ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

Кафедра ВС

Самостоятельная работа

Отказоустойчивая сетевая игра «Виселица»

ВЫПОЛНИЛИ:

студенты гр. ВМ-88

Егоров И.Ю.

Барановский В.А.

ПРОВЕРИЛ:

Майданов Ю.С.

Новосибирск 2011

**Цель работы:**

Разработать сервер-клиент приложение учитывая некоторые аспекты реализации:

1. При отказе клиента сервер должен продолжить функционирование, сообщив второму пользователю об уходе первого.  
2. При отказе сервера клиенты должны это распознать и вывести сообщение, после этого завершив работу.  
3. Во всех процессах передачи данных по сети реализовать проверку протокола и контроль объема переданных данных. При некорректных действиях клиента сервер должен сделать пометку в логе и отключить данного клиента.  
4. Необходимо реализовать альтернативный сервер управления с протоколом синхронизации между основным и альтернативным серверами. При отказе основного сервера клиенты должны переключится на альтернативный сервер сохранив состояние.

**Листинг программы:**

1. При отказе клиента сервер должен продолжить функционирование, сообщив второму пользователю об уходе первого.

**protocol\_server.py:**

try:

text = sock.recv(128)

parse = text.split("@")

except socket.error, detail:

userscount -= 1

**logger.info(name + " has been disconnected!")**

del users[sock]

new = None

**sendmsg(ANSWER\_USERCOUNT + "\_%s@" % (userscount), sock)**

break

if not text:

**logger.info(name + " has been disconnected!") #запись события отключения в лог**

new = None

sleep(1)

userscount -= 1

sock.close()

del users[sock]

**sendmsg(ANSWER\_USERCOUNT + "\_%s@" % (userscount), sock)**

1. При отказе сервера клиенты должны это распознать и вывести сообщение, после этого завершив работу.

**client.py:**

try:

if self.main:

self.sock.connect((CLI\_MAIN\_HOST, CLI\_MAIN\_PORT))

else:

sleep(3)

logger.info("Connecting to alternative server...")

self.sock.connect((CLI\_ALT\_HOST, CLI\_ALT\_PORT))

except socket.error:

if self.main:

try:

sleep(3)

**logger.info("Connecting to alternative server...") #если подключение к основному серверу не удалось**

self.sock.connect((CLI\_ALT\_HOST, CLI\_ALT\_PORT))

except socket.error:

**logger.critical("Connecting to alternative server failed...")**  **#если подключение к альт. серверу не удалось**

**logger.critical("Servers not work!") #вывод ошибки**

sleep(5)

sys.exit(0)

else:

logger.critical("Servers not work!") **#если произошел разрыв соединения с альт. сервером**

try:

self.sock.send(QUERY\_CONN)

self.query\_result = self.sock.recv(4)

if (self.query\_result == CONN\_ALLOW):

logger.info("Connect allowed!")

self.connected = True

else:

logger.error(self.query\_result)

except socket.error, detail:

logger.error(detail)

if self.main:

cli.connect(False)

else:

logger.critical("Alternative server down!") **#если произошел разрыв соединения с альт. сервером**

sys.exit(0)

1. Во всех процессах передачи данных по сети реализовать проверку протокола и контроль объема переданных данных. При некорректных действиях клиента сервер должен сделать пометку в логе и отключить данного клиента.

**protocol\_server.py:**

**if item[0] == "#": первый символ пакета данных — символ «#», иначе — отключение пользователя**

if text[0:4] == QUERY\_CONN:

sock.send(CONN\_ALLOW + "@")

queue\_start.append(sock)

if main\_server:

word = gallows.generate()

usersword = "\*" \* len(word)

s.changed = True

logger.info("\nSecret word generated! [%s]. \nFor users: %s\n" % (word, usersword))

else:

gallows.secret = pinger.secret

gallows.attempts = pinger.attempts

gallows.newuword = pinger.userword

usersword = pinger.userword

gallows.newuword = pinger.userword

gallows.used\_letters = pinger.used\_letters

s.changed = False

sendmsg(PACKET\_USERWORD + "\_%s\_%s@" % (usersword, gallows.attempts), sock)

sendmsg(ANSWER\_USERCOUNT + "\_%s@" % (userscount), sock)

break

lst = item.split("\_")

logger.debug(lst)

if lst[0] == QUERY\_USERCOUNT:

sendmsg(ANSWER\_USERCOUNT + "\_%s@" % (userscount), sock)

logger.info("Userscount is %s" % (userscount))

**if lst[0] == PACKET\_LETTER:**

**if len(lst[1])== 1 and re.match("^[a-z]\*$", lst[1]): #если в пакете PACKET\_LETTER длина данных == 1 и символ входит в множество a—z, то всё в порядке. Иначе — отключение пользователя.**

letter = lst[1]

result = gallows.getletter(usersword, strip(letter))

logger.info("S: %s UW: %s Text: %s Letter: %s Result: %s" % (gallows.secret, usersword, text, letter, result))

usersword = result[1]

s.changed = True

if (gallows.attempts == 0):

sendmsg(WORD\_FAIL + "\_%s\_%s@" % (name, word), sock)

restart = True

else:

if (result[0] != 0):

if (gallows.attempts != 0):

if (result[0] > 0):

sendmsg(LETTER\_WIN + "\_%s\_%s\_%s\_%s@" % (name, letter, usersword, gallows.attempts), sock)

s.changed = True

if (result[0] < 0) or (gallows.attempts == 0):

if (result[0] == -1):

sendmsg(WORD\_WIN + "\_%s\_%s@" % (name, usersword), sock)

s.changed = True

restart = True

if (result[0] == -2):

sendmsg(LETTER\_ALREADY + "\_%s@" % letter, sock)

else:

sendmsg(LETTER\_FAIL + "\_%s\_%s\_%s\_%s@" % (name, letter, usersword, gallows.attempts), sock)

gallows.attempts -= 1

s.changed = True

break

if lst[0] == QUERY\_USERWORD:

sendmsg(PACKET\_USERWORD + "\_%s\_%s@" % (usersword, gallows.attempts), sock)

restart = True

logger.info("Userscount is %s" % (userscount))

**else: #отключение пользователя, если пришел пакет с неверным заголовком**

**kick = True**

**else: #отключение пользователя, если первый символ пакета ≠ «#»**

**kick = True**

if (kick):

logger.info("Client %s kicked!" % name)

sock.send(CONN\_CLOSE\_KICK)

userscount -= 1

del users[sock]

new = None

sendmsg(ANSWER\_USERCOUNT + "\_%s@" % (userscount), sock)

sock.close()

1. Необходимо реализовать альтернативный сервер управления с протоколом синхронизации между основным и альтернативным серверами. При отказе основного сервера клиенты должны переключится на альтернативный сервер сохранив состояние.

Реализовано через отдельный классы:

**protocol\_server.py**

**class Pinger(Thread): #«синхронизирующий», основной сервер**

def \_\_init\_\_(self):

Thread.\_\_init\_\_(self)

self.packets = None

self.lastsync = None

def parsesync(self, packets):

packets = packets[0].split("\_")

if (len(packets) == 5) and (packets[0] == SYNC\_SERVER\_PACKET):

self.secret = packets[1]

self.attempts = int(packets[2])

self.userword = packets[3]

self.used\_letters = list(packets[4])

start()

def run(self):

try:

self.sock = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

self.sock.connect((HOST\_PONG, PORT\_PONG))

except:

logger.critical("Main server don't work! %s %s" % (HOST\_PING, PORT\_PING))

sys.exit(0)

while True:

try:

self.sock.send(CONN\_PING + "@")

data = self.sock.recv(128)

except socket.error, detail:

logging.error(detail)

logger.error("Ping server error! %s" % self.packets)

if self.lastsync:

self.parsesync(self.lastsync)

else:

logger.error("Main server not send sync packet! :(")

sleep(5)

sys.exit(0)

break

if not data: logger.error("Ping server error! %s" % self.sock.fileno())

else:

try:

self.packets = data.strip()

self.packets = self.packets.split("$")

for pack in self.packets:

answers = pack.strip()

answers = pack.split("\_")

for item in answers:

if len(item) > 0:

if item[0] == "#":

code = item[:4]

if code == CONN\_PONG:

logger.info("Ping server success! %s" % self.sock.fileno())

self.sock.send(CONN\_PING + "$")

elif code == SYNC\_SERVER\_PACKET:

print "SYNC: " + str(answers)

self.lastsync = self.packets

self.sock.send(SYNC\_SERVER\_PACKET\_APPLY + "$")

else:

logger.error("Ping server error! CODE: '%s'" % code)

self.parsesync(self.lastsync)

except socket.error:

logger.error("Ping server error! %s" % self.packets)

self.parsesync(self.lastsync)

**class Ponger(Thread): #«синхронизуемый», альтерантивный сервер**

def \_\_init\_\_(self):

Thread.\_\_init\_\_(self)

def run(self):

try:

pong = socket.socket(AF\_INET, SOCK\_STREAM)

pong.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

pong.bind((HOST\_PONG, PORT\_PONG))

logger.debug("Ponger server binded")

pong.listen(5)

logger.debug("Ponger server listen on port " + str(PORT\_PONG))

sleep(15)

except socket.error, detail:

logger.error(detail)

pingsock, addr = pong.accept()

logger.info("Ponger connected %s" % str(addr))

try:

while True:

data = pingsock.recv(32)

ping = data.strip()

ping = ping.split("$")

if not data:

logger.error("Pong server error! %s" % pingsock.fileno())

sleep(5)

pingsock, addr = pong.accept()

pingsock.send(SYNC\_SERVER\_PACKET + "\_%s\_%s\_%s\_%s$" % (gallows.secret, str(gallows.attempts), gallows.newuword, str(gallows.used\_letters)))

else:

for item in ping:

if len(ping) > 0:

if len(item) > 0:

if item[0] == "#":

code = item[:4]

if code == CONN\_PING:

if (hasattr(gallows, 'secret') and not main\_server) or (s.changed and main\_server):

pingsock.send(SYNC\_SERVER\_PACKET + "\_%s\_%s\_%s\_%s$" % (gallows.secret, str(gallows.attempts), gallows.newuword, str(gallows.used\_letters)))

s.changed = False

logger.info("Send: " + SYNC\_SERVER\_PACKET + "\_%s\_%s\_%s\_%s$" % (gallows.secret, str(gallows.attempts), gallows.newuword, str(gallows.used\_letters)))

sleep(2)

else:

pingsock.send(CONN\_PONG + "$")

sleep(2)

break

elif code == SYNC\_SERVER\_PACKET\_APPLY:

logger.info("SYNC OK! %s" % pingsock.fileno())

elif item[0] != "#":

break

else:

logger.info("Pong server unknown answer code! CODE: %s" % code)

except socket.error, detail:

logger.error("Pong server error! %s" % pingsock.fileno())