文章编号： 1000-5013(2011)02-0178-04

利用 WIN32 动态链接库的 TFTP 服务器的设计与实现

姜林美

(华侨大学 计算机科学与技术学院，福建 泉州 362021)

摘要： 为了实现在设备相关的桌面应用配置软件中嵌入简单文件传输协议(TFTP)服务功能，开发针对嵌入式系统的 FLASH 烧录及其配置和文件升级的接口程序. 考虑到软件复用的需要，采用 WIN 32 动态链接库技术，整个 TFTP 服务模块仅由一个动态链接库文件 tftp. dll 组成. 使用多线程技术，以适应 TFTP 收发过程与应用程序的其他过程并发执行的需要. 将所述方法应用于实际的路由器配置产品中，并验证其有效性.

关键词： 简单文件传输协议；用户数据包协议；传输控制协议/因特网互联协议；动态链接库；多线程；文件传输；嵌入式系统

中图分类号： TP 393. 093 **文献标志码：** A

简单文件传输协议(TFTP)在各类嵌入式产品中应用广泛. 在这些嵌入式产品中，主要使用 TFTP 来代替 JTAG 方式的 FLASH 烧录[1-2]，以及使用 TFTP 实现软件或配置的在线更新[3]. 通常，网络服务器总是在装有网络操作系统(如 Unix，Windows 2003 等)的专用服务器设备中实现，而此种方式具有设备移动不方便，服务器需要有人值守，以及网络服务安全需要特殊考虑等弊端，故不适用于 TFTP. 对于嵌入式设备而言，往往需要在设备和小型桌面应用软件中嵌入 TFTP 文件传输功能. 即使用一个专用的桌面应用配置软件与设备进行通信，以完成 BOOT 文件、映像文件或配置文件的传输，而不是使用专用的主机设备来充当 TFTP 服务器的角色. 因此，开发一个通用的 TFTP 传输模块，以供任意桌面应用软件在需要时调用就显得很有价值. 本文讨论在 Visual C^{++} 2005(VC 2005)环境下，使用 WIN 32 动态链接库[4]实现 TFTP 服务器的一种方法.

1 数据结构

1.1 接口

使用两个接口 ISink 和 ITftp，来隔离 TFTP 服务器的实现和使用，其定义如图 1 所示.

```
// 回调事件接口
struct ISink
{
    virtual HRESULT OnTftpRecvOver(LPCTSTR strIp, LPCTSTR strFilename) = 0;
    virtual HRESULT OnTftpSendOver(LPCTSTR strIp, LPCTSTR strFilename) = 0;
    virtual HRESULT OnTftpError(LPCTSTR strIp, LPCTSTR strFilename, int iErrorCode) = 0;
};

// Tftp服务器接口
struct ITftp
{
    virtual HRESULT Start(USHORT uPort) = 0;
    virtual HRESULT Stop() = 0;
    virtual HRESULT Advise(ISink* pSink) = 0;
    virtual HRESULT SetRootDir(LPCTSTR strDir) = 0;
};
```

图 1 接口 ISink 和 ITftp 的定义

Fig. 1 Definitions of interface ISink and ITftp

收稿日期： 2009-09-18

通信作者： 姜林美(1976-)，男，讲师，主要从事计算机网络安全及应用软件开发的研究. E-mail：clough@hqu. edu. cn.

在 C++ 中，接口的定义是使用结构体加纯虚函数来完成的. ISink 接口由应用程序实现，并被传递给 DLL. 在 DLL 中，当适当事件发生时就会调用接口中相应的事件方法. OnTftpRecvOver()方法在文件接收(即上传)完毕时被调用，OnTftpSendOver()方法在文件发送(即下载)完毕时被调用，OnTftpError()方法则在文件接收或发送出错时被调用. 3 个方法的参数 strIp 和 strFilename 用于指明是哪一个连接(一个 TFTP 服务器可以同时与多个客户端建立连接)触发的事件，OnTftpError()方法中的参数 iErrorCode 给出错误发生时的错误代码，方便在应用程序中对出错情况进行处理.

ITftp 接口由 DLL 实现，是 TFTP 服务器模块与外层应用程序交互的唯一接口. 其中：Start()方法用于在参数 uPort 指定的端口上启动 TFTP 服务，Stop()方法则用于停止 TFTP 服务并释放所有资源；Advise()方法用于建立一个外层应用中实现的 ISink 接口与 DLL 中 TFTP 服务的一个连接，其唯一参数 pSink 用于传入外层应用中实现的 ISink 指针；SetRootDir()方法设置 TFTP 服务器文件放置的根目录，参数 strDir 为目录名.

```
// 确认报文
struct TAck
{
   WORD   wOpCode;                    // 操作码
   WORD   wBlock;                     // 块号
};
// 通用报文
struct TPacket
{
   WORD   wOpCode;                    // 操作码
   WORD   wBlock;                     // 块号
   char   szData[TFTP_DATA_SIZE];     // 数据
};
// 差错报文
struct TTfTpError
{
   WORD   wOpCode;                    // 操作码
   WORD   wErrorCode;                 // 错误码
   char   szErrorMsg[TFTP_ERROR_MSG_SIZE + 1];  // 错误信息
// 构造函数和复制函数
   TTfTpError() {}
   explicit TTfTpError(WORD wErrorCode, const char* pMsg=NULL);
   TTfTpError& operator = (const TTfTpError& err);
};
// TFTP 连接
struct TRequest
{
   char              szMapName[MAP_NAME_SIZE + 1]; // 连接映射名，含 IP 和端口号
   enum {EN_UP, EN_DOWN}    enUpDown;             // 上传或下载
   enum {EN_ASCII, EN_OCTET} enMode;              // 模式
   int               iReqId;                      // 连接标识
   char              szFileName[MAX_FILENAME_SIZE + 1];  // 文件名
   FILE*             fp;                          // 文件指针
   int               iBlock;                      // 块号
   time_t            tmExpiry;                    // 发送时间，用作超时判断
   int               iAttemptTimes;               // 重发次数
   sockaddr_in       sockPeer;                    // 客户端
   int               iSockSize;                   // 客户端结构大小
   TTfTpError        err;                         // 错误报文
   int               iBytesRecv;                  // 最后收到字节数
   int               iBytesSend;                  // 最后发送字节数
   TPacket           packetRecv;                  // 最后收到的 TFTP 报文
   TPacket           packetSend;                  // 最后发出的 TFTP 报文
};
```

图 2　处理 TFTP 报文的数据结构注释

Fig. 2　Comments of the data strutures for processing TFTP datagrams

1.2　TFTP 报文

相应于 TFTP 报文格式，采用几个数据结构来处理 TFTP 报文，其定义如图 2 所示. 各个结构体中的字段含义由各字段后的注释给出. 使用 1 个类 CRequests 来管理所有的 TFTP 连接，其定义如图 3 所示. 图 3 中：最后一行的 TRequest 类型使用标准模板库中的 map 容器定义：typedef std::map〈std::string, TRequest * 〉 TRequests.

```
class CRequests
{
public:
   CRequests();// 构造函数
   ~CRequests(); // 析构函数，关闭所有连接关联的文件和 SOCKET，并释放所有连接占用的资源
   BOOL IsExists(std::string strMapName);          // 判断连接是否已存在
   TRequest* GetReq(std::string strMapName);       // 取一个连接
   void AddReq(const TRequest* pReq);              // 添加一个连接
   TRequests::iterator RemoveReq(const TRequest* pReq); // 删除一个连接
   TRequests::iterator RemoveReq(std::string strMapName);// 删除一个连接
   TRequests::iterator Begin();                    // 取第一个连接的迭代
   TRequests::iterator End();                      // 结束迭代
   // 使用谓词遍历所有连接
   void Traverse(UnaryFuncRequest<TRequests::iterator*>* pOp, PredRequest* pPred);
private:
   TRequests m_requests;                           // 保存 tftp 连接的 map 结构
};
```

图 3　类 CRequests 的定义

Fig. 3　Definition of class CRequests

1.3　TFTP 实现类

使用 CTftpS 类来具体实现 TFTP 服务器的功能. 该类从接口 ITftp 继承，实现声明的所有方法，

其定义如图4所示. TFTP服务器实现代码均封装在CTftpS类中,因此该类的实现主要是ITftp接口声明的4个方法的实现.

(1) Start()方法用于启动TFTP服务,进行SOCKET库的实始化并创建服务器端SOCKET,让其监听在指定的端口(默认端口号为69). 然后,对类变量(包括线程同步变量)进行初始化. 最后,创建TFTP收发线程,专门负责TFTP连接的建立和TFTP报文的接收与发送,其入口函数即CTftpS类的静态成员函数ThreadFunc(). 在创建的线程的时候,CTftpS类的实例指针将作为线程的参数传入ThreadFunc().

```
class CTftpS : public ITftp
{
public:
  // ITftp接口声明的4个方法(略)
private:
  static unsigned WINAPI ThreadFunc(void* pParam);// TFTP收发线程函数
  void ElapseTime();                    // 判断是否超时,决定是否重发
  BOOL RecvData(TRequest* pReq);        // 接收数据
  BOOL ProcessNew(TRequest* pReq);      // 处理新的连接请求
  void SendFileContent(TRequest* pReq); // 发送文件内容
  int SendTo(/*参数略*/);               // 通用发送函数
  void SendErrorMsg(/*参数略*/);        // 发送错误信息
  void TerminateConn(const TRequest* pReq, int iErrorCode); // 中断一个连接
private:
  TThread          m_thread;            // TFTP收发线程句柄
  HANDLE           m_hEventEndThread;   // 线程结束事件句柄
  CRITICAL_SECTION m_cs;                // 临界区变量
  CRequests        m_requests;          // 连接请求
  SOCKET           m_socketLocal;       // 本地SOCKET
  int              m_iLastError;        // 最后的错误码
  char *           m_pRootDir;          // TFTP服务的根目录
```

图4 CTftpS类的定义

Fig. 4 Definition of class CTftpS

(2) Stop()方法用于停止TFTP服务,要触发停止TFTP收发线程的事件并等待该线程退出,最后释放所占用的各种资源. 另两个方法的实现很简单,限于篇幅不赘述.

2 TFTP服务的实现

TFTP服务实现的关键在于TFTP收发线程函数ThreadFunc(),其处理流程图如图5所示. 在线程函数中,通过线程函数的参数取得CTftpS对象的指针,而这只要通过对函数参数进行一个类型强制转换即可完成. 线程的主循环具有一个唯一的退出条件,即退出线程事件(事件句柄为CTftpS::m_hEventEndThread)在主线程中被触发,通过WINDOWS API函数WaitForSingleObject()可以捕获该事件.

在线程的主循环内,调用CTftpS::ElapseTime()方法扫描当前已建立的所有用于下载文件的TFTP连接,比较当前时间与连接记录(TRequest类型变量)的tmExpiry字段的差值. 如果该差值超过了设定的超时时间(比如3 s),则重发上一个数据报文,检查连接记录的iAttemptTimes字段;而如果该字段的值(重发次数)大于3,则触发ISink接口的OnTftpError()事件并挂断连接.

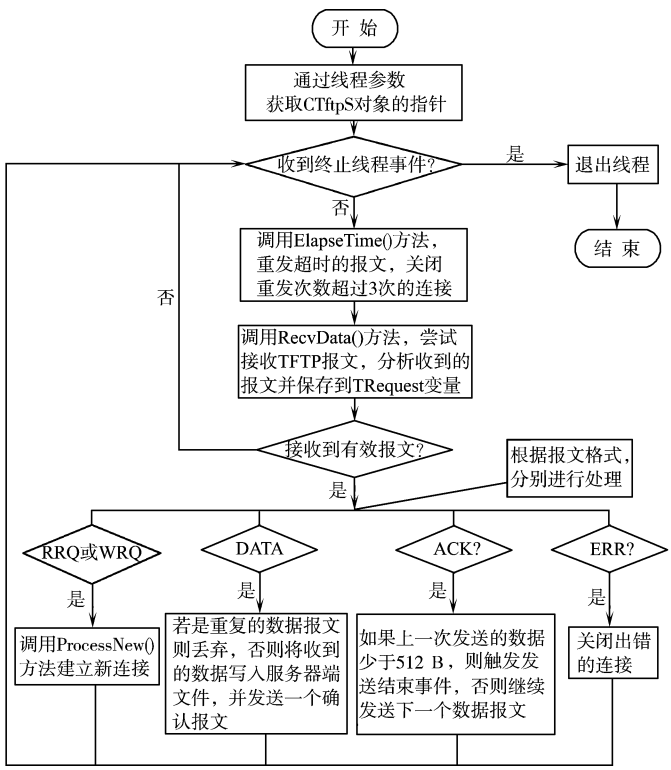

图5 TFTP线程函数处理流程

Fig. 5 Program flow in TFTP thread function

调用CTftpS::ElapseTime()方法后,继续调用CTftpS::RecvData()方法尝试接收TFTP报文. 通过对收到的有效报文进行解析,把解析结果保存在一个TRequest类型变量中,并将该结果通过指针类型参数返回给线程函数. 如果通过CTftpS::RecvData()方法接收到任何有效的TFTP报文,则根据报文格式的不同分别进行处理.

对于RRQ或WRQ报文,需要调用CTftpS::ProcessNew()方法建立一个新的TFTP连接. 建立

连接的过程是：首先从报文中获取文件名及模式，在设定的服务器根目录下建立一个相应的文件．其次，对于 RRQ 请求需要读取文件的头 512 B，并将其发送给客户端；而对于 WRQ 请求，则直接发送一个块号为零的确认报文．最后，生成一个 TRequest 数据，并将其加入 CTftpS:: m_requests 记录的连接请求列表中.

对于 DATA 报文，先判断该报文是不是由于超时重发而形成的重复报文，若是则直接发回一个块号与所收报文块号相同的确认报文；否则，将收到的数据写入服务器端文件，然后发回相应的确认报文.最后，如果收到的数据少于 512 B，说明已是最后一个数据报文，文件传送完毕，触发 ISink 接口的 OnTftpRecvOver()事件，关闭该 TFTP 连接.

对于 ACK 报文，首先判断块号与上一次发送的数据报文的块号是否一致.如果不一致，说明是一个已处理过的重复报文，可忽略该报文；如果一致，则继续判断上一次发送的数据报文的数据长度是否小于 512 B，小于 512 B 说明已是最后一块数据，文件传送完毕，触发 ISink 接口的 OnTftpSendOver()事件，关闭该 TFTP 连接.

对于 ERR 报文处理比较简单，直接触发 ISink 接口的 OnTftpError()事件并挂断连接.TFTP 线程中调用的其他 CTftpS 类的方法的实现均较为简单，比如 TerminateConn()方法用于关闭连接，其操作就是简单地关闭相应的 SOCKET，并从 CTftpS:: m_requests 记录的连接请求列表中删除该连接对象.

3　结束语

所讨论的 TFTP 服务器模块，最初基于某公司的 3G 路由器产品的桌面配置软件而开发，并最终应用于该产品．实践证明，其应用方便，用于产品运行稳定可靠.需要说明的是，由于协议的简单性，TFTP 没有提供任何安全方面的特性[5].

参考文献：

[1]　汪小燕，连晓平，黄燕，等.基于 TFTP 协议的嵌入式系统开发方法设计与实现[J].华中科技大学学报：自然科学版，2006，34(12)：56-58.

[2]　李虔华，臧习飞.TFTP 协议在嵌入式系统中的应用[J].电脑与电信，2008(2)：40-42.

[3]　甘育裕，李婷.在嵌入式 Linux 中实现 TFTP Server[J].中国有线电视，2005(Z3)：4945-4949.

[4]　郭晓鹏，李存斌.Vicsual C^{++}高级编程及其项目应用开发[M].北京：中国水利水电出版社，2004.

[5]　RICHARD S W.TCP/IP Illustrated：Volume 1[M].Boston：Addison Wesley，1995.

Design and Implementation of TFTP Server Based on WIN 32 Dynamic Link Library

JIANG Lin-mei

(College of Computer Science and Technology, Huaqiao University, Quanzhou 362021, China)

Abstract: To add a TFTP (trivial file transfer protocol) service to a device-related desktop configuration software, a program interface targeted at the flash burning and configuration upgrade of embeded systems is developed. Considering the requirement of the software reusablity, the WIN 32 DLL (dynamic link library) technology is used and the whole TFTP module is encapsulated in one DLL file (tftp.dll). In addition, to satisfy the requirement of concurrent running of the TFTP process and the other process in an application, the multi-thread technology is used. Finally, the validity is proofed by implementing this scheme in an configuration software of a router product.

Keywords: trivial file transfer protocol; user datagram protocol; transmission control protocol/internet protocol; dynamic link library; muti-thread; file transfer; embeded system

(责任编辑：黄晓楠　　**英文审校：**吴逢铁**)**