VoIP 技术的一种核心框架实现

陈彧晖，王六凤，邹丰美
（厦门大学计算机科学系人工智能所　福建 厦门　361003）

【摘　要】：本文设计了 VoIP 客户端系统的整个架构。通过分析 VoIP 的基本原理，针对 Windows 操作系统的特点，阐述 VoIP 客户端的实现中几个模块的设计关键点。实验结果表明该框架一直表现良好，用户界面的响应友好，在选择合适的算法后能适应较为恶劣的网络环境，能很好的部署到有 VoIP 业务需求的软件中。

【关键字】：VoIP，消抖缓冲，多线程

1. 引言

20 世纪 90 年代中期，伴随着 Internet 的大范围普及，VoIP 作为 Internet 上的联机应用诞生了。VoIP 就是指应用于 IP 网络上实现话音及传真信号传输的一门全新的集成业务数据网络技术，双方只要同时拥有相同的客户端通信软件，就可以在 Internet 上进行实时通话。为了让 VoIP 更方便的运用到需要该业务的软件中，本文设计了 VoIP 客户端核心系统框架，具体描述了构架框架时遇到的关键技术。

2. VoIP 客户端的系统结构

VoIP 客户端主要涉及到三大部分的交互：网络模块、用户界面模块、语音模块。三个模块分别处理来自不同媒介的信息，并统一由 VoIP 核心控制模块调度管理，在相互配合下完成有效的通讯过程。为了减小网络带宽开销，提高 VoIP 的服务质量，并满足不同网络环境的需求，语音模块还包含了编码与解码的模块，及消抖缓冲模块。图 1 是 VoIP 客户端的系统结构图，它说明了整个系统的各个界面组成及相互之间控制指令以及数据流动关系。其中带斜线的箭头表示指令数据，灰色箭头表示网络数据包，白色箭头表示原始语音数据。

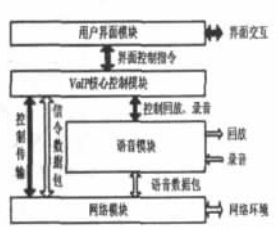

图 1 VoIP 客户端系统模块结构图

3. 模块设计关键点

3.1 VoIP 核心控制模块

VoIP 核心控制模块为其余三个模块提供统一的操作接口，该模块实现为 TVoIPSession 虚基类。在框架设计时，该模块规范统一了个界面的调用操作方式，整个系统的模块化提供了有效的保证；同时该模块为网络模块的操作，语音模块的操作提供有效封装，为界面模块提供了更简单的操作接口，使得替换界面模块变得更加容易，只需简单的继承，就能很容易的将本文设计的 VoIP 内核整合到各种商业软件中。在核心控制模块中，也模仿传统 PSTN 电话机，将系统设计成四个状态：离线，在线，呼叫和通话。四个状态的转换关系如图 2：

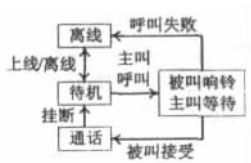

图 2 VoIP 客户端状态转换图

由于本系统涉及到网络传输、音频录制回放，有些 API 不能很迅速返回。为了给前台用户界面模块提供更好的相应能力，VoIP 核心控制模块自含一条线程，该线程功能在于接收用户界面模块的操作消息，并指挥其他模块做相应操作。由于在操作消息确认到达之后，用户界面模块即可马上从调用返回，因此兼顾了操作的正确完成，且保证了界面操作的友好。

3.2 声音控制模块

声音控制模块主要包含了声音的回放、录制、编码和解码四个子模块。其中录音与回放需要根据 Windows 对声音控制的支持讨论；编码和解码模块可采用对当前通用的 G.711、G.723[1]、G.729[2]、speex、iLBC 等方式进行简单封装完成。

3.2.1 录音回放调用函数

Windows 定义了 WAVEFORMATEX 数据结构，用于提供给用户指定打开录音回放时使用的数据类型等相关参数。要录音则调用 waveInOpen 函数打开录音通道、要回放则调用 waveOutOpen 函数打开回放通道。这些音频通道都均具备多种打开方式。不同的方式决定了对音频数据处理后的相应方法。本系统中使用了 CALLBACK_FUNCTION 方式，其中设置的回调函数，处理声音播放或录制完一个片断后的操作。

Windows 的声音播放模型采用队列形式让用户维护声音缓冲。在必要的初始化过程后，用户播放声音需要做几个步骤：将数据帧包装在 WAVEHDR 的结构内；再通过 waveOutPrepareHeader 函数准备该结构的数据；之后通过 waveOutWrite 函数将包含数据帧的结构推入系统缓冲，在调用 waveOutWrite 函数后对应数据帧交由另一线程处理，函数立即返回；通过判断结构体内 dwFlags 的数据位 WHDR_DONE，确认该数据帧播放结束后，需调用 waveOutUnprepareHeader。录制一个片断方法与回放雷同，通过调用 waveInPrepareHeader、waveInAddBuffer 及 waveInUnprepareHeader 三个函数实现。同时在录音与回放过程中可能遇到其他情况，可以调用更多的音频处理函数进行处理。例如 waveInReset、waveInStop、waveOutReset、waveOutStop 等。

在结束录制和回放后，需要用 waveInClose 和 waveOutClose 将打开的相应音频通道关闭。

3.2.2 缓冲讨论

如果不发生丢包，在网络抖动范围较小（不发生错序）的情况下，将即时收到的数据帧依次不断推入系统缓冲，就能达到较好的 QoS。但当网络抖动较大，出现后发送的数据包比先发送的数据包提前到达，仅用系统缓冲机制无法有效地识别丢包还是抖动。抖动过大而造成的错序误判为丢包，会影响系统 QoS。解决这个问题的办法是设置消抖缓冲：如果刚刚到达的数据帧之前有数据帧没有到达，则把该数据帧推入消抖缓冲等它以前的数据帧到达后再播放；如果刚刚到达的数据帧是系统缓冲播放完后应当播放的数据帧，则将其直接推入系统缓冲。

当不出现丢包且缓冲可以无限长时，该机制也能保证很好的话音质量，但它可能由于抖动过大而造成很大的延时。而事实上不可能开设无限长的缓冲，因此在数据填满消抖缓冲时，须将消抖缓冲中较早应到达但还未到达的数据当作丢包用 FEC 机制处理[3][4]，出现这种情况语音质量自然会有一定下降。为了保证声音质量尽可能好又保证延时尽可能小，就要选择合适的延时作为消抖缓冲的长度。可以采用平滑策略[5]及自适应算法动态的选择消抖缓冲的长度[6]。

3.2.3 编码解码部分

编码解码模块分别使用如下两个接口：

class TEncoder

基金项目：教育部"985 工程"二期"智能化国防安全信息技术"科技创新平台项目和"IP 电话 NetTone 的 QoS 设计与实现"项目的部分资助。

```
{
public:
 virtual RET_TVOIP Init()=0;
 virtual RET_TVOIP Encode(const char *, int , char *, uint8 &)=0;
 virtual RET_TVOIP Cleanup()=0;
};
class TDecoder
{
public:
virtual RET_TVOIP Init()=0;
 virtual RET_TVOIP Decode(const char *, uint8 , char *, int )=0;
 virtual void Cleanup()=0;
};
```

任何编码解码方式只要封装了这种接口，均可直接被系统的相关部分调用，很容易应用到本系统中。本系统经过比对各个编码方式的性能，最终选择了 speex 编码方式完成设计。

3.3 网络模块

网络模块负责封装数据包，直接与音频模块相连，处理发送和接收音频压缩数据事务。本模块基于 UDP 协议，自建 VoIP 通讯协议，通过自动机的方法建立 VoIP 状态机，实现 VoIP 拨号、断开等业务的可靠处理。同时使用多线程技术，及网络事件模型，完成了网络模块的构架。

3.3.1 VoIP 状态机

VoIP 通话的过程与 TCP 协议类似，但需要处理更多的状态以适应拨号、断开、超时等各种需求。为了保证拨号时候拨号的数据包能够有效到达对方，设计了类似于 TCP 协议中的三路握手四路断开方式。在一次通话中，根据发起呼叫人可分为主叫方与被叫方，根据提出断开人可分为主断方与被断方。如图 3，通话建立和断开采用类似于 TCP 协议中三路握手四路断开的过程，其中的 DialReply_AccpetA 包在 TCP 的握手协议中并没有对应的包，这是为被叫方接受呼叫设计的。

图 3 通常呼叫建立到断开连接状态转换过程

分别为拨号建立、断开连接、传输数据等操作及相应的出错情况制作全部完整的状态转换后，就很容易为这些转换建立对应的状态机。将这些状态机整合起来就得到一个 VoIP 状态机。之后处理 IP 电话状态转换的过程即转化为简单的状态机转换。

3.3.2 事件模型

在 Windows 程序设计中存在许多网络模型，有阻塞模型和非阻塞模型。阻塞模型由于相应速度慢，在对网络流量需求不太高的应用上采用。由于 Windows 引入了窗口消息机制，因此非阻塞模型除了事件触发模型外还有消息回调模型。虽然消息回调模型在 MFC 中有很好的封装，且在开发中易于使用，但由于它需要一个窗口作为消息到达的承载体浪费资源，消息机制性能不高，因此针对 VoIP 应用，事件触发模型是性能最佳的选择。同时使用网络事件模型还能方便的与线程间数据交互使用的事件对象很好的磨合。

3.3.3 线程间数据交互

本文设计的网络模块存在两条线程：将网络传输模块中的接收数据- 解码- 播放音频制成一条接受线程，而录音- 压码- 发送制一条发送线程。无论发送线程还是接收线程，都存在于别的线程交互数据的问题。在 Windows 中同一进程中的不同线程之间的内存区域是共享的，为了让两条线程之间交换数据尽可能的快，因此在两条线程之间设计公共循环队列，使用事件对象为两条线程读写队列做互斥，两线程均能直接操作队列中的实时数据，这种方法有效避免了拷贝数据需要的额外时间。处理事件对象主要使用 CreateEvent、WaitForSingleObject、WaitForMultipleObjects、SetEvent、ResetEvent、CloseHandle 等 Windows API。这些函数也能直接处理 Windows 网络事件模型中的事件，网络处理使用事件模型比使用其他模型具有更好性能。

3.4 用户界面模块

用户操作软件，在屏幕上直接接触的就是用户界面模块，所有与 VoIP 核心控制系统相关的用户界面指令，都需要送达核心控制系统，以便进行进一步调度。由于设计内核的过程需要更多的调试信息，当本系统被商业软件采用的时候，需将用户界面替换使界面更加友好。TVoIPSession 为用户界面模块提供了虚函数接口，它们会在相应事件发生的时候被 VoIP 核心控制模块中内含的线程调用。以下是全部虚函数接口：

```
virtual void OnTipMsg(TVoIPSession *sess);
virtual void OnDebugMsg(TVoIPSession *sess);
virtual void OnChangeLocalVolume(uint16 nVolume);
virtual void OnChangeRemoteVolume(uint16 nVolume);
virtual void OnDialing(const TCHAR *strIP,uint16 nPort);
virtual void OnDialReply(const TCHAR *strIP,uint16 nPort);
virtual void OnDialNoReply(const TCHAR *strIP,uint16 nPort);
virtual void OnNoAcceptA(const TCHAR *strIP,uint16 nPort);
virtual void OnNoAcceptB(const TCHAR *strIP,uint16 nPort);
virtual void OnAcceptA(const TCHAR *strIP,uint16 nPort);
virtual void OnAcceptB(const TCHAR *strIP,uint16 nPort);
virtual void OnRefuseA(const TCHAR *strIP,uint16 nPort);
virtual void OnBusyA(const TCHAR *strIP,uint16 nPort);
virtual void OnHangup(const TCHAR *strIP,uint16 nPort);
```

OnTipMsg 与 OnDebugMsg 函数提供给界面设计者实时读取显示 TVoIPSession 中参数值（如上行速率、音量参数）的方法，它们每隔一小段时间被核心线程调用一次，间隔长短通过 TIPMSG_INTERVAL 及 DBGMSG_INTERVAL 常量设置。同时利用 #ifdef _DEBUG … #endif 预编译字段将 OnDebugMsg 等调试代码隔离，在 Release 版本中该函数自动失效。

OnChangeLocalVolume 与 OnChangeRemoteVolume 为当本地麦克风音量被调整和当远程回放音量被调整时反馈给界面的虚函数。

OnDialing、OnDialReply、OnAcceptA、OnAcceptB、OnRefuseA、OnBusyA、OnHangup 为响应数据包到达时调用的虚函数。OnDialNoReply、OnNoAcceptA、OnNoAcceptB 为对方不在线、主叫方无应答、被叫方无应答时调用的虚函数。这些情况都需要用户界面模块给用户提示信息。

图 4 显示了本文设计系统的调试界面模块运行时图片。左边是简单的拨号面板以及各种调试用的开关设置，右边是 Debug 信息窗口。该窗口的显示信息都在 OnDebugMsg 中设置。

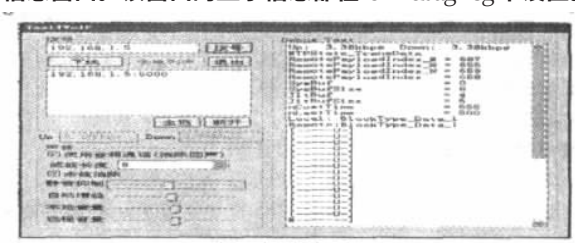
图 4 调试界面图

4. 结束语

本文分析了构建 VoIP 客户端平台的实现方案中几个关键技术的具体设计。VoIP 促进了网络资源的利用，降低语音业务成本，在全球范围内得到迅速发展，是当今世界发展最快、普及最快的一门应用服务技术之一。尽管 VoIP 技术在许多方面还需要不断改进与提高，单可以相信，随着 VoIP 技术的不断发展和完善，必将在新一代电信网络中得到广泛应用。

参考文献：

1. ITU- T, Recommendation G.723.1 (Dual Rate Speech Coder for Multimedia Communications Transmitting at 5.3 & 6.3 Kbit/s), ITU- T, July. 1996[S].

2. ITU- T, Recommendation G.729.A (Coding of speech at 8 kbit/s using conjugates- turcture algebraic- code- excited linear- prediction (CS- ACELP), Mar. 1996[S].

3. 彭波，韦岗，一种基于卷积码的网络丢包恢复新方法[J]，通信学报，第 23 卷，第 3 期，21- 26，2002。

4. 胡飞，朱耀庭，朱光喜，基于 Galois 域 Reed- Solomon 码的数据包层 FEC 编码软件实现[J]，通信学报，第 23 卷，第 3 期，57- 64，2002。

5. Shyh- Fang Huang, Eric Hsiao- Kuang Wu, and Pao- Chi Chang, Adaptive VoIP Smoothing of Pareto Traffic Based on Optimal E- Model Quality, LNCS 3768, P747- 758, 2005.

6. 苟先太；金炜东；靳蕃，一种自适应 IP 语音缓冲算法的研究与应用[J]. 计算机研究与发展，2005 年 12 期.