

基于 JMF 架构的流媒体 RTP / RTCP 传输模型设计

胡开明, 陈建华, 王玉贤

(广东松山职业技术学院计算机系, 广东 韶关 512126)

摘要: JMF 提供了对 RTP/RTSP (实时传送`议和实时流转`议)的支持, 通过在分析 RTP/RTCP`议时得出的反馈机制的方向, 实现一种基于 JMF 的 RTP/RTCP 传输模型的整体设计。实现了一个动态的网络反馈机制, 并利用其提供的动态的反馈信息实现了对发送端和接收端 Buffer 的控制, 保证流媒体传输的 QoS 以提高流媒体传输的效率。

关键词: JMF 架构; 流媒体; RTP/RTCP; 传输模型

中图分类号: TP37

文献标识码: A

在网络带宽得到日益改善的今天, 网络服务的内容成为今后网络应用的一个重点发展方向, 流媒体技术及其相关产品将更广泛应用于远程教育、网络电台、视频点播、收费播放等。对于一个基本的流媒体系统而言, 它的基本目的是把流媒体传输到客户端, 是一个典型的 S/C (Server/Client) 架构。在过去几乎所有的 S/C 架构中, 出现了一个不可避免的问题, 就是在客户端数量过多的时候, 出现了服务器端由于负载过重而导致系统性能急剧下降甚至崩溃。为了解决这个问题, 在一个 S/C 架构中的各个层次采用了多种方法来解决这个问题, 如服务器集群技术及相关技术如负载均衡代理等^[1]。利用 JAVA 的 JMF (Java Media Framework)。利用其提供的多媒体功能加上对其原有接口的扩展, 实现了一种基于 JMF 的 RTP/RTCP 传输模型的整体设计。在对 RTP/RTCP 工作模型的详细分析中, 实现了一个动态的网络反馈机制, 并利用其提供的动态的反馈信息实现了对发送端和接收端 Buffer 的控制, 保证流媒体传输的 QoS 以提高流媒体传输的效率。

1 JMF 架构

1.1 JMF 简介

JMF (Java Media Framework) 是 Sun 公司提出的 Java 媒体架构。它是对应 Java 2 平台标准版 (J2SE) 的一种可选用的应用编程接口 (API) 软件。JMF 的源代码将

通过 SCSL (Sun 社团源代码许可模式) 发布。这一强大的媒体工具包可以在任何版本 (1.1.x 及以上版本) 的 Java 平台上的运行^[2]。

JMF 2.1.1 技术提供了先进的媒体处理能力, 从而扩展了 Java 平台的功能^[3]。JMF 所提供的多媒体功能如下:

(1) 可以在 Java Applet 和应用程序中播放各种媒体文件。它提供了对各种主要媒体形式和编码的支持, 如 M-JPEG, H.263, MP3, Macromedia Flash, IBM 的 HomeMedia 和 Beatniks 的 Rich Media Format (RMF) 等。JMF 2.1.1 还支持多种媒体类型, 如 Quicktime MOV, Microsoft AVI 和 MPEG-1^[4]。

(2) 可以播放从互联网上下载的媒体流。

(3) 可以利用麦克风和摄像机一类的设备截取音频和视频, 并保存成多媒体文件。

(4) 处理多媒体文件, 转换文件格式。

(5) 向互联网上传音频和视频数据流。

(6) 在互联网上广播音频和视频数据。

JMF 架构中还包括了一个开放和统一的媒体架构, 可使开发人员灵活采用各种媒体回放、捕获组件, 或采用他们自己的定制的内插组件。

1.2 JMF 中媒体的播放

在 JMF 中对应播放器的接口是 Player, Player 对象将音频、视频数据流作为输入, 然后将数据流输出到音

箱或屏幕上, 就像 CD 播放机读取 CD 唱片中的歌曲, 然后将信号送到音箱上一样。Player 对象有多种状态, MF 中定义了 MF 的六种状态^[5], 在正常情况下 Player 对象需要经历每个状态, 才能播放多媒体。下面是对这些状态的说明。状态转换如图 1 所示。

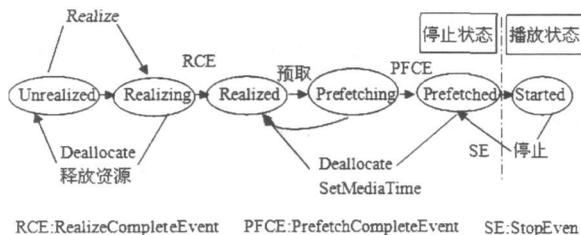


图 1 JMF Player 的状态转换图

(1) Unrealized 在这种状态下, Player 对象已经被实例化, 但是并不知道它需要播放的多媒体的任何信息。

(2) Realizing 当调用 realize() 方法时, Player 对象的状态从 Unrealized 转变为 Realizing, 在这种状态下, Player 对象正在确定它需要占用哪些资源。

(3) Realized 在这种状态下 Player 对象已经确定了它需要哪些资源, 并且也知道需要播放的多媒体的类型。

(4) Prefetching 当调用 prefetch() 方法时, Player 对象的状态从 Realized 变为 Prefetching, 在该状态下的 Player 对象正在为播放多媒体做一些准备工作, 其中包括加载多媒体数据, 获得需要独占的资源等。这个过程被称为预取 (Prefetch)^[6]。

(5) prefetched 当 Player 对象完成了预取操作后就到达了该状态。

(6) Started 当调用 start() 方法后, Player 对象就进入了该状态并播放多媒体。

2 RTP/RTCP 模型及动态网络反馈机制

2.1 RTP/RTCP 基本模型

RTP 协议是一种应用型的传输层协议, 它并不提供任何传输可靠性的保证和流量的拥塞控制机制。RTP 协议位于 UDP 协议之上, 在功能上独立于下面的传输层 (UDP) 和网络层, 但不能单独作为一个层次存在, 通常是利用低层的 UDP 协议对实时视音频数据进行组播 (Multicast) 或单播 (Unicast), 从而实现多点或单点视音频数据的传输。

RTP 协议被设计成能够为某种特定的应用提供服务的一种协议。实际上, RTP 协议的实现已经被融合到应用程序中来。RTP 没有连接的概念, 它可以建立在

面向连接的底层协议上, 也可以建立在面向无连接的底层协议上, 因此 RTP 协议对传输层是独立的。RTP 协议一般由两个部分组成: 数据报文部分 (RTP 报文) 和控制报文部分 (RTCP)。与传统注重的高可靠的数据传输的传输层协议相比, RTP 更加侧重数据传输的实时性。此协议提供的服务包括时间戳、数据序列、时戳、传输控制等。RTP 与辅助控制协议 RTCP 一起得到数据传输的一些相关的控制信息。其基本模型如图 2 所示:

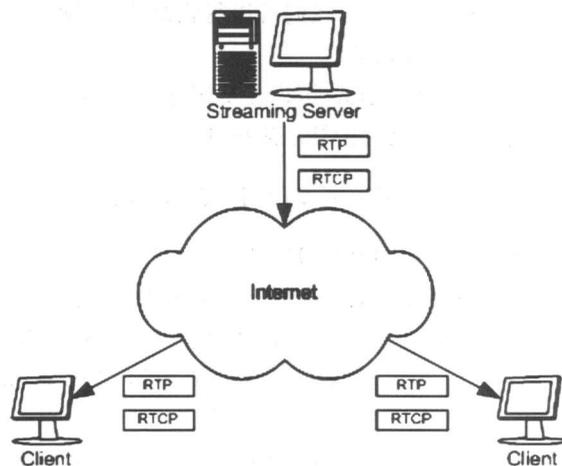


图 2 RTP/RTCP 基本工作模型

2.2 基于 RTCP 的动态网络反馈机制

RealSystem 的 SureStream 技术是实现流式媒体传输关键技术之一, 可以根据不同连接速率创建多个文件和采用一种复杂客户/服务器机制探测带宽变化。但是, 对于网络环境而言, 网络的状况是不断变化的, SureStream 技术显然没有能够对网络状况的变化进行任何的响应, 即使网络状况发生变化, 它对客户端发送的文件仍是在针对客户端请求时连接速率创建的。对于一个变化不大的网络而言, 这种方法当然行之有效, 但是对于一个不断变化的网络而言, 建立动态的带宽侦测和反馈机制将是解决问题的有效途径。

就可以利用 RTCP 的报告来侦测网络变化的趋势, 根据网络变化的趋势, 可以对系统其他部分进行调整。

首先, 通过 RR 控制包获取反馈信息。即读取 RTCP 的 RR (接收者报告) 包并做统计分析, 必须利用 RTCP 提供的 3 个性能指标来确定网络当前状况:

(1) 传输时延抖动估计 Interarrival jitter 记作 J, 传输时延抖动估计是指两个相邻数据包到达事件的平均偏差 (Mean Deviation) 估算, 若到达时间比较规律, 则该值为零, 否则该数值比较大。网络状况的变化并非毫无规律, 在大体上, 网络状况的变化只是一个状态到另外一个状态的转换过程。而 Interarrival jitter 必然体现

这个状态的转换过程,即由 0 到非 0 再到 0 的所代表的稳定到变化再到稳定的过程。

(2) 连续数据包传送的时延差值。记作 D_i 。RTCP 定义的公式为:

$$D(i-i-1) = (R_{i-1} - R_i) - (S_{i-1} - S_i) = (R_{i-1} - S_{i-1}) - (R_i - S_i)$$

其中 R_i , S_i 分别代表第 i 个数据包接收和发送的 RTP 时间戳,因此 $R_i - S_i$ 就代表第 i 个数据包的相对发送时间, $D(i-i-1)$ 则代表相邻两个数据包的时延差。当 $D(i-i-1)$ 不断增大时,我们可以得到结论,即网络可用带宽正在变小,反之变大。

根据这个变化趋势,可以在 MF 平台上实现一个基于动态侦测带宽变化的智能流机制。

3 构建基于 MF 架构的流媒体传输

3.1 SessionManager 接口

SessionManager 接口定义了应用程序之间建立、参与一个会话和释放资源并退出整个会话的一整套方法。同样可以利用其来建立在 RTP 传输中的 RTP Session, 通过建立两个端口的 Session 对话,为 RTP, RTCP 建立了两个 Port 的连接。分别用来进行 RTP 的数据传送和 RTCP 的反馈和控制信息的传送。下面将利用 MF 中的 SessionManager 来实现 RTP Session

(1) Session Statistics

通过 SessionManager 中的 Session Statistics, 可以保存每个会话的信息,即每个 RTP/RTCP 包的发送和接收信息。SessionManager 保存了两个可以用来记录发送和接收信息包的属性:

GlobalReceptionStats 保存了会话中全局的接收信息。

GlobalTransmissionStats 保存了所有发送者的累计发送信息。

它们将被用来和 RTCP 包中的 Sender's packet count 等字段对应。

(2) Session Participants(会话参与者)

MF 的 SessionManager 能够记录一个会话中所有参与者的信息。每个参与者是通过一个类的实例 (Instance) 来确认的。该实例使用了会话中 Participant 接口。当 SessionManager 收到一个以前从未收到包含了 SDES (Source Description) 和 CNAME 的 RTP 数据包时,将创建一个 Participant 接口。参与者这时就可以加入一个会话。

在 SessionManager 中还有一个 LocalParticipant 属性,可以用来表示本地参与会话的 Client 或 Server。本地参与者可以发送 RTCP 控制信息。每个会话参与者

可以参与多个数据流,每个数据流将通过 RTP 中的 SSRC 来区分数据流的源。

(3) Session Stream

通过 SessionManager 还为每个 RTP 数据流保持一个 RTPStream 对象。ManagerSession 将 RTP 数据流分两类: ReceiveStream 表示从远端参与者接收到的数据流。SendStream 将表示通过 Processor 或者输入 DataSource 向网络发送的数据流。当 SessionManager 侦测到一个新的 RTP 数据包的时候,将自动建立一个 ReceiveStream, 而如果要向外发送 RTP 数据流,将调用 SessionManager 中的 createSendStream 方法。

3.2 RTP 事件

通过继承 MF 中 MediaEvent 的类,可以创建响应的 RTP 事件。为了得到 RTP 事件,必须相应的 RTP Listener 和 SessionManager 结合。

(1) SessionListener 通过它得到一个会话状态的改变。包括:

NewParticipantEvent 表示一个新的参与者加入会话。

LocalCollisionEvent 表示参与者请求的同步资源正在使用。

(2) SendStreamListener 通过它得到一个正在传送的 RTP 数据流的状态的改变。

NewSendStreamEvent 表示本地参与者已经创建一个新的发送数据流。

ActiveSendStreamEvent 表示从 DataSource 创建的数据流已经开始发送。

InactiveSendStreamEvent 表示从本地 DataSource 创建的数据流已经停止。

LocalPayloadChangeEvent 表示数据流格式已经开始改变。

StreamClosedEvent: 表示数据流已经停止。

(3) ReceiveStreamListener 通过它得到一个正在接收的 RTP 数据流的状态的改变。

NewReceiveStreamEvent 表示 SessionManager 已经创建了一个从新侦测到的地址来的接收数据流。

InactiveReceiveStreamEvent 表示数据的传送已经停止。

TimeoutEvent 表示数据传送超时。

RemotePayloadChangeEvent 表示接收到的数据流格式已经改变。

ApplicationEvent 表示收到了一个 RTCP App 数据包。

(4) RemoteListener 通过它得到远端会话参与者

的时间或 RTP 控制信息。

ReceiveReportEvent 表示接受到一个 RTCP 的 RR 包。

SenderReportEvent 表示收到 RTCP 的 SR 包。

RemoteCollisionEvent 表示两个远端的参与者使用了相同的 SSRC, 出错。

3.3 RTP会话控制及基于反馈的智能流机制的实现

MF 架构中的 SessionManager 同样提供了对会话的控制功能。SessionManager 中的控制功能是继承了 MF 中的 Control。这样, 通过 GetControls 方法, 就可以得到例如 BufferControl 的接口。

无论对于 Stream Server 还是 Client 而言, 对于一个发起的流媒体传输会话, 都会在系统资源中占用一定的 Buffer 资源。一般来说, 一个 Session 的 Buffer 是固定不便的。MF 的会话控制提供了对 Buffer 的控制能力。同时, MF 对 RTP/RTCP 协议的支持使得能够获得 RTCP 包中包含的信息。在上面对 RTCP 协议进行分析的时候, 已经得到了一个基于 RTCP 报告、可获取对网络状况变化趋势的反馈机制, 使得能够根据网络状况, 改变相应 Session 的 Buffer 大小, 使其适应当前网络的情况。实现智能流。如图 3 所示。

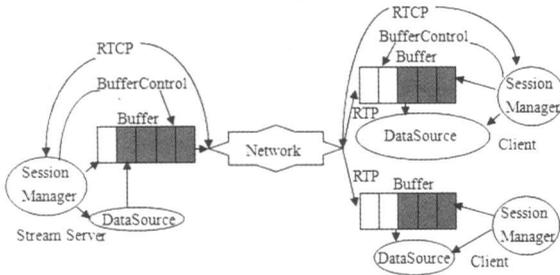


图 3 MF 架构上的基于 RTCP 报告的智能流模型

假设 Buffer 的大小和相邻包的时延只是一个简单的反向关系, 即时延越大, 按增加比例缩小 Buffer 反之则按减少比例减少 Buffer。如图 4 所示。

(1) 对 J 的获取方法如下:

```
import javax.media.rtp.rtcp;
Long J= GetJitter();
```

(2) 对 Buffer 的控制接口的加入如下:

```
public class CustomBufferControl extends Frame implements ControllerListener {
    /* 给定一个 DataSource 并利用一个 Processor 对其进行控制 */
    public boolean open(DataSource ds, int captureBufferSize, int BufSize) {
        System.err.println(" create processor for "+ ds.get
```

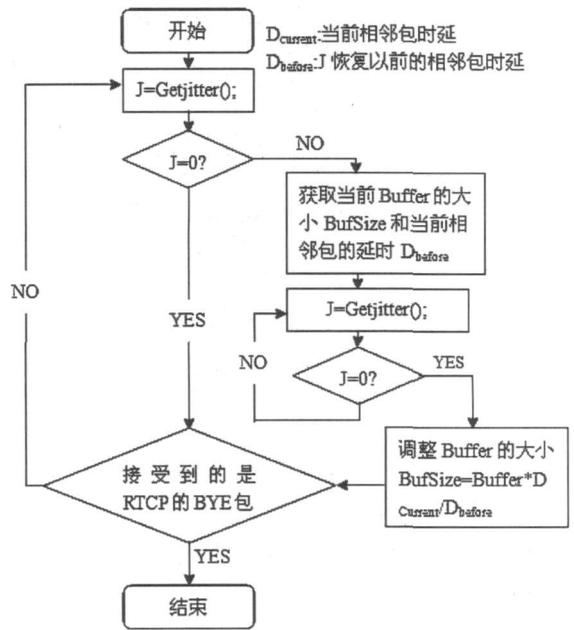


图 4 获取 RTCP 包信息控制 Buffer 大小的算法流程图

ContentType(); //检查是否能够对 DataSource 进行 BufferControl 的控制

```
Control c= (Control) ds.getControl(" javax.media.control.BufferControl");
if (c == null) ((BufferControl) c). setBufferLength(captureBufferSize);
try {p=Manager.createProcessor(ds);
} catch(Exception e) {
System.err.println("Failed to create a processor from the given DataSource "+ e);
return false;
}
p.addControllerListener(this); //加入了控制接口
Control a= p.getControls();
((BufferControl) a). setBufferLength();
```

4 MF 架构上 RTP/RTCP 模型的实现

通过对 MF 架构的系统分析和对 SessionManager 等接口的扩展, 可以用 MF 架构实现基于 RTP 数据流的传输。可以利用 SessionManager 对一个会话的创建和控制进行网络流媒体的传输。数据将从 DataSource 获取。整体设计如图 5 所示。例如, 为了传送实时捕捉的数据。可以按如下步骤进行:

- (1) 为该会话创建并初始化一个 SessionManager
- (2) 构建一个处理器 (Processor), 它将使用相应的 DataSource, 这个 DataSource 将是由捕捉设备输入的。
- (3) 将输出格式定义为 RTP 格式, 并加入相应的 RTP 解码。

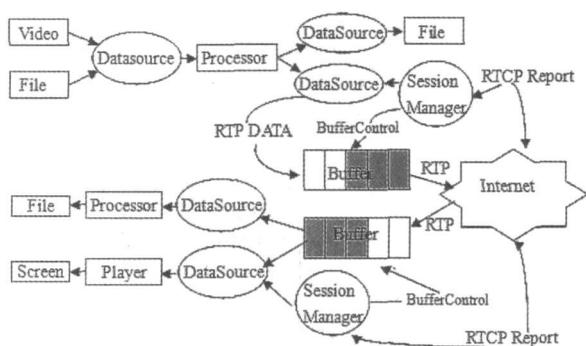


图 5 基于 JMF 的 RTP/RTCP 传输模型的整体设计

(4) 从 Processor 得到输出 DataSource。

(5) 调用 SessionManager 的 CreateSendStream 方法发送输出 DataSource 中的数据。

(6) 不断获取 RTCP 报告信息, 对 Buffer 大小控制, 以保证传输效率。

可以通过 SendStream 中的 Start 和 Stop 方法来对数据的发送进行控制。当第一次开始发送数据时, SessionManager 将充当一个接受者的角色(发送 RTCP 的 RR 数据包)。一旦创建了 SendStream, 将发送 RTCP 的 SR 数据包, 并在数据流传输期间充当一个发送者的角色。当所有数据包发送完毕, 可以关闭该 SessionManager。

在接收方要做的工作是相同的, 用 SessionManager 创建一个 RTP 会话以接收数据包并交换控制信息。在创建接收 DataSource 的同时创建一个 Player 对接受中的 DataSource。

Design of the Streaming Media RTP/RTCP Transmission Model Based on JMF

HU Kaiming, CHEN Jian-hua, WANG Yuxian

(Department of Computer Science, Guangdong Songshan Vocational College, Shaoguan 512126, China)

Abstract JMF supports RTP/RTSP (real-time transmission protocol and real-time transfer protocol). After analyzing RTP/RTSP to get the direction of the feedback mechanism, integrally RTP/RTSP transmission model on the basis of JMF is realized. This kind of model has realized a dynamic network of feedback mechanism, the buffer control of sending terminal and receiving terminal by utilizing dynamic feedback information which provides. It can guarantee the QoS of fluid transmission to increase the efficiency of the fluid transmission.

Key words JMF framework; streaming media; RTP/RTCP; transmission mode

5 结束语

JMF 技术提供了先进的媒体处理能力, 从而扩展了 Java 平台的功能。JMF 提供了对 RTP/RTSP (实时传送协议和实时流传输协议) 的支持, 通过在分析 RTP/RTCP 协议时得出的反馈机制的方向, 实现一种基于 JMF 的 RTP/RTCP 传输模型的整体设计。在对 RTP/RTCP 工作模型的详细分析中, 实现了一个动态的网络反馈机制, 并利用其提供的动态的反馈信息实现了对发送端和接收端 Buffer 的控制, 保证流媒体传输的 QoS 以提高流媒体传输的效率。

参考文献:

- [1] 胡泽, 赵新梅. 流媒体技术与应用 [M]. 北京: 中国广播电视出版社, 2006
- [2] 庄捷. 流媒体原理与应用 [M]. 北京: 中国广播电视出版社, 2007
- [3] 肖磊, 陈卓. 流媒体技术与应用完全手册 [M]. 重庆: 重庆大学出版社, 2003
- [4] 梁振军. 计算机网络通信与协议 [M]. 北京: 石油工业出版社, 1990
- [5] 彭波, 孙一林. Java 多媒体技术 [M]. 北京: 清华大学出版社, 2004
- [6] 王永乐, 徐书欣. 基于 P2P 覆盖树网络的流媒体传输技术 [J]. 四川理工学院学报: 自然科学版, 2008, 21 (3): 60-62