;  protected JButton clear;  protected JButton refresh;  protected JButton headimg;  protected JLabel username\_head;  protected ImageIcon headimage;  protected JComboBox<String> OnlineStatus;  public ClientGUI(){  super(3);  headimage = new ImageIcon("src//image//HeadImg.png");  headimage.setImage(headimage.getImage().getScaledInstance(80, 80, Image.SCALE\_DEFAULT));  //headimage.setImage(image);  jb[0].setText("连接 ");  jb[0].setForeground(Color.red);  jb[0].setBorderPainted(false);  jb[1].setText("断开");  jb[1].setForeground(Color.red);  jb[1].setBorderPainted(false);  jlbs[2].setText("登录名称");  jtchat.setColumns(50);    OnlineStatus = new JComboBox<String>();  OnlineStatus.addItem("在线");  OnlineStatus.addItem("请勿打扰");  OnlineStatus.addItem("隐身");  OnlineStatus.addItem("离线");    username\_head = new JLabel("null");  inputmess=new JTextField(30);  chatlist=new JLabel("选择聊天对象");  chatobj=new JComboBox();  refresh=new JButton("刷新");  sent=new JButton("发送");  clear=new JButton("清除");  headimg = new JButton("");  headimg.setBorderPainted(false);  headimg.setBackground(Color.white);  headimg.setIcon(headimage);  centerPanel.add(headimg);  centerPanel.add(username\_head);  centerPanel.add(OnlineStatus);  southPanel.add(inputmess);  southPanel.add(chatlist);  southPanel.add(chatobj);  southPanel.add(refresh);  southPanel.add(sent);  southPanel.add(clear);  } |

### 3.建造者模式

运用建造者模式，外界可以从直接创建具体产品对象的尴尬局面中摆脱出来。外界与具体类隔离开来，偶合性低。明确区分了各自的职责和权力，有利于整个软件体系结构的优化。

|  |
| --- |
| package domain;  public interface RobotFactory {  public Robot createRobot();  }  //工厂模式 |

|  |
| --- |
| package domain;  //构造者  public class LRCreator implements RobotFactory {  public LRCreator() {  // TODO Auto-generated constructor stub  }  @Override  public LRobot createRobot() {  // TODO Auto-generated method stub  return new LRobot("茉莉");  }  } |

**五．代码**

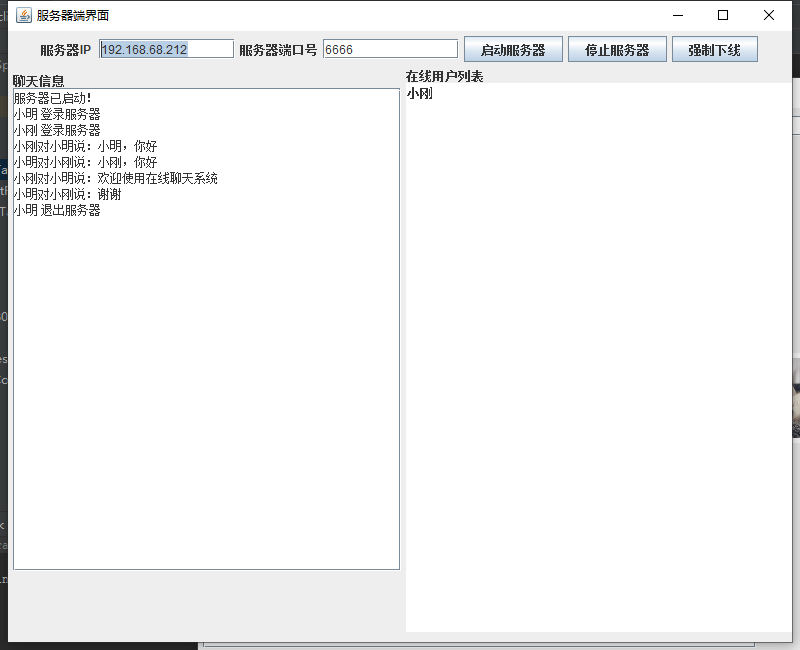
服务器端代码：

|  |
| --- |
| public class ServerTask extends ServerGUI implements ActionListener{  protected ServerSocket ss;  protected Socket currs;  protected IOHandler ioh;  protected ListThread lst;  protected int usrport;  List<String> nameL = new ArrayList();  List<DataOutputStream> outL = new ArrayList();  List<DataInputStream> inL = new ArrayList();  List<Integer> UPL = new ArrayList<>();  //userportlist    public ServerTask(){  try{  jtfs[0].setText(InetAddress.getLocalHost().getHostAddress());  }catch(IOException e){}  jb[0].addActionListener(this);  jb[1].addActionListener(this);  ForceOffline.addActionListener(this);  jluser.getSelectedIndex();  nameL.add("3&&");  }    @Override  public void actionPerformed(ActionEvent e){  if(e.getSource()==jb[0])  doStart();  if(e.getSource()==jb[1])  doExit();  if(e.getSource()==ForceOffline)  doOffLine();    }    public void doStart(){  String port=jtfs[1].getText();  try {  if(port == null || port.equals("")){  JOptionPane.showMessageDialog(null, "请输入端口号");  return; }  else  {  ss = new ServerSocket(Integer.parseInt(port));  jtchat.setText(null);  jtchat.append("服务器已启动！\n");  new ListThread().start();  }  }catch(IOException e){  jtchat.append("服务器启动失败！\n");  }  }      public boolean doOffLine()  {  List<String> temp = new ArrayList();  String flag = "4&&";  String username = jluser.getSelectedValue().toString();  // System.out.println(username);  int n = nameL.indexOf(username);  temp.add(flag);  temp.add(username);  try {  nameL.remove(username);  dm.removeElement(username);  //发送用户信息  JSONArray ja = new JSONArray(temp);  for(int i=0;i<outL.size();i++){  outL.get(i).writeUTF(ja.toString());  }  jtchat.append(username + " 被踢出服务器\n");  } catch (IOException e) {  // TODO Auto-generated catch block  JOptionPane.showMessageDialog(null, "请确认已有用户连接到服务端！");  e.printStackTrace();  }    return false;  }      class ListThread extends Thread{  public void run(){  currs=null;  while(true){  try{  currs=ss.accept();  // System.out.println(currs.getRemoteSocketAddress());  DataInputStream dis = new DataInputStream(currs.getInputStream());  DataOutputStream dos = new DataOutputStream(currs.getOutputStream());  inL.add(dis);  outL.add(dos);  new IOHandler(dis).start();  }catch(IOException e){}  }  }  }  class IOHandler extends Thread{  private DataInputStream s;  public IOHandler(DataInputStream s){  this.s=s;  }  public void run(){  try{  while(true) {  String mess = s.readUTF();  //发送姓名  if(mess.startsWith("1&&")) {  String username = mess.substring(3);  nameL.add(username);  dm.addElement(username);  jtchat.append(username + " 登录服务器\n");  //发送用户信息  JSONArray ja = new JSONArray(nameL);  for(int i=0;i<outL.size();i++){  outL.get(i).writeUTF(ja.toString());  }  }  else if(mess.startsWith("2&&")) {  String username = mess.substring(3);  dm.removeElement(username);  int n = nameL.indexOf(username);  nameL.remove(n);  inL.remove(n-1);  outL.remove(n-1);  jtchat.append(username + " 退出服务器\n");    List<String> list = new ArrayList();  list.add("2&&");  list.add(username);  JSONArray ja = new JSONArray(list);  for(int i=0;i<outL.size();i++){  outL.get(i).writeUTF(ja.toString());  }  }  //处理群发消息  //处理一对一聊天信息  else {  JSONArray ja = new JSONArray(mess);  String fromName = ja.getString(0);  String content = ja.getString(1);  String toName = ja.getString(2);  if(toName.equals("all")) {  jtchat.append(fromName + "对所有人说："+content+"\n");  for(int i=0;i<outL.size();i++) {  DataOutputStream toOut = outL.get(i);  toOut.writeUTF(mess);  }  }else {  jtchat.append(fromName + "对"+toName+"说："+content+"\n");  int n = nameL.indexOf(toName);  DataOutputStream toOut = outL.get(n-1);  toOut.writeUTF(mess);  }  }  }  }catch(IOException e) {  e.printStackTrace();  }  }  }  public void doExit(){  jtchat.setText(null);  jtchat.append("服务器已关闭！\n");  dm.removeAllElements();  nameL.removeAll(nameL);    try{  ss.close();  }catch(IOException e){}  }  public static void main(String args[]){  //创建登陆窗口  AdLogin adlog = new AdLogin();    while(!adlog.getFlag()) {  try {  Thread.sleep(1000);  } catch (InterruptedException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  System.out.println(adlog.getFlag());  if(adlog.getFlag()) {  adlog.setVisible(false);  JFrame app=new JFrame("服务器端界面");  app.getContentPane().add(new ServerTask());  app.setSize(800,650);  app.setVisible(true);  app.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  app.setLocationRelativeTo(adlog);  }  }  } |

客户端代码：

|  |
| --- |
| public class ClientTask extends ClientGUI implements ActionListener{  public Socket cs;  protected DataOutputStream toServer;  protected DataInputStream fromServer;  protected String username;  protected ChangeHeadImage CHI = new ChangeHeadImage(headimg);  public ClientTask() {  // TODO Auto-generated constructor stub  jb[0].addActionListener(this);  jb[1].addActionListener(this);  sent.addActionListener(this);  clear.addActionListener(this);  refresh.addActionListener(this);  headimg.addActionListener(CHI);  OnlineStatus.setSelectedIndex(3);  OnlineStatus.addItemListener(new ItemListener() {    @Override  public void itemStateChanged(ItemEvent e) {  // TODO Auto-generated method stub  if (e.getStateChange() == ItemEvent.SELECTED)  {  if(OnlineStatus.getSelectedIndex()==0) {  if(cs==null)  doConn();  }  //if(OnlineStatus.getSelectedIndex()==1) doConn();  //if(OnlineStatus.getSelectedIndex()==2) doConn();  if(OnlineStatus.getSelectedIndex()==3) {  if(cs!=null)  doExit();  }  }  }  });  }  public void actionPerformed(ActionEvent e){  if(e.getSource()==sent)  doSent();  else if(e.getSource()==clear)  doClear();  else if(e.getSource()==jb[0])  doConn();  else if(e.getSource()==jb[1])  {  OnlineStatus.setSelectedIndex(3);  doExit();  }  else if(e.getSource()==refresh)  doRefersh();    }  private void doRefersh() {  // TODO Auto-generated method stub  try {  toServer.writeUTF("3&&");  } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }      }  public void doSent() {  // TODO Auto-generated method stub  if(cs==null) JOptionPane.showMessageDialog(null, "服务端未连接！！");  //判断未连接，不向服务端发送json数据  else {  String message = inputmess.getText();  jtchat.append(username +": @"+chatobj.getSelectedItem()+" "+message+"\n");  // toServer = new DataOutputStream(cs.getOutputStream());  // toServer.writeUTF("2&&"+message);  // System.out.println(cs);  String toServerUser = (String)chatobj.getSelectedItem();  List<String> chatMsg = new ArrayList();  chatMsg.add(username);  chatMsg.add(message);  chatMsg.add(toServerUser);  JSONArray ja = new JSONArray(chatMsg);  try {  toServer.writeUTF(ja.toString());  } catch (IOException e) {  // toServerDO AutoServer-generated catch block  e.printStackTrace();  }  }  }  public void doExit() {  // TODO Auto-generated method stub  username\_head.setForeground(Color.GRAY);  jtchat.append("Disconnect Succeed~\n");  try {  toServer.writeUTF("2&&"+username);  cs.close();  cs = null;  OnlineStatus.setSelectedIndex(3);  } catch (IOException e) {  // TODO Auto-generated catch block  e.printStackTrace();  }  }    public void doConn() {  // TODO Auto-generated method stub  if(cs==null) {  username = jtfs[2].getText();  String ip = jtfs[0].getText();  String port = jtfs[1].getText();  if(port == null || port.equals("")||username == null||username.equals("")){  JOptionPane.showMessageDialog(null, "请填写IP/端口号/用户名");  OnlineStatus.setSelectedIndex(3);  return;}  else  {  try {  cs= new Socket(ip, Integer.parseInt(port));  jtchat.append("Connected Succeed~\n");  fromServer = new DataInputStream(cs.getInputStream());  toServer = new DataOutputStream(cs.getOutputStream());  toServer.writeUTF("1&&"+username);  jtchat.append(username+"登陆成功\n");  username\_head.setText(username);  username\_head.setForeground(Color.green);  OnlineStatus.setSelectedIndex(0);  new getMsgThread().start();  System.out.println(cs.getLocalPort());  }  catch (IOException e) {  jtchat.append("Connected Failed~\n");  // TODO Auto-generated catch block  e.printStackTrace();  }  }  }  else  {  JOptionPane.showMessageDialog(null, "请尝试断开链接后重新连接！！！");  }  }  public void doClear() {  // TODO Auto-generated method stub  inputmess.setText("");  }    class getMsgThread extends Thread{  public void run() {  while(true) {  try {  String jaStr = fromServer.readUTF();  JSONArray ja = new JSONArray(jaStr);  String flat = ja.getString(0);  if(flat.equals("3&&")){  chatobj.removeAllItems();  for(int i=1;i<ja.length();i++) {  String s = ja.getString(i);  if(s.equals(username)) continue;  chatobj.addItem(s);  }  chatobj.addItem("all");  }  else if(flat.equals("2&&")){  String s = ja.getString(1);  chatobj.removeItem(s);  }  else if(flat.equals("4&&"))  {  doExit();  jtchat.append("您已被踢出服务器！");  }  else {  // 发送信息用户的name不能是"3&&"，"2&&"  String fromServerName = ja.getString(0);  String content = ja.getString(1);  JSONObject messjson;  if(content.startsWith("{")){  messjson = new JSONObject(content);  content= messjson.getString("content");  }  if(!fromServerName.equals(username))  jtchat.append(fromServerName+"对我说："+content+"\n");  }  }catch(IOException e) {  }  }  }  } |

**六．结论**





优势

1.在大多数情况下，客户机服务器体系结构允许的角色和责任的计算系统是其中几个已知只有通过网络互相独立的计算机分发。这将创建一个额外的优势，这种体系结构：提高维护的简便性。例如，它可以更换，维修，升级，甚至迁移服务器，同时它的客户都不知情，并保持该更改的影响。

2.所有数据都存储在服务器上，通常比大多数客户更大的安全控制。服务器可以更好地控制访问和资源，以保证只有那些具有适当权限的用户可以访问和更改数据。

3.由于数据的集中存储，对数据的更新是更容易管理相比，一个P2P模式。在后者，数据更新可能需要分发和应用到每个网络中的对等，既费时又容易出错，因为可以有成千上万甚至数百万的同龄人。

4.许多成熟的客户端服务器技术已经可以方便的目的是确保安全，用户界面友好，易用性。

5.具有不同功能的多个不同的客户它的功能。

劣势

1.作为客户端同时请求数给定的服务器的增加，服务器可以成为重载。对照到一个P2P网络，在其总带宽实际上增加为节点的添加，由于P2P网络的总带宽可以大致计算作为该网络中每个节点的带宽的总和。

2.客户机服务器模式缺乏一个良好的P2P网络的鲁棒性。在客户端服务器，如果一个重要的服务器失败，客户的要求不能得到满足。在P2P网络，资源通常分布在许多节点。即使一个或多个节点出发，放弃一个下载文件，例如，剩下的节点应该还是有必要来完成下载的数据。