

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:

```
1  #ifndef TCPSEVERWINDOW_H
2  #define TCPSEVERWINDOW_H
3
4  #include <QMainWindow>
5  #include <QLabel>
6  #include <QTcpServer>
7  #include <QTcpSocket>
8  #include <QVector>
9
10 QT_BEGIN_NAMESPACE
11 namespace Ui { class TcpServerWindow; }
12 QT_END_NAMESPACE
13
14 class TcpServerWindow : public QMainWindow
15 {
16     Q_OBJECT
17
18 public:
19     TcpServerWindow(QWidget *parent = nullptr);
20     ~TcpServerWindow();
21
22 private slots:
23     void on_m_actListen_triggered();
24     void on_m_actClose_triggered();
25     void on_m_actClear_triggered();
26
27     void on_m_btnSend_clicked();
28
29     void on_m_server_newConnection();
30
31     void on_m_socket_readyRead();
32     void on_m_socket_disconnected();
33
34     void on_m_socket_stateChanged(
35         QAbstractSocket::SocketState socketState);
36     void on_m_socket_error(
37         QAbstractSocket::SocketError socketError);
38
39 private:
40     void broadcast(QByteArray const& data);
41
42     Ui::TcpServerWindow *ui;
43     QLabel* m_labListenState;
44     QLabel* m_labSocketState;
45     QLabel* m_labSocketError;
46     QTcpServer* m_server;
```

```

47     QVector<QTcpSocket*> m_sockets;
48 };
49
50 #endif // TCPSERVERWINDOW_H

```

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

```

1  #include <QHostInfo>
2  #include <QMetaEnum>
3
4  #include "tcpserverwindow.h"
5  #include "ui_tcpserverwindow.h"
6
7  TcpServerWindow::TcpServerWindow(QWidget *parent)
8      : QMainWindow(parent)
9      , ui(new Ui::TcpServerWindow)
10     , m_labListenState(new QLabel("监听状态: 关闭"))
11     , m_labSocketState(new QLabel("套接字状态: "))
12     , m_labSocketError(new QLabel("套接字错误: "))
13     , m_server(new QTcpServer(this))
14 {
15     ui->setupUi(this);
16
17     for (QHostAddress address : QHostInfo::fromName(
18         QHostInfo::localHostName()).addresses())
19         if(address.protocol() == QAbstractSocket::IPv4Protocol)
20             ui->m_comboLocalAddr->addItem(address.toString());
21
22     m_labListenState->setMinimumWidth(196);
23     ui->m_statusBar->addWidget(m_labListenState);
24     m_labSocketState->setMinimumWidth(197);
25     ui->m_statusBar->addWidget(m_labSocketState);
26     m_labSocketError->setMinimumWidth(197);
27     ui->m_statusBar->addWidget(m_labSocketError);
28
29     connect(m_server, SIGNAL(newConnection()),
30         this, SLOT(on_m_server_newConnection()));
31 }
32
33 TcpServerWindow::~TcpServerWindow()
34 {
35     delete ui;
36 }
37
38 void TcpServerWindow::on_m_actListen_triggered()
39 {
40     QHostAddress address(ui->m_comboLocalAddr->currentText());
41     quint16 port = ui->m_editLocalPort->text().toUShort();
42
43     if (m_server->listen(address, port))
44     {
45         ui->m_editOutput->appendPlainText(QString("监听%1:%2成功").
46             arg(address.toString()).arg(port));
47         m_labListenState->setText(QString("监听状态: %1:%2").
48             arg(address.toString()).arg(port));

```

```

49     }
50     else
51         ui->m_editOutput->appendPlainText(QString("监听%1:%2失败").
52             arg(address.toString()).arg(port));
53     }
54
55 void TcpServerWindow::on_m_actClose_triggered()
56 {
57     m_server->close();
58
59     ui->m_editOutput->appendPlainText("关闭监听");
60     m_labListenState->setText("监听状态: 关闭");
61 }
62
63 void TcpServerWindow::on_m_actClear_triggered()
64 {
65     ui->m_editOutput->clear();
66 }
67
68 void TcpServerWindow::on_m_btnSend_clicked()
69 {
70     broadcast((ui->m_editSend->text() + "\n").toUtf8());
71 }
72
73 void TcpServerWindow::on_m_server_newConnection()
74 {
75     QTcpSocket* socket = m_server->nextPendingConnection();
76
77     connect(socket, SIGNAL(readyRead()),
78         this, SLOT(on_m_socket_readyRead()));
79     connect(socket, SIGNAL(disconnected()),
80         this, SLOT(on_m_socket_disconnected()));
81
82     connect(socket, SIGNAL(stateChanged(QAbstractSocket::SocketState)),
83         this,
84         SLOT(on_m_socket_stateChanged(QAbstractSocket::SocketState)));
85     connect(socket, SIGNAL(error(QAbstractSocket::SocketError)),
86         this, SLOT(on_m_socket_error(QAbstractSocket::SocketError)));
87
88     ui->m_editOutput->appendPlainText(QString("接受%1:%2连接").
89         arg(socket->peerAddress().toString()).arg(socket->peerPort()));
90
91     m_sockets.push_back(socket);
92     ui->m_btnSend->setEnabled(!m_sockets.isEmpty());
93 }
94 void TcpServerWindow::on_m_socket_readyRead()
95 {
96     QTcpSocket* socket = (QTcpSocket*)sender();
97
98     while (socket->canReadLine())
99     {
100         QByteArray data = socket->readLine();
101
102         ui->m_editOutput->appendPlainText(QString("从%1:%2接收%3字节: %4").
103             arg(socket->peerAddress().toString()).arg(socket->peerPort()).

```

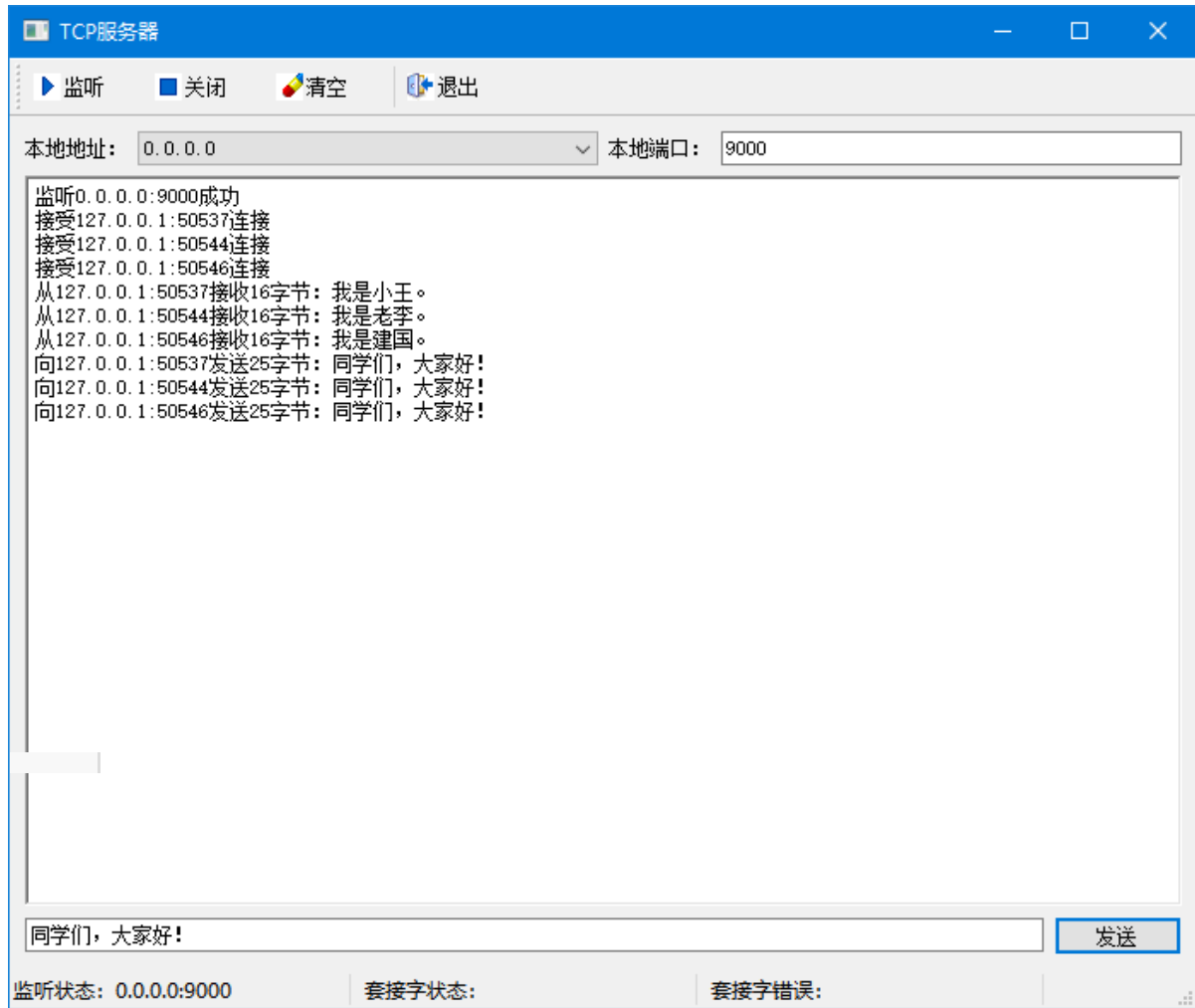
```

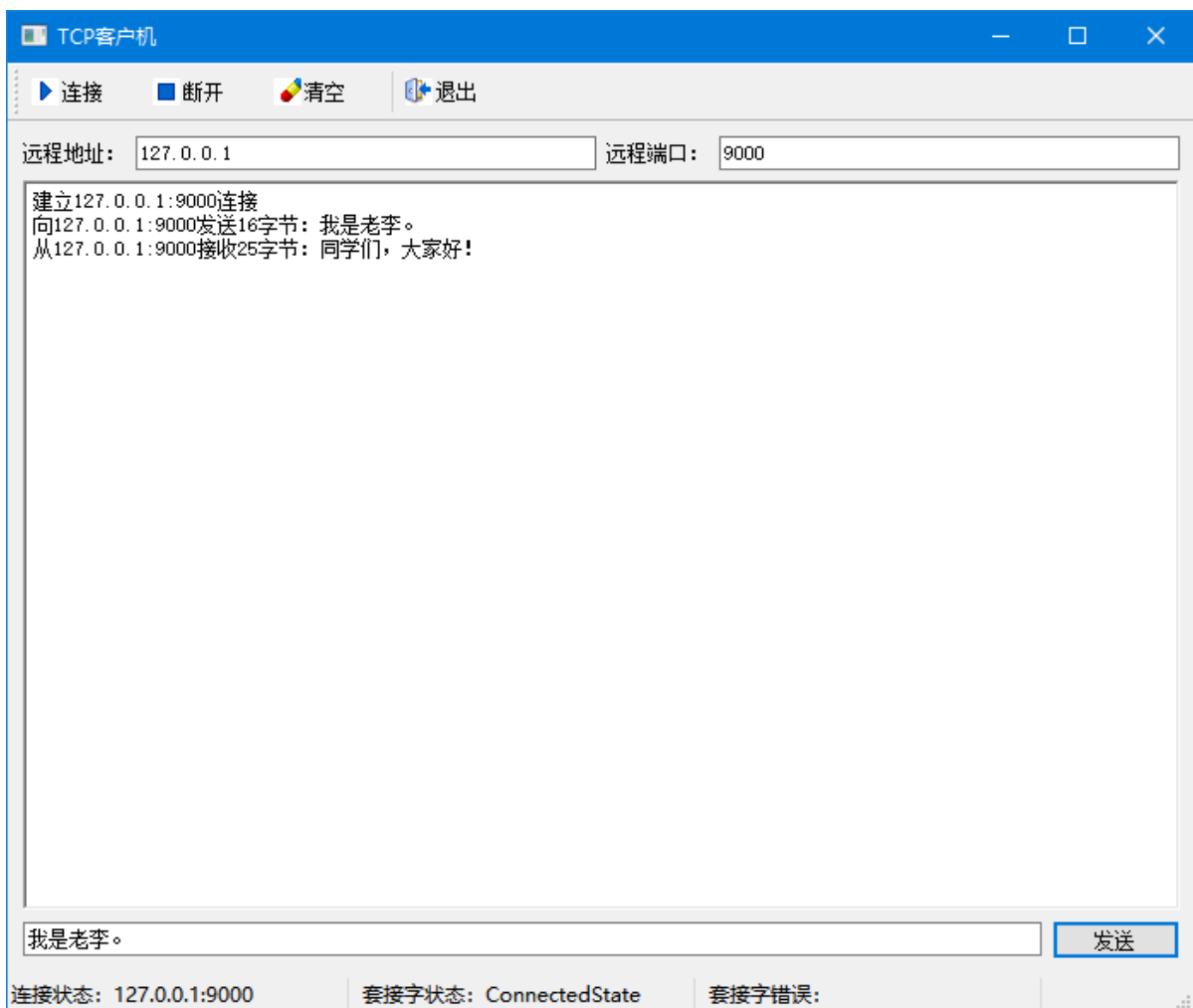
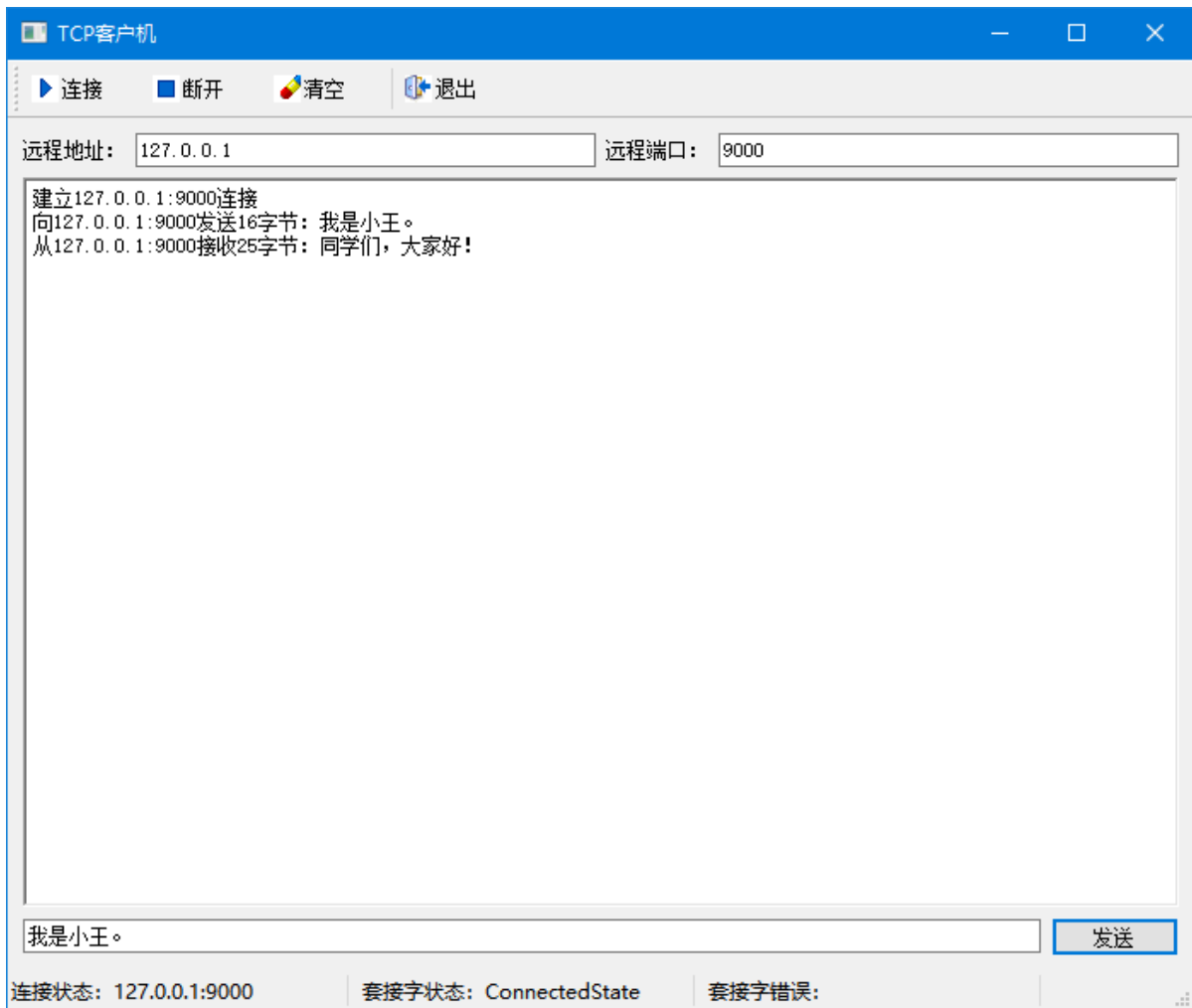
104         arg(data.size()).arg(QString(data).trimmed());
105     }
106 }
107
108 void TcpServerWindow::on_m_socket_disconnected()
109 {
110     QTcpSocket* socket = (QTcpSocket*)sender();
111
112     m_sockets.removeOne(socket);
113     ui->m_btnSend->setEnabled(!m_sockets.isEmpty());
114
115     ui->m_editOutput->appendPlainText(QString("断开%1:%2连接").
116         arg(socket->peerAddress().toString()).arg(socket->peerPort()));
117
118     socket->deleteLater();
119 }
120
121 void TcpServerWindow::on_m_socket_stateChanged(
122     QAbstractSocket::SocketState socketState)
123 {
124     m_labSocketState->setText(QString("套接字状态: %1").arg(
125         QMetaEnum::fromType<QAbstractSocket::SocketState>().
126         valueToKey(socketState)));
127 }
128
129 void TcpServerWindow::on_m_socket_error(
130     QAbstractSocket::SocketError socketError)
131 {
132     m_labSocketError->setText(QString("套接字错误: %1").arg(
133         QMetaEnum::fromType<QAbstractSocket::SocketError>().
134         valueToKey(socketError)));
135 }
136
137 void TcpServerWindow::broadcast(QByteArray const& data)
138 {
139     for (QTcpSocket* socket : m_sockets)
140     {
141         qint64 nbytes = socket->write(data);
142
143         if (nbytes == -1)
144             ui->m_editOutput->appendPlainText("发送失败");
145         else
146             ui->m_editOutput->appendPlainText(QString("向%1:%2发送%3字节:
147 %4").
148                 arg(socket->peerAddress().toString()).arg(socket->peerPort()).
149                 arg(nbytes).arg(QString(data).trimmed()));
150     }
151 }

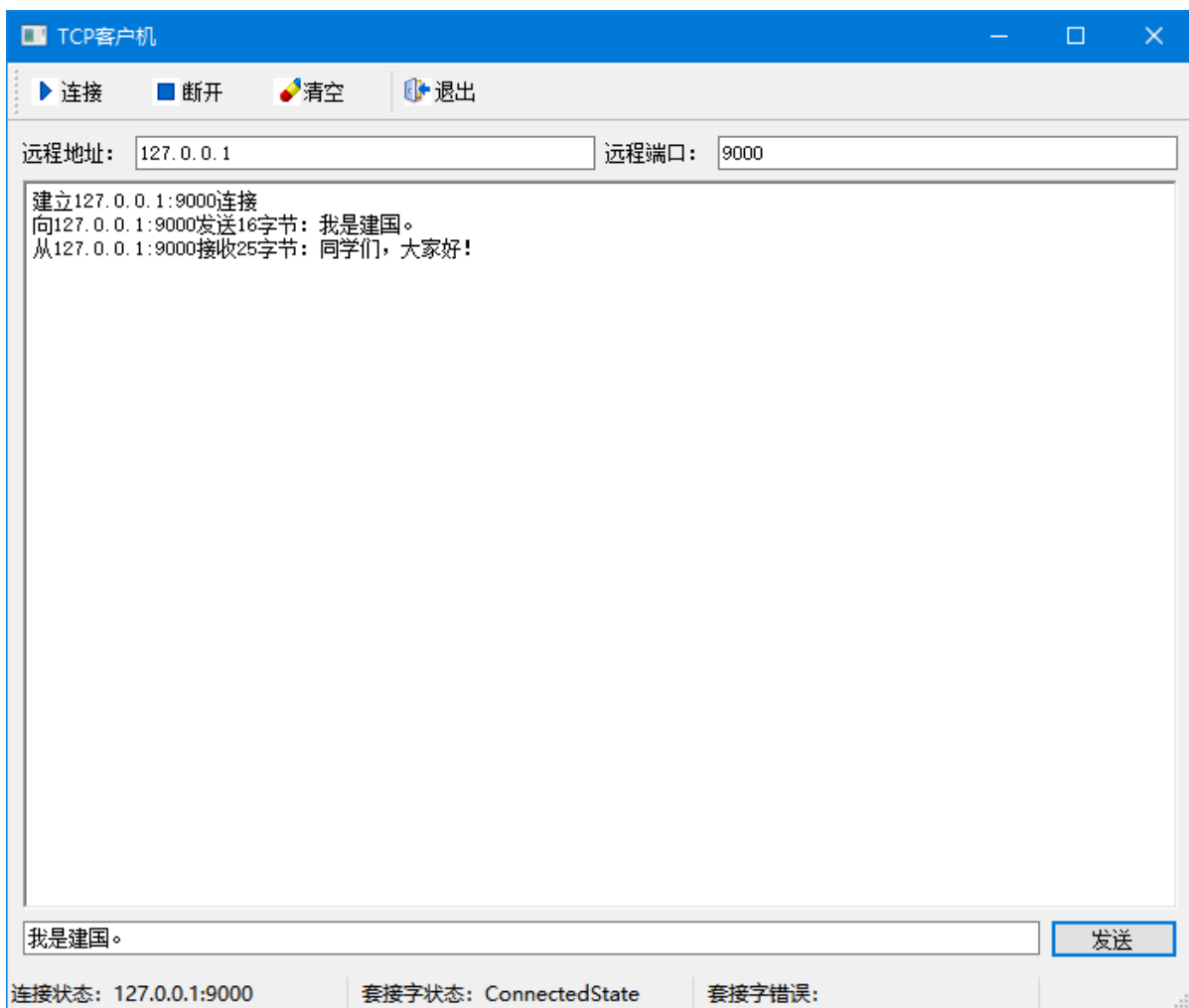
```

98.1.2 测试验证

运行效果如图所示：







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:

```
1  #include <QHostInfo>
2  #include <QMetaEnum>
3
4  #include "tcpserverwindow.h"
5  #include "ui_tcpserverwindow.h"
6
7  TcpServerwindow::TcpServerwindow(QWidget *parent)
8      : QMainWindow(parent)
9      , ui(new Ui::TcpServerwindow)
10     , m_labListenState(new QLabel("监听状态: 关闭"))
11     , m_labSocketState(new QLabel("套接字状态: "))
12     , m_labSocketError(new QLabel("套接字错误: "))
13     , m_server(new QTcpServer(this))
14 {
15     ui->setupUi(this);
16
17     for (QHostAddress address : QHostInfo::fromName(
18         QHostInfo::localHostName()).addresses())
19         if(address.protocol() == QAbstractSocket::IPv4Protocol)
20             ui->m_comboLocalAddr->addItem(address.toString());
21
22     m_labListenState->setMinimumWidth(196);
```

```

23     ui->m_statusBar->addWidget(m_labListenState);
24     m_labSocketState->setMinimumWidth(197);
25     ui->m_statusBar->addWidget(m_labSocketState);
26     m_labSocketError->setMinimumWidth(197);
27     ui->m_statusBar->addWidget(m_labSocketError);
28
29     connect(m_server, SIGNAL(newConnection()),
30            this, SLOT(on_m_server_newConnection()));
31 }
32
33 TcpServerWindow::~TcpServerWindow()
34 {
35     delete ui;
36 }
37
38 void TcpServerWindow::on_m_actListen_triggered()
39 {
40     QHostAddress address(ui->m_comboLocalAddr->currentText());
41     quint16 port = ui->m_editLocalPort->text().toUShort();
42
43     if (m_server->listen(address, port))
44     {
45         ui->m_editOutput->appendPlainText(QString("监听%1:%2成功").
46            arg(address.toString()).arg(port));
47         m_labListenState->setText(QString("监听状态: %1:%2").
48            arg(address.toString()).arg(port));
49     }
50     else
51         ui->m_editOutput->appendPlainText(QString("监听%1:%2失败").
52            arg(address.toString()).arg(port));
53 }
54
55 void TcpServerWindow::on_m_actClose_triggered()
56 {
57     m_server->close();
58
59     ui->m_editOutput->appendPlainText("关闭监听");
60     m_labListenState->setText("监听状态: 关闭");
61 }
62
63 void TcpServerWindow::on_m_actClear_triggered()
64 {
65     ui->m_editOutput->clear();
66 }
67
68 void TcpServerWindow::on_m_btnSend_clicked()
69 {
70     broadcast((ui->m_editSend->text() + "\n").toUtf8());
71 }
72
73 void TcpServerWindow::on_m_server_newConnection()
74 {
75     QTcpSocket* socket = m_server->nextPendingConnection();
76
77     connect(socket, SIGNAL(readyRead()),
78            this, SLOT(on_m_socket_readyRead()));

```



```

79     connect(socket, SIGNAL(disconnected()),
80             this, SLOT(on_m_socket_disconnected()));
81
82     connect(socket, SIGNAL(stateChanged(QAbstractSocket::SocketState)),
83             this,
84             SLOT(on_m_socket_stateChanged(QAbstractSocket::SocketState)));
85     connect(socket, SIGNAL(error(QAbstractSocket::SocketError)),
86             this, SLOT(on_m_socket_error(QAbstractSocket::SocketError)));
87
88     ui->m_editOutput->appendPlainText(QString("接受%1:%2连接").
89                                     arg(socket->peerAddress().toString()).arg(socket->peerPort()));
90
91     m_sockets.push_back(socket);
92     ui->m_btnSend->setEnabled(!m_sockets.isEmpty());
93 }
94
95 void TcpServerWindow::on_m_socket_readyRead()
96 {
97     QTcpSocket* socket = (QTcpSocket*)sender();
98
99     while (socket->canReadLine())
100     {
101         QByteArray data = socket->readLine();
102
103         ui->m_editOutput->appendPlainText(QString("从%1:%2接收%3字节: %4").
104                                         arg(socket->peerAddress().toString()).arg(socket->peerPort()).
105                                         arg(data.size()).arg(QString(data).trimmed()));
106
107         broadcast(data);
108     }
109 }
110
111 void TcpServerWindow::on_m_socket_disconnected()
112 {
113     QTcpSocket* socket = (QTcpSocket*)sender();
114
115     m_sockets.removeOne(socket);
116     ui->m_btnSend->setEnabled(!m_sockets.isEmpty());
117
118     ui->m_editOutput->appendPlainText(QString("断开%1:%2连接").
119                                     arg(socket->peerAddress().toString()).arg(socket->peerPort()));
120
121     socket->deleteLater();
122 }
123
124 void TcpServerWindow::on_m_socket_stateChanged(
125     QAbstractSocket::SocketState socketState)
126 {
127     m_labSocketState->setText(QString("套接字状态: %1").arg(
128         QMetaEnum::fromType<QAbstractSocket::SocketState>().
129         valueToKey(socketState)));
130 }
131
132 void TcpServerWindow::on_m_socket_error(
133     QAbstractSocket::SocketError socketError)
134 {

```

```

134     m_tabSocketError->setText(QString(" 套接字错误: %1").arg(
135         QMetaEnum::fromType<QAbstractSocket::SocketError>().
136         valueToKey(socketError)));
137 }
138
139 void TcpServerWindow::broadcast(QByteArray const& data)
140 {
141     for (QTcpSocket* socket : m_sockets)
142     {
143         qint64 nbytes = socket->write(data);
144
145         if (nbytes == -1)
146             ui->m_editOutput->appendPlainText("发送失败");
147         else
148             ui->m_editOutput->appendPlainText(QString("向%1:%2发送%3字节:
149             %4").
150             arg(socket->peerAddress().toString()).arg(socket->
151             >peerPort()).
152             arg(nbytes).arg(QString(data).trimmed()));
153     }
154 }

```

98.2.2 测试验证

运行效果如图所示:

