

个人资料



长沙三毛

访问：395638次

积分：4846

等级：

BLOG

5

排名：第5520名

原创：80篇

转载：1篇

译文：4篇

评论：482条

文章搜索

文章分类

.NET委托与事件 (7)

.NET实现与应用技术 (25)

ComboBox控件 (3)

DataGridView控件 (11)

DateTime控件 (2)

GZipCompress控件 (1)

html/asp技术 (4)

ProgressBar控件 (2)

Socket组件与框架 (5)

SQL Server技术 (2)

TextBox控件 (3)

WebService技术 (2)

技术人生随笔 (12)

软件运行维护 (4)

软件项目开发 (1)

计算机教学与实践 (4)

ASP.NET技术 (7)

jQuery与Javascript (3)

文章存档

2017年05月 (2)

2017年03月 (1)

2017年01月 (1)

2016年12月 (1)

2016年04月 (1)

展开

阅读排行

可扩展多线程异步Socket

(42976)

可扩展多线程异步Socket服务器框架EMTASS 2.0

标签：服务器 socket 框架 扩展 多线程 数据库

2008-10-27 15:02 42976人阅读

☰ 分类： Socket组件与框架 (4) ▾

版权声明：本文为博主原创文章，未经博主允许不得转载。

目录(?)

[+]

(原创文章，转载请注明来源：<http://blog.csdn.net/hulihui>)

0 前言

>>[前言]、[第1节]、[第2节]、[第3节]、[第4节]、[第5节]、[第6节]

在程序设计与实际应用中，Socket数据包接收服务器够得上一个经典问题了：需要计算机与网络编程知识（主要是Socket），与业务处理逻辑密切（如：包组成规则），同时还要兼顾系统运行的稳定、效率、安全与管理等。具体应用时，在满足业务处理逻辑要求的基础上，存在侧重点：有些需要考虑并发与效率，有些需要强调稳定与可靠等等。虽然.NET 2.0 Framework上的IOCP（I/O完成端口）异步技术可以有效解决并发等问题，但完全的异步模式也缺乏一些控制上的灵活性，例如：Socket暂停操作等。

本文介绍的是一个传统Socket数据包服务器解决方案，该方案改自笔者2005年底的一个交通部省级公路交通流量数据服务器中心（DSC）项目。当时.NET Framework 2.0 与 Visual Studio 2005 发布没多久，笔者接触C#的时间不长。于是Google了国内国外网，希望找点应用C#解决Socket通信问题的思路和代码。最后，找到了两篇帮助最大的文章：一篇是国人2005年3月写的Socket接收器框架——[在C#中使用异步Socket编程实现TCP网络服务的C/S的通讯构架\(一\)](#)（分（一）、（二）两篇），该文应用了客户端Socket会话（Session）概念；另一篇是美国人写的，提出了多线程、分段接收数据包的技术方案，描述了对多线程、异步Socket的许多实现细节，该文坚定了笔者采用多线程和异步方式处理Socket接收器的技术路线。第一个版本EMTASS 1.0（EMTASS，Extensible Multi-Thread Asynchronous Socket Server）于2006年初完成并投入使用。

今年暑假，笔者修改了原Socket接收服务器代码，即EMTASS 1.1。最近，又按框架的可扩展性、可重用性等要求重新构思和设计了EMTASS，即EMTASS 2.0。下面的介绍共分六个部分：

1. 总体思路与架构

2. 关键实现技术

3. 框架使用简介

4. 一般测试结果

5. 总结与展望

6. 版本与源码

2 总体思路与架构

>>[前言]、[第1节]、[第2节]、[第3节]、[第4节]、[第5节]、[第6节]

2.1 总体思路

总体构思上，主要考虑多线程、异步Socket和可扩展性三个方面。

1) 三个核心线程

在Internet环境下的Socket应用中，客户端和网络容易出现异常，此时必须释放异常退出的Socket资源。考虑到服务器的高并发能力，一般采取包接收和处理分开的策略：将接收到的包添加到包队列，然后处理队列中的数据包。当然，侦听远程客户端的连接请求可以用Socket的AcceptAsync()异步方法（IOCP，I/O完成端口由此开始）。考虑到暂停、关闭同步操作，仍然用一个线程。这样，清理资源、处理数据包、侦听客户连接请求就是组成了EMTASS架构的三个核心线程，它们由.NET线程池统一管理：

定制C# TextBox中只允
(19708)
C#实现的可复用Socket
(14947)
译文：异步Socket服务
(13319)
译文：如何使用SocketA
(11973)
ASP.NET MVC4中@mo
(11345)
定制DataGridView快捷
(9851)
带合计行的多层表头(多
(9715)
C#实现的多列数据绑定
(9222)
ASP.NET Web Api中使
(8383)

评论排行

可扩展多线程异步Socket
(245)
带合计行的多层表头(多
(36)
开源框架EMTASS完善与
(28)
.NET事件传递的实现与
(15)
定制C# TextBox中只允
(14)
译文：构建DataGridView
(10)
C#实现的多列数据绑定
(9)
技术讲座：.NET委托、
(9)
定制DataGridView快捷
(7)
C#实现的可复用Socket
(7)

推荐文章

* CSDN日报20170622——
《程序 Dog 的大梦想》

* 【Java高级开发工程师】近一个
个月的面试总结

* 一个文科生的工程师之路

* JavaWeb 与 MySQL 人情情
未了

* PermissionsDispatcher、
RxPermissions和
easypPermissions的使用和对比

* 每周荐书：架构、Scratch、
增长黑客（评论送书）

最新评论

技术讲座：.NET委托、事件及应
lyglary: 该文档“2008技术讲
座程序代码及PPT”已经失效，麻烦
更新链接，谢谢。

译文：C# 中的弱事件(Weak Ev
lyglary: 牛逼

一个C#读取DBF文件的类TDbfT
苏门答腊: 可用，非常感谢！

C#实现的表达式解析与计算类Tl
ighting_pig: left函数可以用，
right会出bug,right挺重要的，
能改一下不

一个C#读取DBF文件的类TDbfT
长沙三毛: @songja:只能读数值
型数据、字符串数据和日期数
据，不能写。但应该可以写，只
要符合格式规定即可。

一个C#读取DBF文件的类TDbfT
长沙三毛: @qq_22578335:这
个类本来就不能读Memo字段，
主要用来读数值型数据、字符串
数据和日期数据...

ASP.NET 4.0引入的视图状态属
长沙三毛: 谢谢指正！

NPOI的RemoveSheetAt()方法
长沙三毛: 另存到Excel文件，然
后把这个文件发送给打印机。

1. 客户端连接侦听线程 StartServerListen(): 循环侦听远程客户端的Socket连接请求。如果存在，通过适当规
范性判断后创建该Socket的客户端会话TSessionBase对象（实际上是该类的派生类对象），同时调用该会话
对象的Socket异步数据接收方法BeginReceive(), 接收到的数据包存放在会话对象的包队列中。当然，新增
的TSessionBase对象将添加到会话队列m_sessionTable（一个Dictionary<>泛型对象）中，该队列表就是清
理和处理线程的遍历对象；
2. 数据包处理线程 CheckDatagramQueue(): 循环检测TSessionBase队列中的会话对象
方法完成数据包解析、判断类型、数据存储等任务；
3. 会话表检测线程 CheckSessionTable(): 循环检查会话表m_sessionTable中的各个会话
已经超时、无效或异常的会话对象，清理会话对象的缓冲区，释放其Socket资源。

2) 异步处理模式

.NET Framework中的Socket具有完整的异步处理能力：侦听后异步接收（AcceptAsync()）、数据接收（BeginReceive()）、数据异步发送（BeginSend()）等。EMTASS框架采取了异步接收和发送方式
TSessionBase类中。在EMTASS的版本1.0、1.1中，这些方法在主类TSocketServerBase中实现，显然不符合类封
装原则。

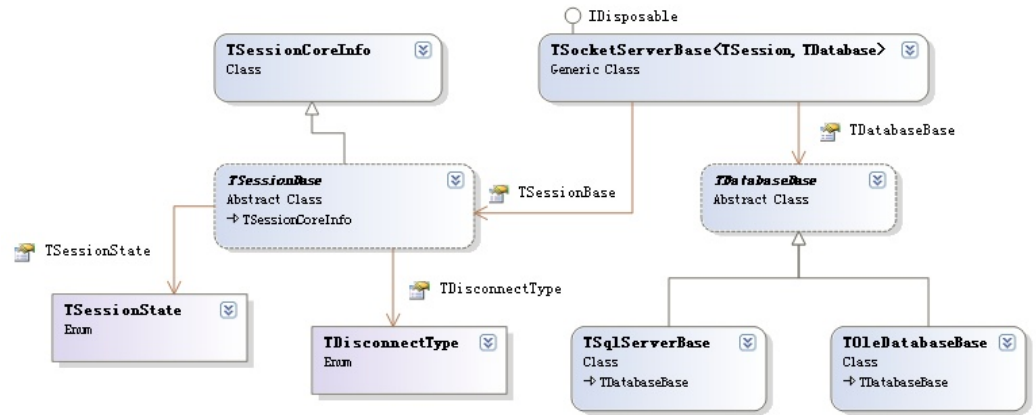
3) 系统可扩展性

可扩展性主要考虑不同的业务处理逻辑和应用场景，即：数据包格式、数据存储方法、数据库服务器等。框架
EMTASS的可扩展性体现在类的泛型与抽象设计、方法虚拟和保护等方面：

- 抽象类：会话基类TSessionBase、数据库基类TDatabaseBase均是抽象类，分别提供了数据包分析与判
断、数据存储的虚拟方法；
- 泛型类：主要基类TSocketServerBase有TSessionBase、TDatabaseBase两个泛型约束参数，可以根据这
两个抽象类的派生类产生具体的服务器类型；
- 方法抽象(abstract)：TSessionBase的数据包分析方法AnalyzeDatagram()、TDatabaseBase的数据库打开
方法Open()均是抽象的，必须在派生类中根据业务处理逻辑和数据库类型重写；
- 方法保护(protected)：与事件处理有关的方法、与业务处理逻辑相关的方法全部是protected方法，可以根据
实际情况重写。

1.2 类架构

1) 主要类层次结构



(图1 主要类层次关系)

按应用类别分，EMTASS主要有四组类：Socket服务器类、Session会话类、Database数据库类和枚举类型。

Socket服务器类

- 服务器泛型类 TSocketServerBase：该类包括了一个服务器Socket对象、一个TDatabaseBase派生
类对象、一个会话TSessionBase类派生对象的列表（Dictionary<>泛型对象），封装了
TDatabaseBase、TSessionBase的全部事件，提供统一的对外公开接口和事件；
- 泛型参数：服务器基类TSocketServerBase有两个泛型参数：TSessionBase类和TDatabaseBase类，
定制它们的派生类后可以确定泛型类的具体版本。

客户端会话类

会话核心成员类 TSessionCoreInfo：是TSessionBase的基类，包括会话的核心字段：登录时间、最
近会话时间、IP地址、客户端名、对象状态等，是会话列表清单和事件参数的基类之一。该类的成员
字段全部是protected的，需要在派生类中赋予具体值；

ASP.NET 4.0引入的视图状态属
、罪歌:

ViewStateMode="true" 应该是
是ViewStateMode="Enable"

NPOI的RemoveSheetAt()方法
xiaocnie: 请教一下: 除直接打
印Excel文件外(该问题已经通
过其他方式获得解决) 这个其它
方式是什么方式, 能指...

抽象会话类 TSessionBase : 封装了客户端Socket、数据接收缓冲区、数据包缓冲区、数据包队列等
数据结构, 包括了与客户端通信相关的全部方法: 数据接收与发送、数据包处理等。

数据库类

抽象数据库基类 TDatabaseBase : 封装了数据库打开与关闭、异常事件处理等方法, 其中数据连接
属性DbConnection是虚属性、数据库打开方法Open()是抽象方法, 需要在派生类

基类TSqlServerBase : 派生自TDatabaseBase, 应用System.Data.SqlClient名称
关类型重定义了数据库连接属性DbConnection、重写了Open()方法;

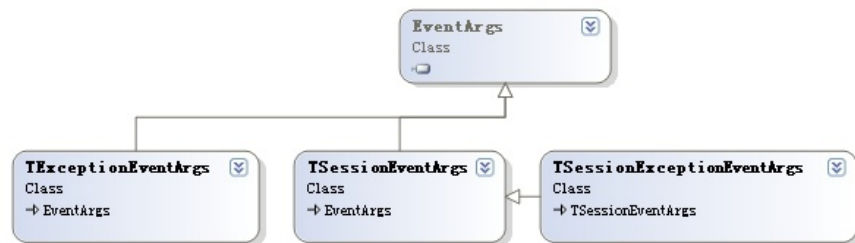
基类TOleDbDatabaseBase : 派生自TDatabaseBase, 应用System.Data.OleDb
据访问相关类型重定义了基类的连接属性DbConnection、重写了Open()方法

枚举类型

会话状态类型 TSessionState : 取4个值: Valid、Invalid、Shutdown和Closed。为Valid时表示会话是有效的, 为Invalid时表示会话将被清理, 为Shutdown时表示会话Socket正在卸载, 为Closed时表示会话已经关闭、资源已经清理;

会话断开类型 TDisconnectType : 取3个值: Normal、Timeout和Exception, 分别表示正常连接、超
时断开、异常断开。其中, 超时表示最近两次会话接收数据的时间超过约定的时限, 防止某些会话长
时间占有资源。

2) 事件参数类型结构图



(图2 事件参数类层次关系)

EMTASS框架的事件包括三类: 第一, 普通事件, 如: 服务器启动与停止; 第二, 异常事件, 接收与发送数据异常、数据库连接或数据存储异常等; 第三, 与会话相关事件, 如: 增加会话对象、接收到一个合法数据包等。异常与会话结合即是会话异常事件。通过泛型委托EventHandler可以定义类事件, 其中的事件参数类型如下:

异常事件参数类 TExceptionEventArgs : 封装了异常Exception对象的Message值;

会话事件参数类 TSessionEventArgs : 封装了一个TSessionCoreInfo对象;

会话异常参数类 TSessionExceptionEventArgs : 派生自TSessionEventArgs, 包括异常消息字段
Message。

2 关键实现技术

>>[前言]、[第1节]、[第2节]、[第3节]、[第4节]、[第5节]、[第6节]

下面介绍主要类TSocketServerBase和辅助类TSessionBase、TDatabaseBase中的主要实现方法。

2.1 TSocketServerBase类

该类包括了全部的对外接口和事件, 主要实现前面介绍过的三个线程。

1) 线程池与同步信号

.NET 提供了线程池方法 ThreadPool.QueueUserWorkItem() 自动将委托对象添加到系统线程池, 见如下的实现代码:

```

if (!ThreadPool.QueueUserWorkItem(this.StartServerListen)) return false;
if (!ThreadPool.QueueUserWorkItem(this.CheckDatagramQueue)) return false;
if (!ThreadPool.QueueUserWorkItem(this.CheckSessionTable)) return false;
  
```

其中, 客户端连接请求侦听方法StartServerListen()、数据包队列检查方法CheckDatagramQueue()和会话表检查方法CheckSessionTable()均使用循环处理方式, 循环条件是m_serverClosed为false。只有该类的Close()方法可以中断这三个线程。在Close()方法中设置m_serverClosed为true终止线程的同时, 还需要考虑线程退出的同步问题, 此时使用手工事件信号对象ManualResetEvent。参考如下数据包队列检查线程方法的代码:

```

private void CheckDatagramQueue(object state)
{
    m_checkDatagramQueueResetEvent.Reset();
  
```

```
while (!m_serverClosed)
{
    lock (m_sessionDictionary)
    {
        // ...其它代码
    }
}

m_checkDatagramQueueResetEvent.Set();
}
```

上述代码是不安全的，一般要需要try{}finally{}保证事件信号对象Reset()与Set()匹配。但EMTASS中各个方法均有自己的异常处理方式，不会抛出异常。下面是关闭服务器方法Close()的主要代码。在设置变量m_serverClosed为true后，使用了三个事件信号等待，同步三个线程的正常终止。

```
private void Close()
{
    if (m_serverClosed)
    {
        return;
    }

    m_serverClosed = true;
    m_serverListenPaused = true;

    m_checkServerListenResetEvent.WaitOne(); // 等待3个线程
    m_checkSessionTableResetEvent.WaitOne();
    m_checkDatagramQueueResetEvent.WaitOne();

    // ...其它代码
}
```

2) 分步骤的清理线程

建立会话对象后，三种情况需要终止会话：1) 关闭服务器；2) 会话异常；3) 会话超时。第1种情况将强制终止会话，第2、3种情况需要清理线程终止会话并释放其资源。为防止立即关闭Socket引发的异常，系统分3个步骤完成：1) 标记该会话为Invalid，此时停止一切与该会话的处理操作；2) 调用Shutdown()方法：Shutdown会话Socket，标记会话状态为Shutdown；3) 调用Close()方法：清除会话缓冲区和数据包队列，释放Socket资源，从会话表中删除该对象。具体操作可以参考TSocketServerBase类中的CheckSessionTable()方法。

3) 事件传递与发布

EMTASS框架的所有事件，包括TSessionBase类和TDatabaseBase类的事件，都通过服务器类TSocketServerBase对外发布。在创建会话对象或数据库对象时，直接传递其事件给TSocketServerBase的相同委托事件，见如下代码举例：

```
session.DatagramAccepted += new EventHandler(this.OnDatagramAccepted);
session.DatagramHandled += new EventHandler(this.OnDatagramHandled);
```

上述代码中，将TSessionBase派生类对象session的两个事件直接绑定（使用+=方法）到当前TSocketServerBase对象上。具体实现代码可以参考TSocketServerBase的初始化方法Initiate()和添加会话对象方法AddSession()。

2.2 TSessionBase类

该抽象类包括客户端Socket、数据接收缓冲区和数据包队列等成员，封装了所有与Socket通信的方法。该类还包括数据包处理方法：数据包解析保护方法ResolveSessionBuffer()和数据包分析虚拟方法AnalyzeDatagram()。

1) 会话缓冲区和数据包队列

TSessionBase包括两个数据接收缓冲区和一个数据包队列：

m_receiveBuffer缓冲区：接收客户端Socket数据的缓冲区，如果数据包比该缓冲区长，Socket将自动（异步）读取几次，每次用方法CopyToDatagramBuffer暂到数据包缓冲区m_datagramBuffer中；

m_datagramBuffer缓冲区：如果m_receiveBuffer接收了非完整的数据包，则使用该缓冲区暂存，直到获得一个完整数据包。一般情况下，设置m_receiveBuffer大于数据包长度，则一次可以接收一个完整包，此时该缓冲区为空。此外，该缓冲区大小根据数据包的长度动态增长；

m_datagramQueue包队列：字节数组的队列（Queue<>泛型），保存了当前会话的数
等待处理线程分析与处理。在EMTASS 1.0与1.1中，该队列结构封装在TSocketServerB
加了TSessionBase类与TSocketServerBase类的耦合程度。

2) 数据包解析方法ResolveSessionBuffer()

该方法是protected的，可以根据数据包结构与具体业务逻辑重写代码。在TSessionBase类中实现的包中规则是：开始字符是<结束字符是>。特别指出，Socket通信中有两个必须考虑的著名问题：

数据包界限问题：数据包字符串（字节数组）如何界限？在交通部的Socket通信协议中，用<开始字符>和<结束字符>包的开始与结束字符；

数据包间断与重叠问题：由于网络或设备故障，一个数据包可能分两次接收。另一种情况就是连续接收到多个数据包。第一种情况，需要先缓存接收到的包直到一个完整的数据包。第二种情况，则需要根据包界限符号分解一个个的包。

3) 数据包分析方法AnalyzeDatagram()

该抽象方法是TSessionBase类必须重写的方法，也是EMTASS框架扩展的主要接口，应该完成如下基本任务：

判断数据包的有效性 & 包类型；
分解包中的各字段数据；
校验包及其数据有效性；
发送确认消息给客户端(调用方法 SendDatagram())；
存储包数据到数据库中；
如果数据包中存在客户端名称或编号，则填写m_name字段。

2.3 TDatabaseBase类

该抽象类定义了3个数据库异常处理事件：DatabaseOpenException、DatabaseCloseException和DatabaseException，以及4个public方法：Open()、Close()、Clear()和Store()。其中，Open()是抽象方法，在派生类中可以增加自己的代码（见demo的实现部分），Close()方法关闭数据库连接，Clear()方法在Close()中被调用——关闭数据库前清理相关资源，虚方法Store()用于数据存储。EMTASS框架给出了该基类的两个派生类：TSqlServerBase和TOleDatabaseBase，可以满足一般的数据库应用需求。

3 架构使用简介

>>[前言]、[第1节]、[第2节]、[第3节]、[第4节]、[第5节]、[第6节]

3.1 一般步骤

EMTASS框架的使用包括如下步骤：

定制满足需求的TSqlServerBase或TOleDatabaseBase派生类

增加成员字段，如：DbCommand、DbDataAdapter等；
重写TDatabaseBase类的虚拟方法Store()，编写保存数据包到数据库中的实现代码；
在TSessionBase的AnalyzeDatagram()方法中直接或间接调用Store()方法。

定制满足业务处理逻辑的TSessionBase派生类

重写ResolveSessionBuffer()方法，按照数据包规则提取缓冲区中的数据包文，并存储到包队列中；
重写AnalyzeDatagram()方法，按前面的要求增加功能。

定制满足需求的TSocketServerBase派生类

定义TSocketServerBase泛型类的派生类（该步可省略）；
创建泛型类的派生类对象，在构造函数中给出数据库连接字符串和TCP通信端口；
设置泛型类的派生类对象的最大数据包长度等参数；
实现泛型类的派生类对象的相关事件处理方法。

3.2 TSocketServerBase的构造、属性、方法和事件

泛型类TSocketServerBase提供了EMTASS框架的所有对外接口（属性、方法和事件），包括内联TSessionBase对象和TDatabaseBase对象的对外属性和事件。

1) TSocketServerBase构造函数

有两个重载版本，默认端口3130。考虑到可扩展性，必须给出数据库连接串，见如下代码：

```
public TSocketServerBase(string dbConnectionString)
{
    this.Initiate(dbConnectionString);
}

public TSocketServerBase(int tcpPort, string dbConnectionString)
{
    m_serverPort = tcpPort;
    this.Initiate(dbConnectionString);
}
```

构造函数中的方法Initiate()完成具体的初始化任务。

2) TSocketServerBase公共属性

ServerPort：服务器端口号，默认值为3130
Closed：服务器已经关闭
ListenPaused：服务器暂时停止客户端连接请求
LoopWaitTime：Socket.Listen方法中的等待时间(ms)，默认值为25ms
MaxDatagramSize：允许数据包的最大长度，默认值为1024K
MaxListenQueueLength：最大侦听队列长度，默认值为16
MaxReceiveBufferSize：允许数据包接收缓冲区的最大长度，默认值为16K
MaxSameIPCount：允许同地址IP的会话Socket个数，默认值为64
MaxSessionTableLength：允许最大会话表长度，默认值为1024
MaxSessionTimeout：允许最大的会话超时间隔(s)，默认值为120s
ErrorDatagramCount：错误数据包个数
ReceivedDatagramCount：接收数据包个数
ServerExceptionCount：服务器异常次数
SessionCount：当前会话个数
SessionExceptionCount：会话异常个数
SessionCoreInfoList：当前会话表信息清单

3) TSocketServerBase公共方法

Start()：启动服务器
Stop()：关闭服务器
PauseListen()：暂停侦听连接请求
ResumeListen()：恢复侦听连接请求
Dispose()：关闭服务器并释放系统资源
CloseSession()：关闭一个会话
CloseAllSessions()：关闭全部会话
SendToSession()：给一个会话发送消息
SendToAllSessions()：给所有会话发送消息

4) TSocketServerBase事件

DatabaseCloseException：数据库关闭异常
DatabaseException：数据库异常
DatabaseOpenException：数据库打开异常
DatagramAccepted：接受了一个完整数据包
DatagramDelimiterError：数据包界限符错误
DatagramError：数据包错误
DatagramHandled：处理了一个数据包
DatagramOversizeError：数据包超长错误

`ServerStarted`: 服务器启动后
`ServerClosed`: 服务器关闭后
`ServerListenPaused`: 服务器暂停连接请求后
`ServerListenResumed`: 服务器恢复连接请求后
`ServerException`: 服务器异常
`SessionRejected`: 连接请求被拒绝
`SessionConnected`: 建立一个会话连接
`SessionDisConnected`: 断开一个会话连接
`SessionReceiveException`: 会话接收数据异常
`SessionSendException`: 会话发送数据异常
`SessionTimeout`: 会话超时

3.3 下载包Demo介绍

下载包中包括EMTASS框源代码和Demo。其中，VS2005的Demo解决方案文件为EMTASS.sln，包含两个项目：服务器项目和客户端项目。/bin/文件夹下的编译文件可直接运行：先启动服务器，然后运行客户端。

1) 服务器端Demo

服务器端包括两个部分：第一，接收服务器窗体程序；第二，Access数据库。服务器端窗体程序包含如下实现：

TSessionBase派生类TTestSession：重写了OnDatagramDelimiterError()和OnDatagramOversizeError()事件处理方法，重写了数据包分析方法AnalyzeDatagram()，增加了一个自定义方法Store()；
TDatabaseBase派生类TAccessDatabase：增加了一个OleDbCommand字段m_command，重写了Open()方法和Store()方法。在重写的Open()方法中，创建了m_command对象及其参数对象，给出了数据库的Insert语句SQL代码。重写的Store()方法中，将TSessionBase的IP、SessionName和数据包长度保存到Access数据库中。方法Store()将被TTestSession()的AnalyzeDatagram()方法调用；
TSocketServerBase<>对象：给出数据库连接字符串后，应用前面定义的两个派生类创建泛型对象，然后注册该对象的事件实现方法。注册的事件方法功能包括：显示服务器的如果计数情况，显示服务器运行状态。

服务器端的主要代码如下：

```
public partial class SocketServerDemo : Form
{
    TSocketServerBase m_socketServer;

    public SocketServerDemo()
    {
        InitializeComponent();
    }

    private void SocketServerDemo_Load(object sender, EventArgs e)
    {
        cb_maxDatagramSize.SelectedIndex = 1;

        // 数据库连接字符串
        string connStr = "Provider=Microsoft.Jet.OLEDB.4.0; Data Source = DemoAccessDatabas

        m_socketServer = new TSocketServerBase(connStr); // 服务器对象
        m_socketServer.MaxDatagramSize = 1024 * int.Parse(cb_maxDatagramSize.Text); // 包最

        this.AttachServerEvent(); // 附加服务器全部事件
    }

    private void SocketServerDemo_FormClosing(object sender, FormClosingEventArgs e)
    {
        m_socketServer.Dispose(); // 关闭服务器进程
    }
}
```

```
    }

    private void AttachServerEvent()
    {
        m_socketServer.ServerStarted += this.SocketServer_Started;
        m_socketServer.ServerClosed += this.SocketServer_Stoped;
        m_socketServer.ServerListenPaused += this.SocketServer_Paused;
        m_socketServer.ServerListenResumed += this.SocketServer_Resume;
        m_socketServer.ServerException += this.SocketServer_Exception;

        m_socketServer.SessionRejected += this.SocketServer_SessionRejected;
        m_socketServer.SessionConnected += this.SocketServer_SessionConnected;
        m_socketServer.SessionDisconnected += this.SocketServer_SessionDisconnected;
        m_socketServer.SessionReceiveException += this.SocketServer_SessionReceiveException;
        m_socketServer.SessionSendException += this.SocketServer_SessionSendException;

        m_socketServer.DatagramDelimiterError += this.SocketServer_DatagramDelimiterError;
        m_socketServer.DatagramOversizeError += this.SocketServer_DatagramOversizeError;
        m_socketServer.DatagramAccepted += this.SocketServer_DatagramReceived;
        m_socketServer.DatagramError += this.SocketServer_DatagramrError;
        m_socketServer.DatagramHandled += this.SocketServer_DatagramHandled;

        m_socketServer.DatabaseOpenException += this.SocketServer_DatabaseOpenException;
        m_socketServer.DatabaseCloseExcption += this.SocketServer_DatabaseCloseException;
        m_socketServer.DatabaseExcpetion += this.SocketServer_DatabaseException;

        m_socketServer.ShowDebugMessage += this.SocketServer_ShowDebugMessage;
    }

    //...其它代码
}
```

下面是服务器端Demo运行图片



2) 客户端Demo

创建一个TcpClient对象，模拟远程客户端与服务器通信。下面是客户端Demo运行图片：



4 一般测试结果

>>[前言]、[第1节]、[第2节]、[第3节]、[第4节]、[第5节]、[第6节]

机器配置：双核CPU E2140，主频1.6G，RAM是1G（含显卡内存）。

正确性测试

测试方法：客户端的数据包含有长度串，在服务器端检测比较，以此判断数据包传输的正确性；

测试情况：单机启动服务器，运行7个客户端程序，其中3个是干扰源——连续做连接/断开操作，另4个连续发送数据包。运行约80分钟检查，服务器平均每秒接收15个包，没有发生异常，也没有错误包，数据包队列稳定在3以下。

速度测试

测试方法：单机运行服务器，然后运行20个客户端程序，用10-50ms的速度发送数据包；

测试情况：运行30分钟检查，服务器平均每秒接收60个包，没有出现数据包队列显著增长等情况。

稳定性测试：客户端Demo中包含一个连接后马上断开的连续操作，在这种情况下服务器端没有发现异常。此外，笔者在30个客户端会话情况下运行服务器1个小时，没有发生服务器端异常，但有个别客户端异常退出。

并发性测试：同时运行了30个客户端Demo，没有发生服务器异常等现象，数据包队列上限也不超过5。

测试结论：在一台机器上做测试，EMTASS 2.0接收与处理正确，每秒可以处理60以上的数据包，运行稳定可靠，有较好的并发处理能力。测试中发现的主要问题如下：

CPU占用率很大：显然是三个线程的循环操作所引起的，在EMTASS1.0、1.1中，使用Thread.Sleep(m_waitTime)等待一段时间。本框架的设计目的是专用Socket服务器，为提高吞吐能力省略了这个操作。如果必要，在以后版本中添加该功能；

服务器启动后拒绝连接请求：特别是关闭后再启动容易发生这种现象，多关闭启动几次后又恢复正常。笔者估计是服务器m_serverSocket对象释放资源不同步的原因；

客户端连接请求被拒绝：有时客户端发生错误后，再连接时被拒绝。有两种可能，第一种是如前所讲的服务器对象的问题；第二种是客户端Socket接收、发送和断开时被阻塞，具体原因待分析。出现这种情况后，做多次连接/断开操作还是可以连接服务器。

显然，这种测试环境和结果有待进一步验证，但存在较大的改进空间。特别，数据包队列最大值一般不超过5，表明服务器接收到包后立即处理，并发性能比较好。

5 总结与展望

>>[前言]、[第1节]、[第2节]、[第3节]、[第4节]、[第5节]、[第6节]

本文介绍的EMTASS 2.0 是笔者一段工作的总结，也是学习基于.NET的类设计、组件设计、模式设计等的一个小结。笔者的目标就是不断修改和完善，设计与实现一个可靠与稳定的、有良好可扩展性的和易于使用的Socket数据包接收服务器框架。由于最初的代码和思路均来自他人的开源架构和设计构思，EMTASS也仿效一般开源做法：公布源码和设计思路。

基于EMTASS 1.0的服务器有连续运行30天完全正常的记录，而框架EMTASS 2.0 虽然具有可扩展性，也进行了一般的测试，但没有投入实际运行，需要时间检验和实践验证。当前，.NET 3.0及3.5 Framework提供的IOCP（完成端口）具有更好的异步并发处理能力，笔者将结合新的运行平台，完善与升级EMTASS，并公布完善与升级计划。如果有读者使用EMTASS 2.0时发现问题，或有更好的建议或想法，请不吝指正。

6 版本与源码

>>[前言]、[第1节]、[第2节]、[第3节]、[第4节]、[第5节]、[第6节]

EMTASS 1.0, 2005年12月

EMTASS 1.1, 2008年09月

关闭

EMTASS 2.0, 2008年10月27日

重设计了类名及类关系, 增加了框架的可扩展性, 重构了大部分代码

TSessionBase的数据包队列取代TSocketServerBase的数据包队列

TSessionBase中封装了全部通信方法

EMTASS 2.1, 2008年11月9日

增加了缓冲区管理类BufferManager, 管理两个可重复使用的发送/接收缓冲区。

TSocketServerBase增加了如下属性:

ReceiveBufferSize: 接收缓冲区大小 (默认16K)

SendBufferSize: 发送缓冲区大小 (默认16K)

CheckDatagramQueueTimeInterval: 数据包处理线程Sleep时间间隔 (默认: 100ms)

CheckSessionTableTimeInterval: 会话资源清理线程Sleep时间间隔 (默认: 100ms)

TSocketServerBase增加了若干构造函数, 包括最大任务数和缓冲区大小




TSocketServerBase使用了Mutex互斥类, 防止同机器创建两个服务器

TSessionBase在异步接收/发送完成后, 调用IAsyncResult.AsyncWaitHandle.Close()方法

TSessionBase接收数据时, 使用BufferManger公共缓冲区ReceivevBuffer

TSessionBase发送数据时, 如果据长度小于发送缓冲区, 则使用SendBuffer, 否则申请字节数组发送

测试结果: 10个干扰客户端 (100ms连续连开) 与15个数据客户端 (3个100K/100ms、3个100K/50ms、4个100K/20ms、2个1M/1s、2个1M/500ms、1个1M/10s), CPU占用率为70-90%, 速度为40/s, 无错误包, 直接显示的连接数为30-50之间

-  下载EMTASS 1.1源码: 1) [csdn注册用户](#)、2) [国外网直接下载](#)
-  下载EMTASS 2.0源码及demo: 1) [csdn注册用户](#)、2) [国外网直接下载](#)
-  下载EMTASS 2.1源码及demo: [csdn用户 \(有bugs\)](#)、2) [国外网直接下载 \(修改了bugs和BufferManager类\)](#)

顶

0

踩

0

上一篇

[.NET事件传递的实现与事件链问题](#)

下一篇

[道路运输行业安全报表与分析软件YGQA: 运行维护](#)

相关文章推荐

- [C#通信二]C#.net同步异步SOCKET通讯和多线...
 - .NET 新手小组 / 最近整理了一份C# WinForm开...
 - 可扩展多线程异步Socket服务器框架EMTASS 2.0...
 - 可扩展的多线程通用Server框架
 - 使用 Java 构造高可扩展应用,如何实现一个高效且...
- 多线程Socket编程实现服务器与客户端的连接
 - [转载]C#实现的可复用Socket接收/发送共享缓冲...
 - C#.net同步异步SOCKET通讯和多线程总结
 - C# Winform 开发资料汇总
 - 可扩展多线程异步Socket服务器框架EMTASS 2.0...


猜你在找

- 深度学习基础与TensorFlow实践
- 【在线峰会】前端开发重点难点技术剖析与创新实践
- 【在线峰会】一天掌握物联网全栈开发之道
- 【在线峰会】如何高质高效的进行Android技术开发
- 机器学习40天精英计划
- Python数据挖掘与分析速成班
- 微信小程序开发实战
- JFinal极速开发企业实战
- 备战2017软考 系统集成项目管理工程师 学习套餐

Python大型网络爬虫项目开发实战（全套）

查看评论


156楼 [ybbhwxhj](#) 2015-02-03 13:50发表

 谢谢分享,我看了Demo,示例运行发送数据后报错，异步调用控件错误，这个需要修改

155楼 [X30513804](#) 2014-04-10 12:59发表

 mark

154楼 [色郎中](#) 2014-01-14 09:29发表


 没把UDP 包括进去呀？
测试了下，，

1 数据包随机上限 选 “10000” K 的时候会出问题


153楼 [shorlyn](#) 2013-10-26 15:49发表

 胡老师，各位网友，能不能教教我在客户端连接的时候SessionName怎么赋值，就是TSessionCoreInfo中的Name，求求了


Re: [长沙三毛](#) 2013-12-08 10:31发表

 回复shorlyn：看源代码。


152楼 [SPTC](#) 2013-09-27 16:43发表

 你好，服务器端的缓冲区管理好像有点小错误。发送缓冲区的可用下标也是从 GetBufferBlockIndex（）获得，但是向 m_bufferBlockIndexStack中push 的值是接收缓冲区中的可用可回收的缓冲区。在Tsession类中，使用的发送缓冲区的下标实际是接收缓冲区的下标。
不知道是楼主没注意，还是我没理解，看错了？


151楼 [Cfreezhan](#) 2013-07-24 13:33发表

 给胡老师赞一个！


150楼 [yang_xz](#) 2013-05-10 15:18发表

 博主还关注这篇博文吗？感谢博主分享，使用中发现一个问题，TCP客户端（非博主演示客户端）连接上演示服务端，定时发送数据包，此时服务端点击停止，会话不会中止，仍然持续接收数据，客户端仍然认为服务正常，应怎样解决？


Re: [长沙三毛](#) 2013-05-23 23:32发表

 回复yang_xz：只停止了接收,没有停止处理,也没有停止已经连接的任务.


149楼 [DFSCL](#) 2013-05-06 16:08发表

 线程间操作无效: 从不是创建控件“tb_SessionCount”的线程访问它。一开始就这样


148楼 [zhouyicheng](#) 2013-04-08 20:50发表

 对数据的处理速度度还是有点慢,主要原因可能是:当网速不是那么快时,发送的多个数据包会粘在一块,一次性被接收,导致处理不及时


147楼 [qq455800175](#) 2013-04-02 10:06发表

 请问下，对于服务器多端口监听，有什么建议和方向吗？
请不吝赐教，坐等~~~~


Re: [长沙三毛](#) 2013-05-23 23:33发表

 回复qq455800175：原来考虑多端口监听的问题,发现处理难度不小.后来想了想,可以做多个服务器来达到同样的目的.

146楼 [glutton](#) 2013-01-06 11:31发表

 楼主好，我现在一个项目采用了您的框架，先谢谢了，目前有个一个问题，看能不能帮我看看
一个客户端，1s发送100个包，每个大约4K左右，会频繁出现错误包，我该怎么改进呢，谢谢。
补充下：服务端触发的是OnDatagramDelimiterError，对方采用的是发送队列，这100个包可能一次性发出

Re: [长沙三毛](#) 2013-05-23 23:35发表

 回复EonianGlutton：从我自己的测试结果看,如果有包分隔符号,虽然有粘包现象,但可以很好地分解处理.

145楼 [viki117](#) 2012-11-13 10:05发表



多个socket之间如何协调通信?写了怎么多的socket,我一直总结不出通用的socket方法,通信协议和业务逻辑的差异,除了socket的收发数据,和数据缓存后,开线程处理数据,其他的完全没办法雷同,就这么点可以通用的东西,也就几百行代码,完全没必要...

144楼 [dgh_85](#) 2012-11-02 15:06发表



您好,我用HP LoadRunner对框架进行500个模拟用户进行测试,数据接收就很慢了,感觉就像是在排队。不知道在数据接收的时候还有没有更好的方法。谢谢。

Re: [长沙三毛](#) 2013-05-23 23:36发表



回复dgh_85: 如果不考虑数据库读写,速度应该快点。

143楼 [helinherong_fan](#) 2012-10-10 11:16发表



看了下demo,楼主根本没有考虑跨线程访问吗?异步消息本分根本无法调试。还有个问题,我觉得数据时,数据保存在成员变量m_receiveBuffer 中,这样难道不会有很大几率数据会覆盖吗?

Re: [长沙三毛](#) 2012-10-15 22:44发表



回复helinherong_fan: 每个连接均创建一个Session对象,不存在覆盖问题

Re: [helinherong_fan](#) 2012-10-16 14:59发表



```
回复hulihui: if (ck_AsyncReceive.Checked) // 异步接收回答
{
    m_socketClient.Client.BeginReceive(m_receiveBuffer, 0, m_receiveBuffer.Length,
    SocketFlags.None, this.EndReceiveDatagram, this);
}
/////然后再EndReceiveDatagram中
private void EndReceiveDatagram(IAsyncResult iar){
    * * * * *
    this.CheckReplyDatagram(readBytesLength);
}
/////然后再CheckReplyDatagram中
private void old_CheckReplyDatagram(int len)
{
    * * * * *
    string replyMessage = Encoding.ASCII.GetString(m_receiveBuffer, 0, len);
    * * * * *
}
这里能确保不会覆盖m_receiveBuffer吗?
异步通讯,以前没搞过,现在一头雾水了。
```

Re: [helinherong_fan](#) 2012-10-16 14:31发表



回复hulihui: 我是说客户端的demo。既然是异步的,就不能确保同一时间只有一个线程在使用这个buffer吧。
服务端还没来的急拜读,不过我仍然觉得会有问题,和客户端是一样的:虽然m_receiveBuffer是session的成员,但是每个session都是异步接收数据。

142楼 [lonet](#) 2012-07-24 14:55发表



我想咨询一下AnalyzeDatagram是多线程处理的还是一个一个处理数据包呢?

Re: [长沙三毛](#) 2012-10-15 22:46发表



回复lonet: 多线程其实是按时间片轮询的,实质是一个数据包一个数据包地处理。

141楼 [lisheng831010](#) 2012-06-17 13:45发表



您好,最近研究了EMTASS,请问一下,如果我想在该框架下,加一个心跳监测线程,请问,可以直接用ThreadPool.QueueUserWorkItem起一个线程,来轮询所有的client客户端吗?

140楼 [steven_rong](#) 2012-05-14 11:26发表



感谢楼主分享。
请问,出现 150楼 错误问题怎么解决?
开发环境: XP SP3 ,.NET F 2.0 SP2 , VS2005
服务器运行环境: win2008 r2 x64

目前服务接入 200 左右的客户端,服务运行 一两天后就会出现上述问题。。。。
请各位帮帮。。。。

Re: [steven_rong](#) 2012-05-14 11:36发表



回复steven_rong: 而且我从网上查了,如果在 App.config 中加入:
<runtime>
<legacyUnhandledExceptionPolicy enabled="true" />
</runtime>

那服务器运行一段时间后就会，只看到 客户端 一直在 尝试连接 (connect) 和断开连接(disconnect)的 info 信息打印出来，感觉是服务器，没有对客户端的连接请求回应，而超时断开的。

139楼 [steven_rong](#) 2012-05-14 11:22发表



[plain]

- 01. EMTASS ServerDemo 已停止工作
- 02. Windows 可以联机检查该问题的解决方案
- 03. 一》联机检查。。。。关闭程序
- 04. 一》关闭程序
- 05. 问题事件名称: CLR20r3
- 06. 问题签01:emtass_serverdemo.exe
- 07.
- 08. 问题签名08:400
- 09. 问题签名09:System.IndexOutOfRangeException
- 10. OS版本: 6.1.7600.2.0.0.274.10
- 11. 区域设置ID:2052

138楼 [jivi](#) 2012-04-26 15:01发表



FE FE FE FE 68 33 01 00 01 40 00 68 0F 87 01 01 3C 43 43 6D 64 31 32 39 3E 3C 4D 65 74 65 72 4D 61 6E 61 67 FE FE
FE FE 68 33 01 00 01 40 00 68 0F 87 01 01 3C 43 43 6D 64 31 32 39 3E 3C 4D 65 74 65 72 4D 61 6E 61 67 65 72 49 64
3E 31 3C 2F 4D 65 74 65 72 4D 61 6E 61 67 65 72 49 64 3E 3C 55 73 65 72 49 64 3E 30 3C 2F 55 73 65 72 49 64 3E 3C
4E 65 77 4D 65 74 65 72 49 64 3E 30 3C 45 51 53 68 6F 75 6C 64 65 72 3E 30 2E 30 3C 2F 45 51 53 68 6F 75 6C 64 65
72 3E 3C 45 51 4F 66 66 50 65 61 6B 3E 30 2E 30 3C 2F 45 51 4F 66 66 50 65 61 6B 3E 3C 2F 43 43 6D 64 31 32 39 3E
09 16

利用你得框架改写成能够接收这样得数据，我用delphi已经实现在c#里面目前没有实现

137楼 [kkylove](#) 2012-03-14 18:27发表



暂时没看懂！很高深

136楼 [唯美德](#) 2011-11-08 10:07发表



受教了，非常好的框架。
多谢博主分享。
“国外网直接下载（修改了bugs和BufferManager类）”已经不能下载了，博主能否提供一个新的链接。谢谢。

135楼 [idaydayup](#) 2011-09-17 08:58发表



请问这个方法该怎么重写才能实现自定义的数据包呢？
protected virtual void ResolveSessionBuffer(int readBytesLength)
规则不一样时不重写的话就会出错,重写时又不知道该在内部实现什么,写个空方法就会导致假死

134楼 [justyinhan](#) 2011-09-14 16:29发表



这个方法主要是看到里面的逻辑,是使用"<>"来区分一条数据的,但如果,有2条数据合成一个tcp包发送过来,如下这种情况
<C05016,0000000020,><C05016,0000000020,>,好像就有问题了

Re: [长沙三毛](#) 2011-10-06 22:48发表



回复justyinhan：这个是没有问题的,有一个接收缓冲区,没有解析完的包,在下个结束符号前是不丢的

133楼 [justyinhan](#) 2011-09-14 16:24发表



感谢博主,最近项目正用到这方面东西,有一个问题,对于沾包处理的逻辑在哪块? 是ResolveSessionBuffer吗?

132楼 [wangli820](#) 2011-09-06 10:35发表



楼主，非常感谢共享这么好的资源，对我帮助非常大。
收发消息的时候，只能用字符串吗？能不能支持二进制消息，毕竟很多时候要发送结构体等消息

Re: [长沙三毛](#) 2011-10-06 22:49发表



回复wangli820：直接处理二进制流,不好判断包界限.事实上,你可以把二进制做转换后发送.

131楼 [lly888333](#) 2011-08-13 14:09发表



什么破网。。。根本就不能下载！！！！

130楼 [xuweije](#) 2011-07-31 21:01发表



外网下载不了。。。

129楼 [无牛刀](#) 2011-07-20 11:38发表

里面的程序有问题，在启动线程池的时候，while 里面增加Thread.sleep(100); cpu 才不会占用得太高



128楼 [steven_rong](#) 2011-07-19 11:25发表



谢谢分享,我看了Demo,示例中数据是以 "<*****>"这样的格式传输的,那请问各位,如果报文是一行一行,也就是以行符("\n")结尾的呢,该例如处理ResolveSessionBuffer 的???

```
/// <summary>
/// 提取包时与包规则紧密相关，根据实际规则重定义
/// </summary>
protected virtual void ResolveSessionBuffer(int readBytesLength)
```

127楼 [steven_rong](#) 2011-07-14 13:17发表



引用“zyszhang”的评论：_____

我在服务器上测试了一小会，客户端设备接有487台（GPS），发现包队列长度达到880多个，[e08]

首先谢谢分享,我也准备将它用于GPS报文接收处理服务,请问哪位有这个框架的,针对这一服务的研究啊?学习学习//

zyszhang 现在你的什么情况了?

126楼 [lx313313](#) 2011-07-14 13:06发表




楼主您好，如果我想要让服务器在接收到数据以后将数据中的某项信息在form中的一个listview中刷新显示的话，我应该怎样做呢？我是一刚开始工作的新手，希望不要嫌我的问题太肤浅哦……

查看更多评论

发表评论

用户名：[github_38757220](#)

评论内容：



提交

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场