98 TCP网络聊天室

98.1 支持多客户机的TCP服务器

98.1.1 完善功能

C:\Users\Minwei\Projects\Qt\TcpServer\tcpserverwindow.h：

```
#ifndef TCPSERVERWINDOW_H
#define TCPSERVERWINDOW_H

#include <QMainWindow>
#include <QLabel>
#include <QTcpServer>
#include <QTcpSocket>
#include <QVector>

QT_BEGIN_NAMESPACE
namespace Ui { class TcpServerWindow; }
QT_END_NAMESPACE

class TcpServerWindow : public QMainWindow
{
    Q_OBJECT

public:
    TcpServerWindow(QWidget *parent = nullptr);
    ~TcpServerWindow();

private slots:
    void on_m_actListen_triggered();
    void on_m_actClose_triggered();
    void on_m_actClear_triggered();

    void on_m_btnSend_clicked();

    void on_m_server_newConnection();

    void on_m_socket_readyRead();
    void on_m_socket_disconnected();

    void on_m_socket_stateChanged(
        QAbstractSocket::SocketState socketState);
    void on_m_socket_error(
        QAbstractSocket::SocketError socketError);

private:
    void broadcast(QByteArray const& data);

    Ui::TcpServerWindow *ui;
    QLabel* m_labListenState;
    QLabel* m_labSocketState;
    QLabel* m_labSocketError;
    QTcpServer* m_server;
```

```
    QVector<QTcpSocket*> m_sockets;
};

#endif // TCPSERVERWINDOW_H
```

C:\Users\Minwei\Projects\Qt\TcpServer\tcpserverwindow.cpp：

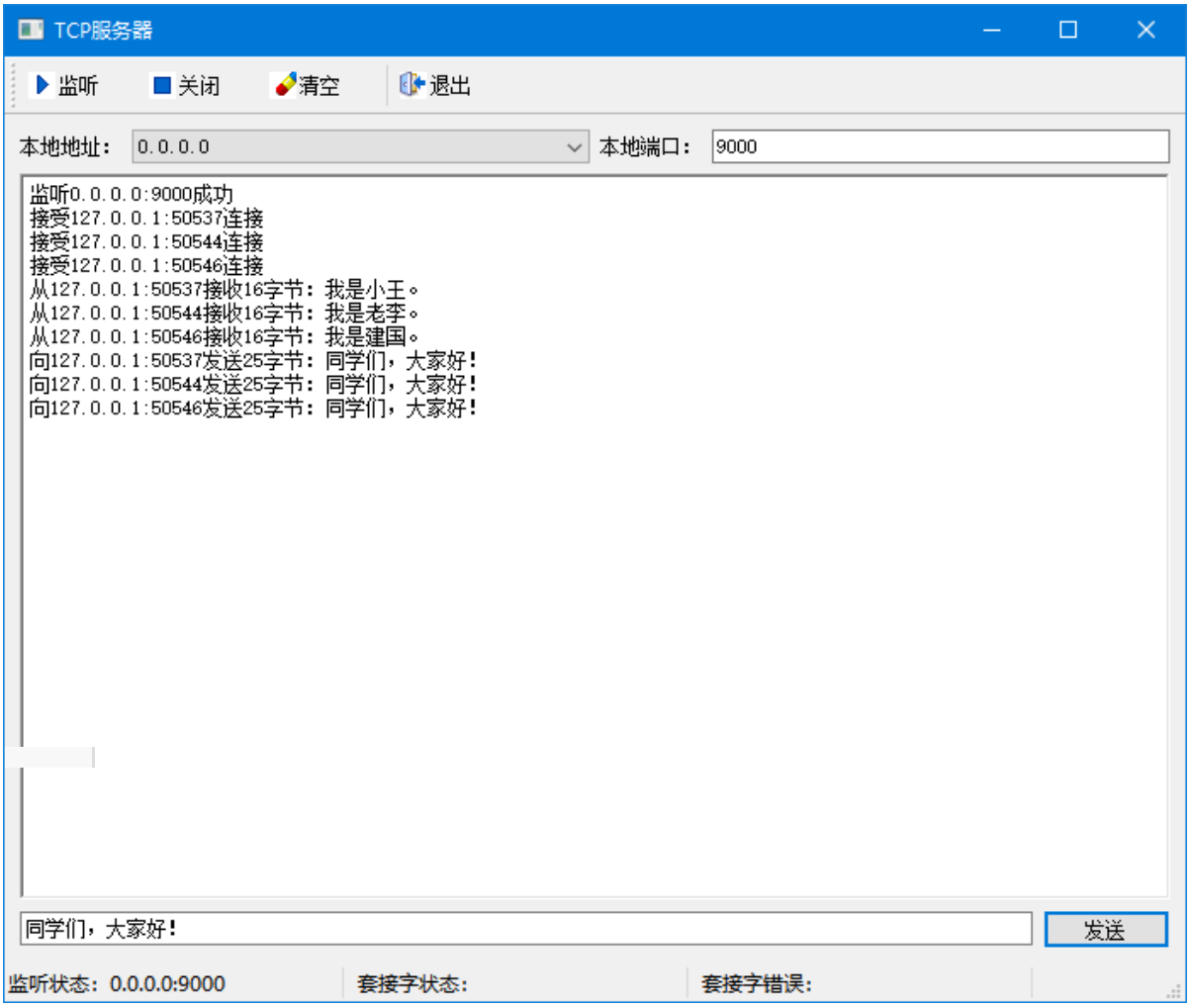
```
#include <QHostInfo>
#include <QMetaEnum>

#include "tcpserverwindow.h"
#include "ui_tcpserverwindow.h"

TcpServerWindow::TcpServerWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::TcpServerWindow)
    , m_labListenState(new QLabel("监听状态：关闭"))
    , m_labSocketState(new QLabel(" 套接字状态："))
    , m_labSocketError(new QLabel(" 套接字错误："))
    , m_server(new QTcpServer(this))
{
    ui->setupUi(this);

    for (QHostAddress address : QHostInfo::fromName(
        QHostInfo::localHostName()).addresses())
        if(address.protocol() == QAbstractSocket::IPv4Protocol)
            ui->m_comboLocalAddr->addItem(address.toString());

    m_labListenState->setMinimumWidth(196);
    ui->m_statusBar->addWidget(m_labListenState);
    m_labSocketState->setMinimumWidth(197);
    ui->m_statusBar->addWidget(m_labSocketState);
    m_labSocketError->setMinimumWidth(197);
    ui->m_statusBar->addWidget(m_labSocketError);

    connect(m_server, SIGNAL(newConnection()),
        this, SLOT(on_m_server_newConnection()));
}

TcpServerWindow::~TcpServerWindow()
{
    delete ui;
}

void TcpServerWindow::on_m_actListen_triggered()
{
    QHostAddress address(ui->m_comboLocalAddr->currentText());
    quint16 port = ui->m_editLocalPort->text().toUShort();

    if (m_server->listen(address, port))
    {
        ui->m_editOutput->appendPlainText(QString("监听%1:%2成功").
            arg(address.toString()).arg(port));
        m_labListenState->setText(QString("监听状态：%1:%2").
            arg(address.toString()).arg(port));
```

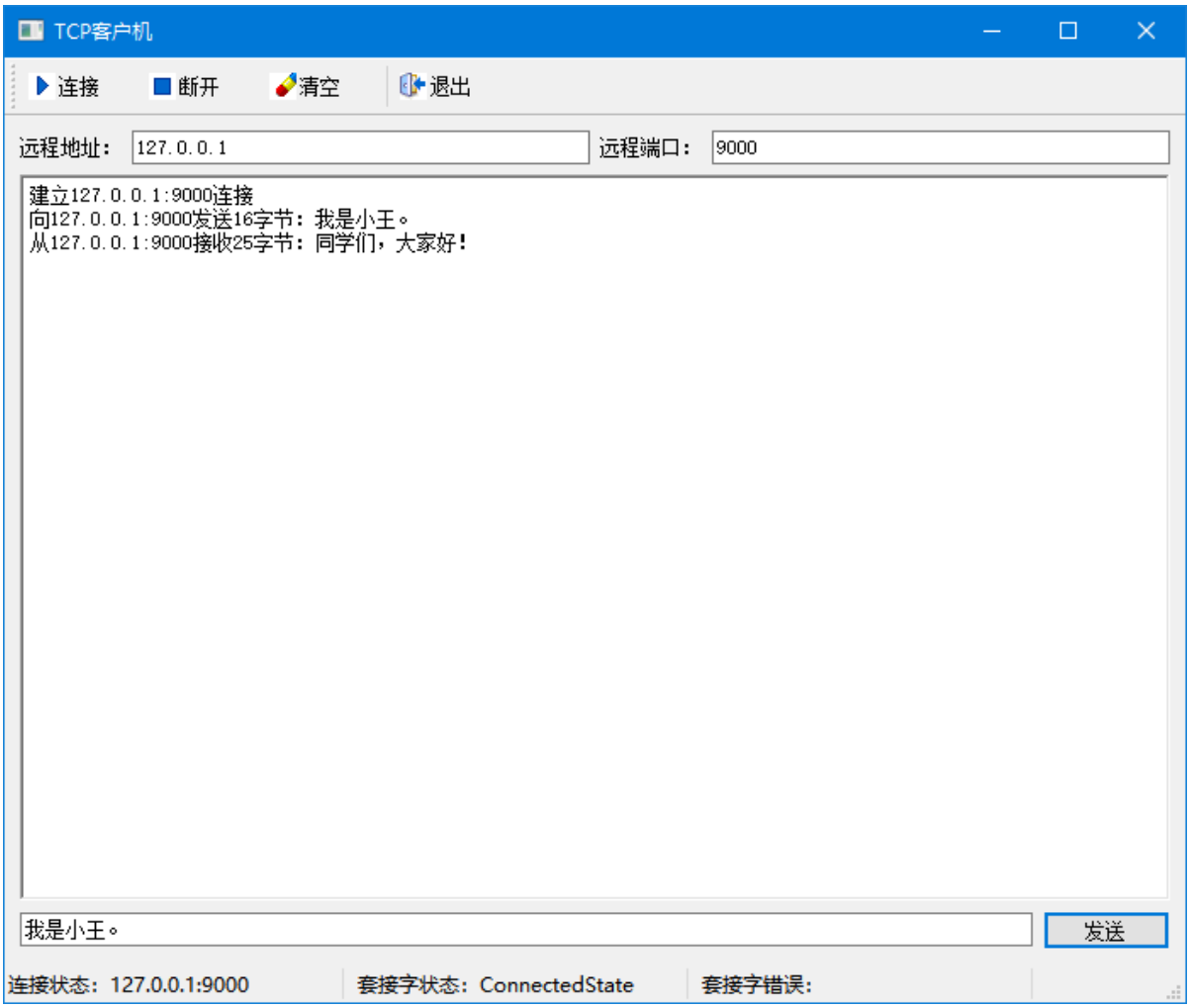
```
    }
    else
        ui->m_editOutput->appendPlainText(QString("监听%1:%2失败").
            arg(address.toString()).arg(port));
}

void TcpServerWindow::on_m_actClose_triggered()
{
    m_server->close();

    ui->m_editOutput->appendPlainText("关闭监听");
    m_labListenState->setText("监听状态：关闭");
}

void TcpServerWindow::on_m_actClear_triggered()
{
    ui->m_editOutput->clear();
}

void TcpServerWindow::on_m_btnSend_clicked()
{
    broadcast((ui->m_editSend->text() + "\n").toUtf8());
}

void TcpServerWindow::on_m_server_newConnection()
{
    QTcpSocket* socket = m_server->nextPendingConnection();

    connect(socket, SIGNAL(readyRead()),
        this, SLOT(on_m_socket_readyRead()));
    connect(socket, SIGNAL(disconnected()),
        this, SLOT(on_m_socket_disconnected()));

    connect(socket, SIGNAL(stateChanged(QAbstractSocket::SocketState)),
        this,
SLOT(on_m_socket_stateChanged(QAbstractSocket::SocketState)));
    connect(socket, SIGNAL(error(QAbstractSocket::SocketError)),
        this, SLOT(on_m_socket_error(QAbstractSocket::SocketError)));

    ui->m_editOutput->appendPlainText(QString("接受%1:%2连接").
        arg(socket->peerAddress().toString()).arg(socket->peerPort()));

    m_sockets.push_back(socket);
    ui->m_btnSend->setEnabled(!m_sockets.isEmpty());
}

void TcpServerWindow::on_m_socket_readyRead()
{
    QTcpSocket* socket = (QTcpSocket*)sender();

    while (socket->canReadLine())
    {
        QByteArray data = socket->readLine();

        ui->m_editOutput->appendPlainText(QString("从%1:%2接收%3字节：%4").
            arg(socket->peerAddress().toString()).arg(socket->peerPort()).
```

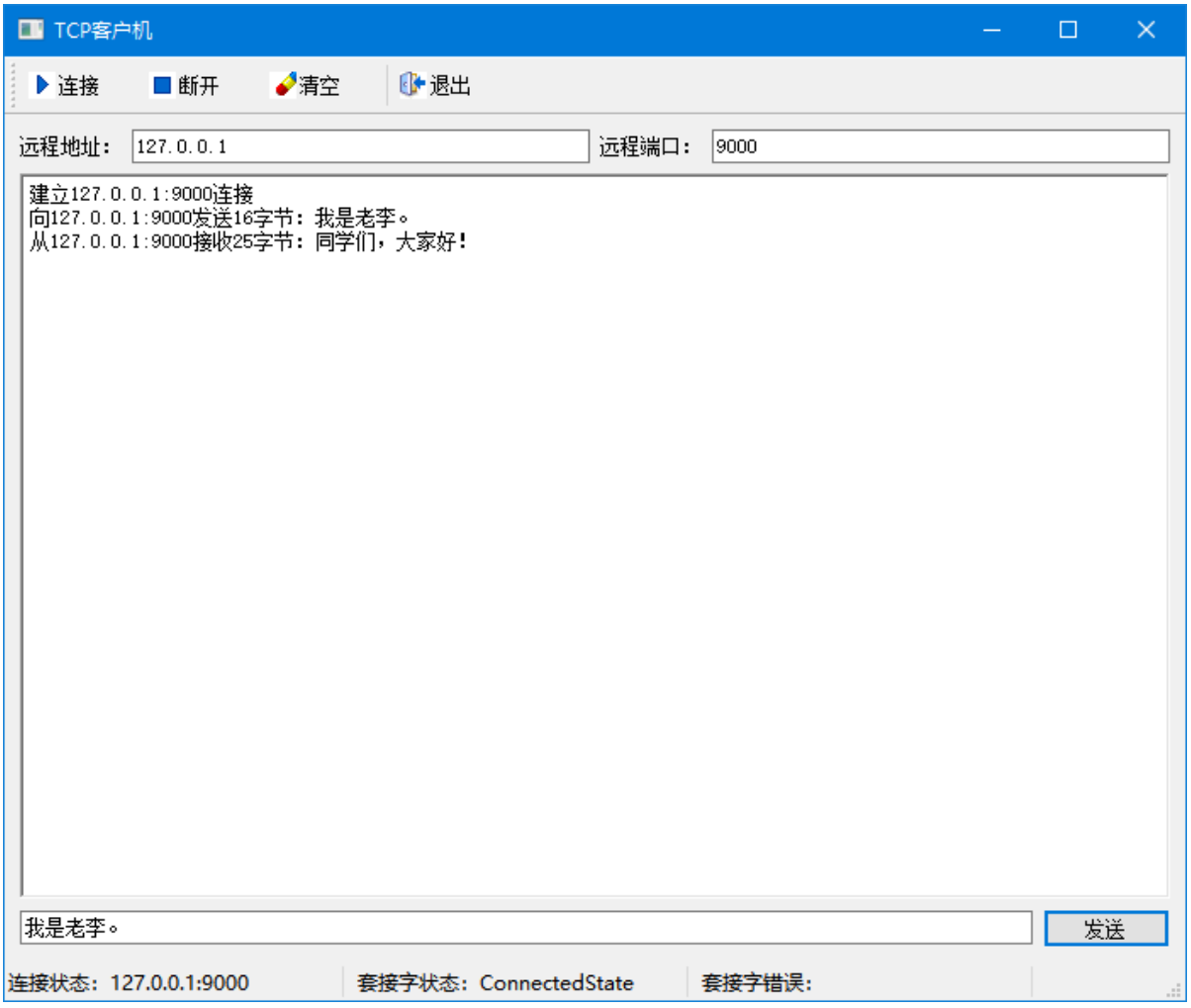
```
            arg(data.size()).arg(QString(data).trimmed()));
    }
}

void TcpServerWindow::on_m_socket_disconnected()
{
    QTcpSocket* socket = (QTcpSocket*)sender();

    m_sockets.removeOne(socket);
    ui->m_btnSend->setEnabled(!m_sockets.isEmpty());

    ui->m_editOutput->appendPlainText(QString("断开%1:%2连接").
        arg(socket->peerAddress().toString()).arg(socket->peerPort()));

    socket->deleteLater();
}

void TcpServerWindow::on_m_socket_stateChanged(
    QAbstractSocket::SocketState socketState)
{
    m_labSocketState->setText(QString(" 套接字状态：%1").arg(
        QMetaEnum::fromType<QAbstractSocket::SocketState>().
        valueToKey(socketState)));
}

void TcpServerWindow::on_m_socket_error(
    QAbstractSocket::SocketError socketError)
{
    m_labSocketError->setText(QString(" 套接字错误：%1").arg(
        QMetaEnum::fromType<QAbstractSocket::SocketError>().
        valueToKey(socketError)));
}

void TcpServerWindow::broadcast(QByteArray const& data)
{
    for (QTcpSocket* socket : m_sockets)
    {
        qint64 nbytes = socket->write(data);

        if (nbytes == -1)
            ui->m_editOutput->appendPlainText("发送失败");
        else
            ui->m_editOutput->appendPlainText(QString("向%1:%2发送%3字节：
%4").
                arg(socket->peerAddress().toString()).arg(socket-
>peerPort()).
                arg(nbytes).arg(QString(data).trimmed()));
    }
}
```

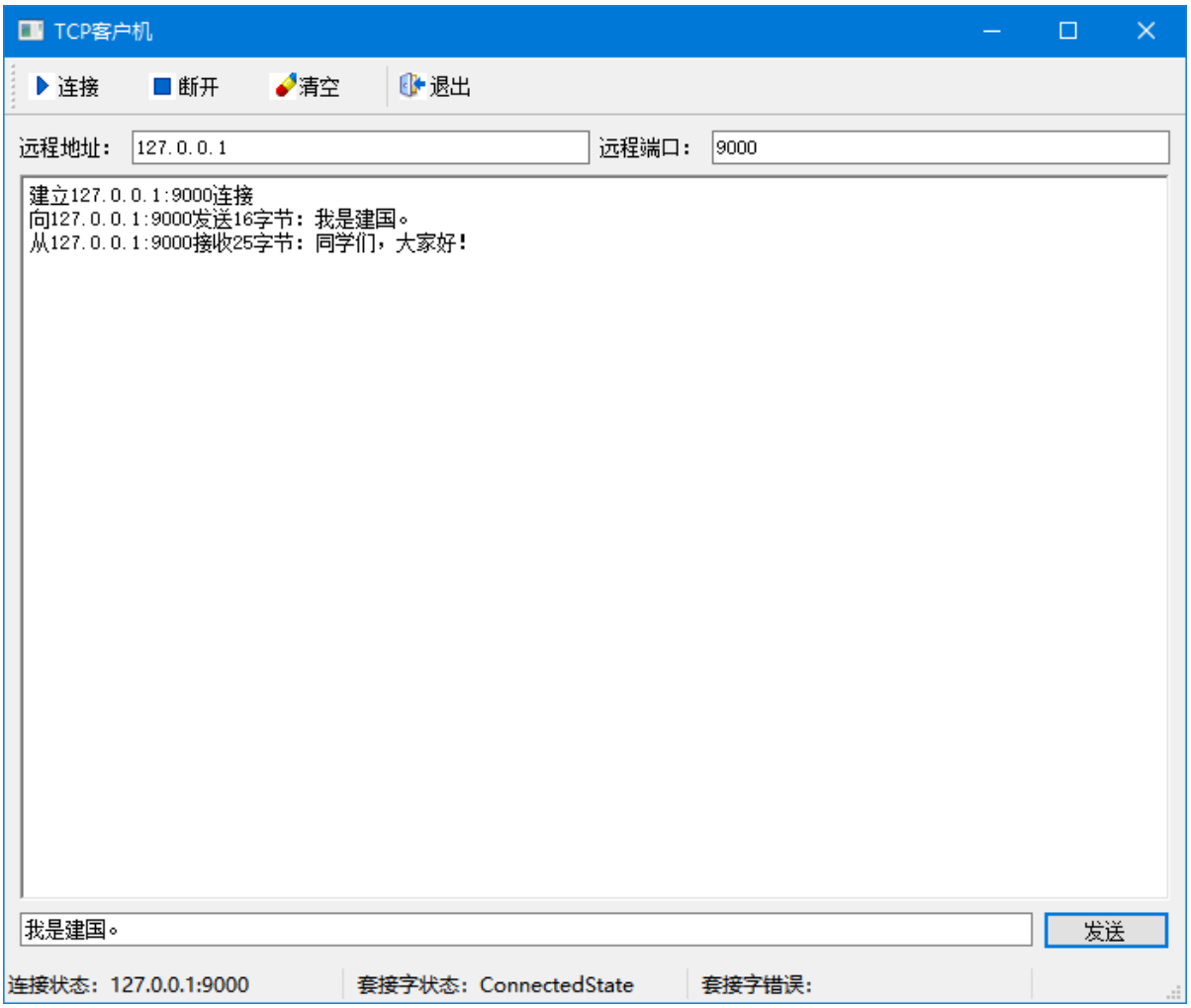
98.1.2 测试验证

运行效果如图所示：




TCP客户机
连接
断开
清空
退出
远程地址：
127.0.0.1
远程端口：
9000
建立127.0.0.1:9000连接
向127.0.0.1:9000发送16字节：我是小王。
从127.0.0.1:9000接收25字节：同学们，大家好！
我是小王。
发送
连接状态：127.0.0.1:9000
套接字状态：ConnectedState
套接字错误：


TCP客户机
连接
断开
清空
退出
远程地址：
127.0.0.1
远程端口：
9000
建立127.0.0.1:9000连接
向127.0.0.1:9000发送16字节：我是老李。
从127.0.0.1:9000接收25字节：同学们，大家好！
我是老李。
发送
连接状态：127.0.0.1:9000
套接字状态：ConnectedState
套接字错误：



98.2 升级为聊天室服务器

98.2.1 升级功能

C:\Users\Minwei\Projects\Qt\TcpServer\tcpserverwindow.cpp：

```
#include <QHostInfo>
#include <QMetaEnum>

#include "tcpserverwindow.h"
#include "ui_tcpserverwindow.h"

TcpServerWindow::TcpServerWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::TcpServerWindow)
    , m_labListenState(new QLabel("监听状态：关闭"))
    , m_labSocketState(new QLabel(" 套接字状态："))
    , m_labSocketError(new QLabel(" 套接字错误："))
    , m_server(new QTcpServer(this))
{
    ui->setupUi(this);

    for (QHostAddress address : QHostInfo::fromName(
        QHostInfo::localHostName()).addresses())
        if(address.protocol() == QAbstractSocket::IPv4Protocol)
            ui->m_comboLocalAddr->addItem(address.toString());

    m_labListenState->setMinimumWidth(196);
```

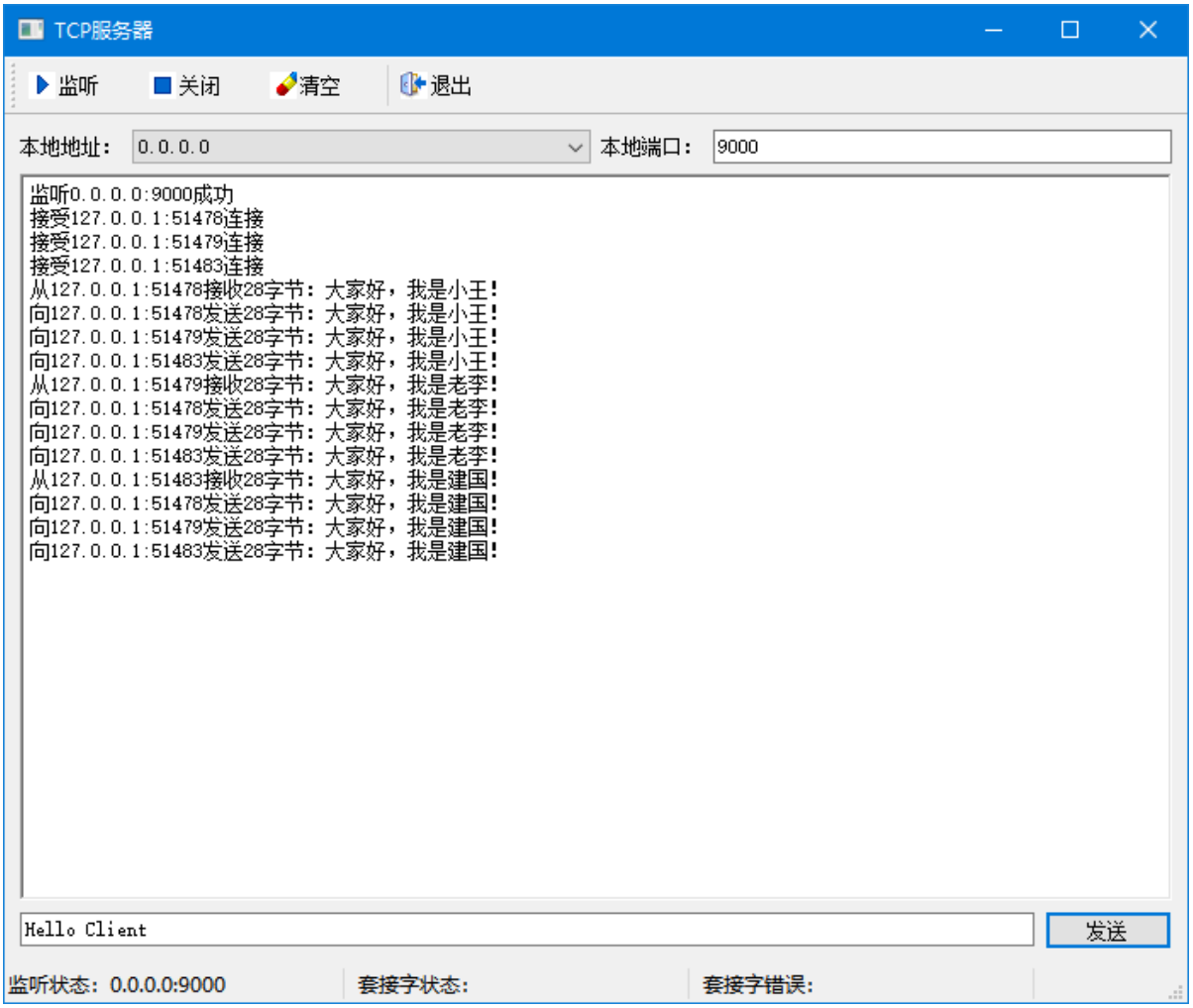
```
    ui->m_statusBar->addWidget(m_labListenState);
    m_labSocketState->setMinimumWidth(197);
    ui->m_statusBar->addWidget(m_labSocketState);
    m_labSocketError->setMinimumWidth(197);
    ui->m_statusBar->addWidget(m_labSocketError);

    connect(m_server, SIGNAL(newConnection()),
        this, SLOT(on_m_server_newConnection()));
}

TcpServerWindow::~TcpServerWindow()
{
    delete ui;
}

void TcpServerWindow::on_m_actListen_triggered()
{
    QHostAddress address(ui->m_comboLocalAddr->currentText());
    quint16 port = ui->m_editLocalPort->text().toUShort();

    if (m_server->listen(address, port))
    {
        ui->m_editOutput->appendPlainText(QString("监听%1:%2成功").
            arg(address.toString()).arg(port));
        m_labListenState->setText(QString("监听状态：%1:%2").
            arg(address.toString()).arg(port));
    }
    else
        ui->m_editOutput->appendPlainText(QString("监听%1:%2失败").
            arg(address.toString()).arg(port));
}

void TcpServerWindow::on_m_actClose_triggered()
{
    m_server->close();

    ui->m_editOutput->appendPlainText("关闭监听");
    m_labListenState->setText("监听状态：关闭");
}

void TcpServerWindow::on_m_actClear_triggered()
{
    ui->m_editOutput->clear();
}

void TcpServerWindow::on_m_btnSend_clicked()
{
    broadcast((ui->m_editSend->text() + "\n").toUtf8());
}

void TcpServerWindow::on_m_server_newConnection()
{
    QTcpSocket* socket = m_server->nextPendingConnection();

    connect(socket, SIGNAL(readyRead()),
        this, SLOT(on_m_socket_readyRead()));
```

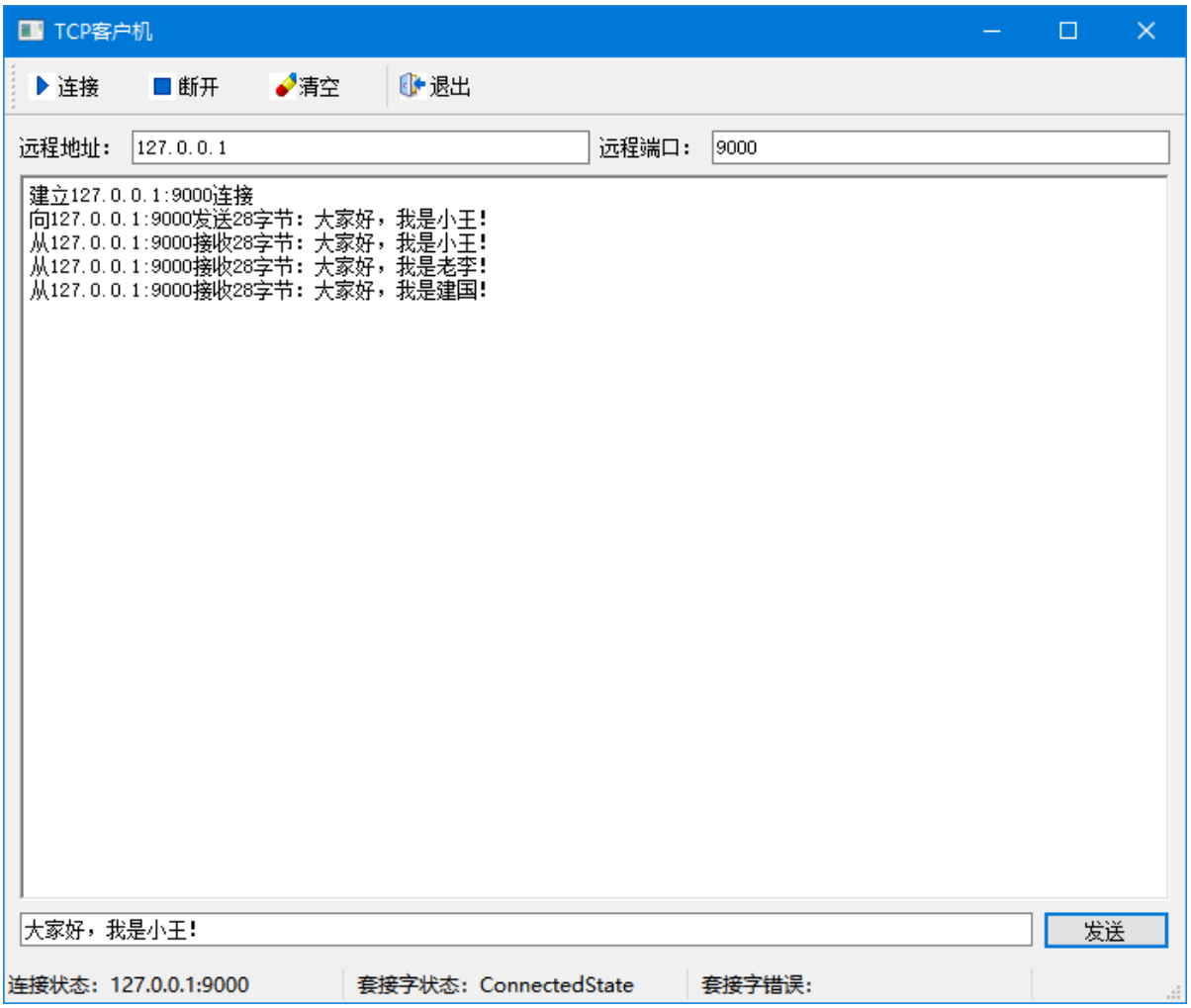
```
    connect(socket, SIGNAL(disconnected()),
        this, SLOT(on_m_socket_disconnected()));

    connect(socket, SIGNAL(stateChanged(QAbstractSocket::SocketState)),
        this,
SLOT(on_m_socket_stateChanged(QAbstractSocket::SocketState)));
    connect(socket, SIGNAL(error(QAbstractSocket::SocketError)),
        this, SLOT(on_m_socket_error(QAbstractSocket::SocketError)));

    ui->m_editOutput->appendPlainText(QString("接受%1:%2连接").
        arg(socket->peerAddress().toString()).arg(socket->peerPort()));

    m_sockets.push_back(socket);
    ui->m_btnSend->setEnabled(!m_sockets.isEmpty());
}

void TcpServerWindow::on_m_socket_readyRead()
{
    QTcpSocket* socket = (QTcpSocket*)sender();

    while (socket->canReadLine())
    {
        QByteArray data = socket->readLine();

        ui->m_editOutput->appendPlainText(QString("从%1:%2接收%3字节：%4").
            arg(socket->peerAddress().toString()).arg(socket->peerPort()).
            arg(data.size()).arg(QString(data).trimmed()));

        broadcast(data);
    }
}

void TcpServerWindow::on_m_socket_disconnected()
{
    QTcpSocket* socket = (QTcpSocket*)sender();

    m_sockets.removeOne(socket);
    ui->m_btnSend->setEnabled(!m_sockets.isEmpty());

    ui->m_editOutput->appendPlainText(QString("断开%1:%2连接").
        arg(socket->peerAddress().toString()).arg(socket->peerPort()));

    socket->deleteLater();
}

void TcpServerWindow::on_m_socket_stateChanged(
    QAbstractSocket::SocketState socketState)
{
    m_labSocketState->setText(QString(" 套接字状态：%1").arg(
        QMetaEnum::fromType<QAbstractSocket::SocketState>().
        valueToKey(socketState)));
}

void TcpServerWindow::on_m_socket_error(
    QAbstractSocket::SocketError socketError)
{
```

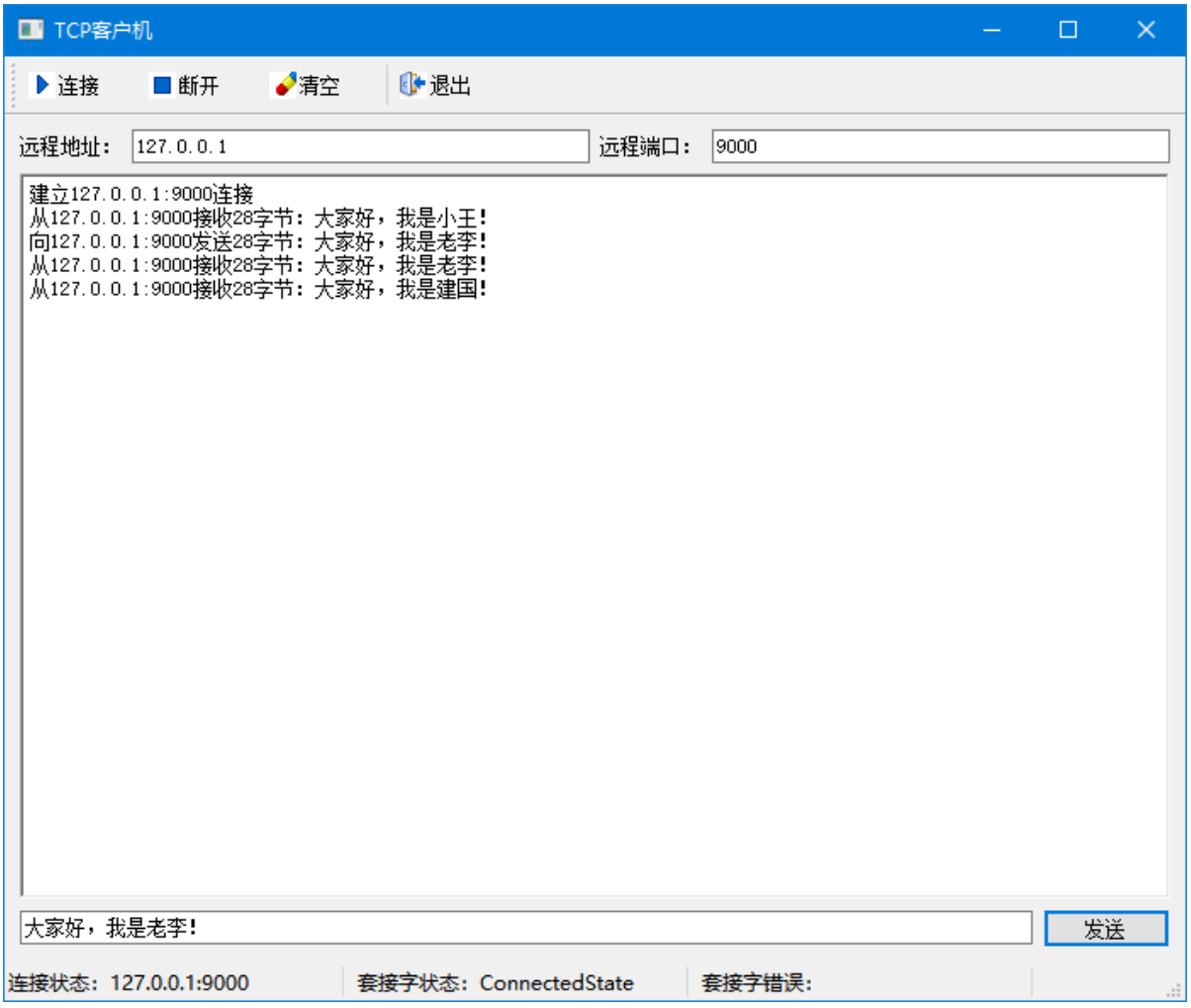
```
    m_labSocketError->setText(QString(" 套接字错误：%1").arg(
        QMetaEnum::fromType<QAbstractSocket::SocketError>().
        valueToKey(socketError)));
}

void TcpServerWindow::broadcast(QByteArray const& data)
{
    for (QTcpSocket* socket : m_sockets)
    {
        qint64 nbytes = socket->write(data);

        if (nbytes == -1)
            ui->m_editOutput->appendPlainText("发送失败");
        else
            ui->m_editOutput->appendPlainText(QString("向%1:%2发送%3字节：
%4").
                arg(socket->peerAddress().toString()).arg(socket-
>peerPort()).
                arg(nbytes).arg(QString(data).trimmed()));
    }
}
```

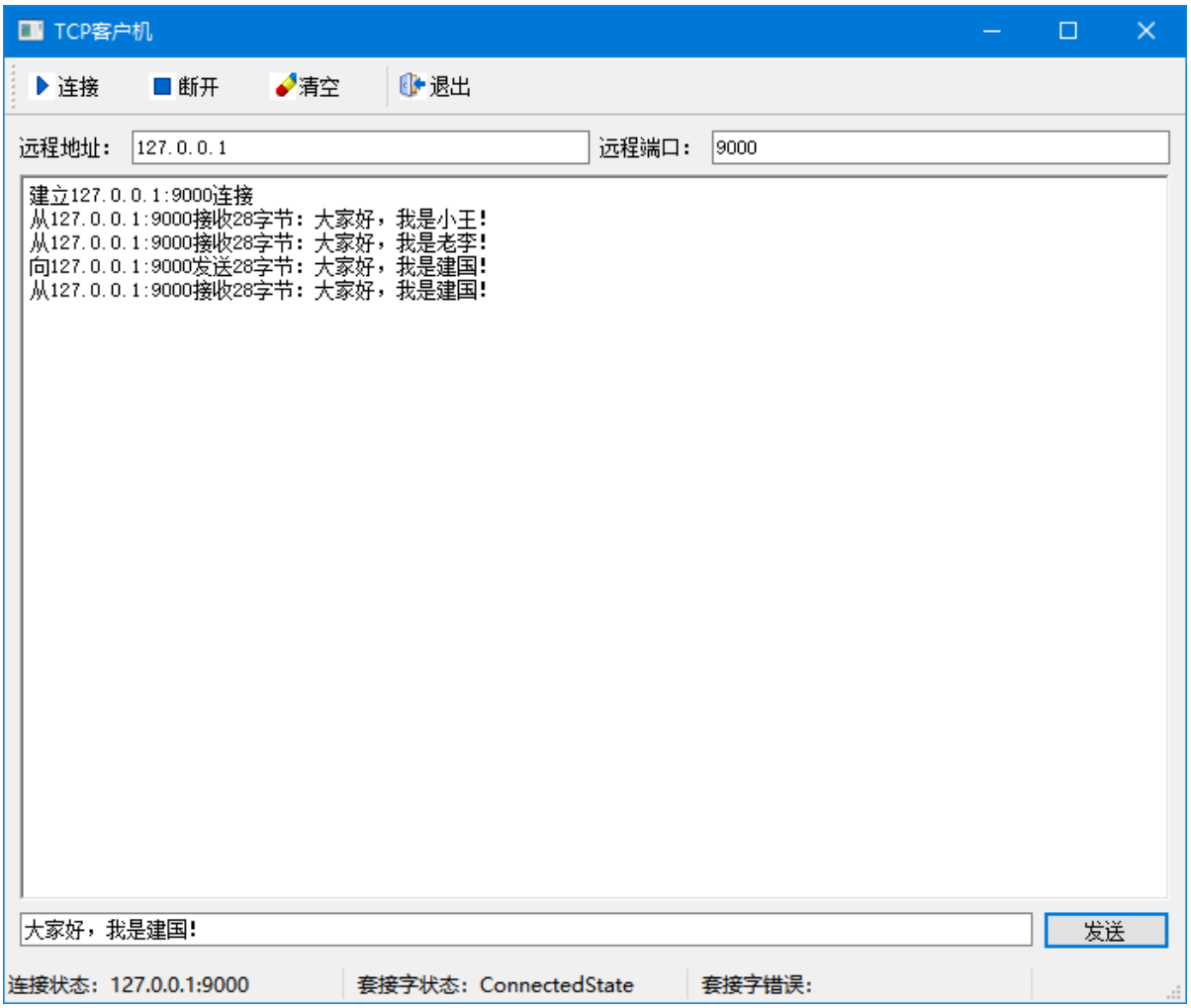
98.2.2 测试验证

运行效果如图所示：




TCP客户机
连接
断开
清空
退出
远程地址：
127.0.0.1
远程端口：
9000
建立127.0.0.1:9000连接
向127.0.0.1:9000发送28字节：大家好，我是小王!
从127.0.0.1:9000接收28字节：大家好，我是小王!
从127.0.0.1:9000接收28字节：大家好，我是老李!
从127.0.0.1:9000接收28字节：大家好，我是建国!
大家好，我是小王!
发送
连接状态：127.0.0.1:9000
套接字状态：ConnectedState
套接字错误：


TCP客户机
连接
断开
清空
退出
远程地址：
127.0.0.1
远程端口：
9000
建立127.0.0.1:9000连接
从127.0.0.1:9000接收28字节：大家好，我是小王!
向127.0.0.1:9000发送28字节：大家好，我是老李!
从127.0.0.1:9000接收28字节：大家好，我是老李!
从127.0.0.1:9000接收28字节：大家好，我是建国!
大家好，我是老李!
发送
连接状态：127.0.0.1:9000
套接字状态：ConnectedState
套接字错误：


TCP客户机
连接
断开
清空
退出
远程地址：
127.0.0.1
远程端口：
9000
建立127.0.0.1:9000连接
从127.0.0.1:9000接收28字节：大家好，我是小王!
从127.0.0.1:9000接收28字节：大家好，我是老李!
向127.0.0.1:9000发送28字节：大家好，我是建国!
从127.0.0.1:9000接收28字节：大家好，我是建国!
大家好，我是建国!
发送
连接状态：127.0.0.1:9000
套接字状态：ConnectedState
套接字错误：