WebRTC 中 RTP/RTCP 协议实现分析

RTP/RTCP 协议是流媒体通信的基石。RTP 协议定义流媒体数据在互联网上传输的数据包格式，而 RTCP 协议则负责可靠传输、流量控制和拥塞控制等服务质量保证。在 WebRTC 项目中，RTP/RTCP 模块作为传输模块的一部分，负责对发送端采集到的媒体数据进行进行封包，然后交给上层网络模块发送；在接收端 RTP/RTCP 模块收到上层模块的数据包后，进行解包操作，最后把负载发送到解码模块。因此，RTP/RTCP 模块在 WebRTC 通信中发挥非常重要的作用。

本文在深入研究 WebRTC 源代码的基础上，以 Video 数据的发送和接收为例，力求用简洁语言描述 RTP/RTCP 模块的实现细节，为进一步深入掌握 WebRTC 的实现精华打下良好基础。

一 前言: RTP/RTCP协议概述

RTP协议是Internet上针对流媒体传输的基础协议，该协议详细说明在互联网上传输音视频的标准数据包格式。RTP协议本身只保证实时数据的传输，RTCP协议则负责流媒体的传输质量保证，提供流量控制和拥塞控制等服务。在RTP会话期间，各参与者周期性彼此发送RTCP报文。报文中包含各参与者数据发送和接收等统计信息，参与者可以据此动态控制流媒体传输质量。RTP和RTCP配合使用，通过有效反馈使使流媒体传输效率最佳化。

RFC3550 [1]定义RTP/RTCP协议的基本内容，包括报文格式、传输规则等。除此之外，IETF还定义一系列扩展协议，包括RTP协议基于档次的扩展，和RTCP协议基于报文类型的扩展，等等。详细内容可参考文献[2]。

二 WebRTC线程关系和数据流

WebRTC对外提供两个线程：Signal和Worker，前者负责信令数据的处理和传输，后者负责媒体数据的处理和传输。在WebRTC内部，有一系列线程各司其职，相互协作完成数据流管线。下面以Video数据的处理流程为例，说明WebRTC内部的线程合作关系。

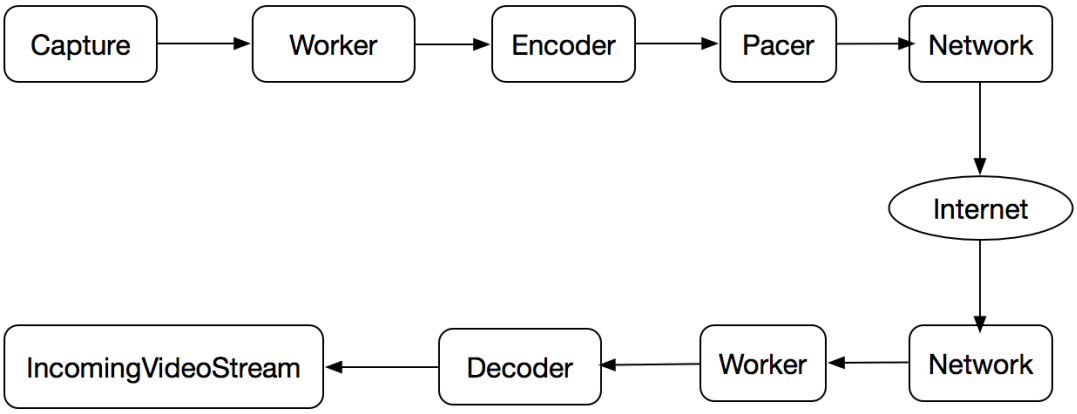

图1 WebRTC线程关系和数据管线

如图1所示，Capture线程从摄像头采集原始数据，得到VideoFrame；Capture线程是系统相关的，在Linux系统上可能是调用V4L2接口的线程，而在Mac系统上可能是调用AVFoundation框架的接口。接下来原始数据VideoFrame从Capture线程到达Worker线程，Worker线程起搬运工的作用，没有对数据做特别处理，而是转发到Encoder线程。Encoder线程调用具体的编码器(如VP8, H264)对原始数据VideoFrame进行编码，编码后的输出进一步进行RTP封包形成RTP数据包。然后RTP数据包发送到Pacer线程进行平滑发送，Pacer线程会把RTP数据包推送到Network线程。最终Network线程调用传输层系统函数把数据发送到网络。

在接收端，Network线程从网络接收字节流，接着Worker线程反序列化为RTP数据包，并在VCM模块进行组帧操作。Decoder线程对组帧完成的数据帧进行解码操作，解码后的原始数据VideoFrame会推送到IncomingVideoStream线程，该线程把VideoStream投放到render进行渲染显示。至此，一帧视频数据完成从采集到显示的完整过程。

在上述过程中，RTP数据包产生在发送端编码完成后，其编码输出被封装为RTP报文，然后经序列化发送到网络。在接收端由网络线程收到网络数据包后，经过反序列化还原成RTP报文，然后经过解包得到媒体数据负载，供解码器进行解码。RTP报文在发送和接收过程中，会执行一系列统计操作，统计结果作为数据源供构造RTCP报文之用。RTP报文构造、发送/接收统计和RTCP报文构造、解析反馈，是接下来分析的重点。

三 RTP报文发送和接收

RTP报文的构造和发送发生在编码器编码之后、网络层发送数据包之前，而接收和解包发生在网络层接收数据之后、解码器编码之前。本节详细分析这两部分的内容。

3.1 RTP报文构造和发送

图2描述发送端编码之后RTP报文的构造和发送过程，涉及三个线程：Encoder、Pacer和Network，分别负责编码和构造RTP报文，平滑发送和传输层发送。下面详细描述这三个线程的协同工作过程。

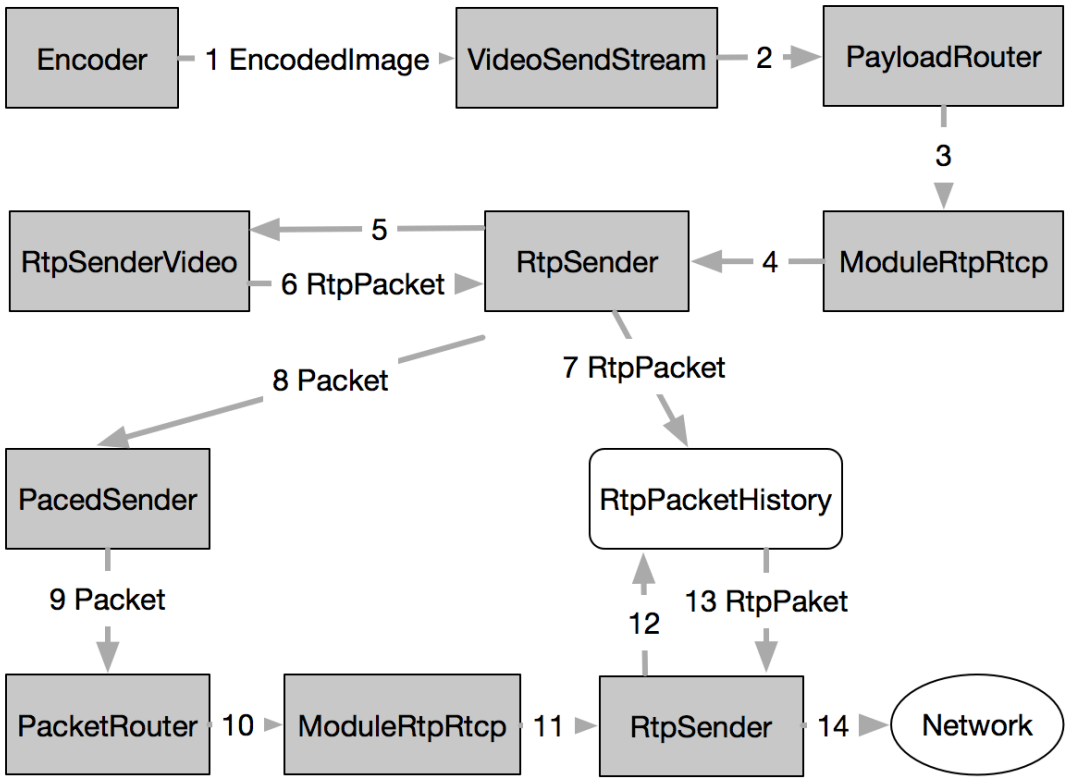

图2 RTP报文构造和发送

Encode 线程调用编码器(比如 VP8)对采集到的 Raw VideoFrame 进行编码，编码完成以后，其输出 EncodedImage 通过回调到达 VideoSendStream::Encoded()函数，进而通过 PayloadRouter 路由到达 ModuleRtpRtcpImpl::SendOutgoingData()。该函数向下到达 RtpSender::SendOutgoingData()，根据媒体数据类型选择调用 RtpSenderVideo::SendVideo()，该函数对 EncodedImage 进行打包，然后填充 RTP 头部构造 RTP 报文；如果配置了 FEC，则进一步封装为 FEC 报文。最后返回 RtpSender::SendToNetwork()进行下一步发送。

RtpSender::SendToNetwork()函数把报文存储到 RTPPacketHistory 结构中进行缓存。接下来如果开启 PacedSending，则构造 Packe 发送到 PacedSender 进行排队，否则直接发送到网络层。

Pacer 线程周期性从队列中获取 Packet，然后调用 PacedSender::SendPacket()进行发送，接下来经过 ModuleRtpRtcpImpl 到达 RtpSender::TimeToSendPacket()。该函数首先从 RtpPacketHistory 缓存中拿到 Packet 的 RTP 负载数据，然后调用 PrepareAndSendPacket()函数：更新 RtpHeader 的相关域，统计延迟和数据包，调用 SendPacketToNetwork()把报文发送到传输模块。

Network 线程则调用传输层套接字执行数据发送操作。

至此，发送端的 RTP 构造和发送流程完成。需要注意的是，在 RtpSender 中进行 Rtp 发送后，会统计 RTP 报文相关信息。这些信息作为 RTCP 构造 SR/RR 报文的数据来源，因此非常重要。

3.2 RTP报文接收和解析

在接收端，RTP 报文的接收和解包操作主要在 Worker 线程中执行，RTP 报文从 Network 线程拿到后，进入 Worker 线程，经过解包操作，进入 VCM 模块，由 Decode 线程进行解码，最终由 Render 线程进行渲染。下图 3 描述 RTP 报文在 Worker 线程中的处理流程。

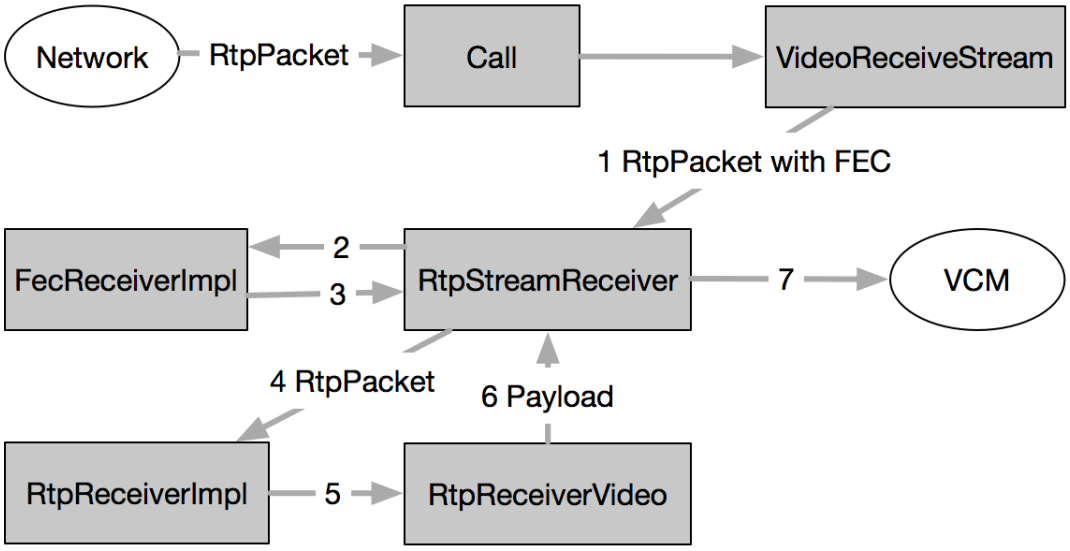

图3 RTP报文接收和解析

RTP 数据包经网络层通过 WebRTCVideoChannel2 到达 Call 对象，根据其 SSRC 找到对应的 VideoReceiveStream，通过调用其 DeliverRtp()函数到 RtpStreamReceiver::DeliverRtp()。该函数首先解析数据包得到 RTP 头部信息，接下来执行三个操作：1.码率估计；2.继续发送数据包；3.接收统计。码率估计模块使用 GCC 算法估计码率，构造 REMB 报文，交给 RtpRtcp 模块发送回发送端。而接收统计则统计 RTP 接收信息，这些信息作为 RTCP RR 报文的数据来源。下面重点分析接下来的数据包发送流程。

RtpStreamReceiver::ReceivePacket()首先判断数据包是否是 FEC 报文，如果是则调用 FecReceiver 进行解包，否则直接调用 RtpReceiver::IncomingRtpPacket()。该函数分析 RTP 报文得到通用的 RTP 头部描述结构，然后调用 RtpReceiverVideo::ParseRtpPacket()进一步得到 Video 相关信息和负载，接着经过回调返回 RtpStreamReceiver 对象。该对象把 Rtp 描述信息和负载发送到 VCM 模块，继续接下来的 JitterBuffer 缓存和解码渲染操作。

RTP 报文解包过程是封包的逆过程，重要的输出信息是 RTP 头部描述和媒体负载，这些信息是下一步 JitterBuffer 缓存和解码的基础。另外对 RTP 报文进行统计得到的信息则是 RTCP RR 报文的数据来源。

四 RTCP报文发送和接收

RTCP 协议是 RTP 协议的控制下可以，负责流媒体的服务质量保证。比较常用的 RTCP 报文由发送端报告 SR 和接收端报告 RR，分别包含数据发送统计信息和数据接收信息。这些信息对于流媒体质量保证非常重要，比如码率控制、负载反馈，等等。其他 RTCP 报文还有诸如 SDES、BYE、SDES 等，RFC3550 对此有详细定义。

本节重点分析 WebRTC 内部 RTCP 报文的构造、发送、接收、解析、反馈等流程。需要再次强调的是，RTCP 报文的数据源来自 RTP 报文发送和接收时的统计信息。在 WebRTC 内部，RTCP 报文的发送采取周期性发送和及时发送相结合的策略：ModuleProcess 线程周期性发送 RTCP 报文；而 RtpSender 则在每次发送 RTP 报文之前都判断是否需要发送 RTCP 报文；另外在接收端码率估计模块构造出 REMB 报文后，通过设置超时让 ModuleProcess 模块立即发送 RTCP 报文。

4.1 RTCP报文构造和发送

在发送端，RTCP 以周期性发送为基准，辅以 RTP 报文发送时的及时发送和 REMB 报文的立即发送。发送过程主要包括 Feedback 信息获取、RTCP 报文构造、序列化和发送。图 4 描述了 RTCP 报文的构造和发送过程。

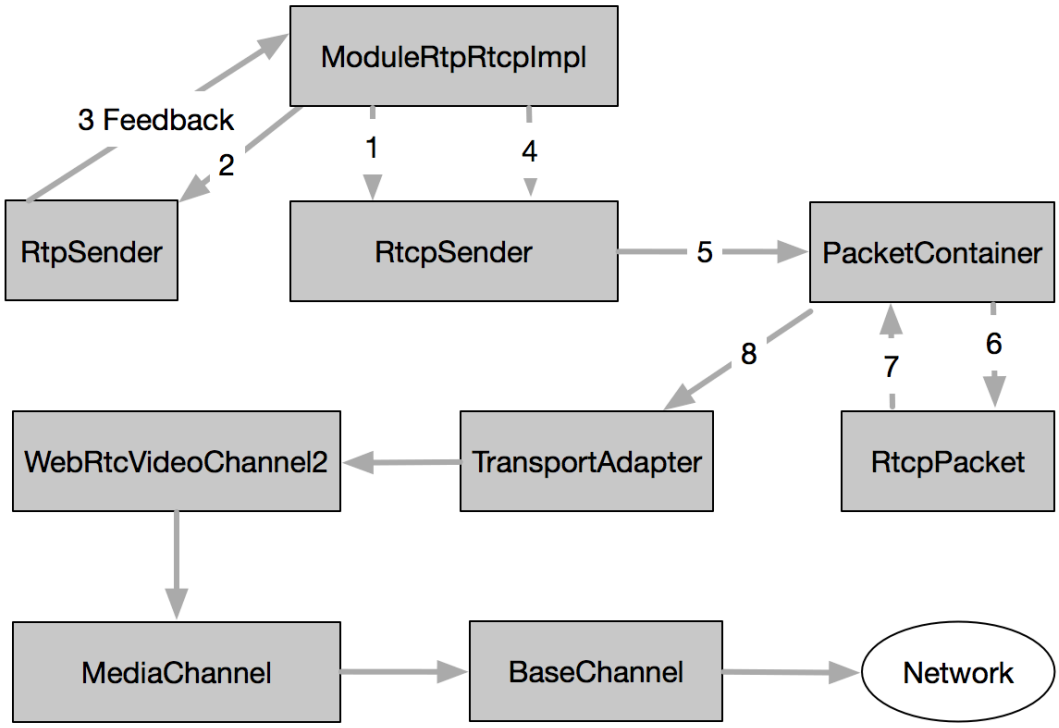

图4 RTCP报文构造和发送

ModuleProcess 线程周期性调用 ModuleRtpRtcpImpl::Process()函数，该函数通过 RTCPSender::TimeToSendRtcpReport()函数确定当前是否需要立即发送 RTCP 报文。若是，则首先从 RTPSender::GetDataCounters()获取 RTP 发送统计信息，然后调用 RTCPSender::SendRTCP()=>SendCompoundRTCP()发送 RTCP 组合报文。关于 RTCP 组合报文的定义，请参考文献[1]。

在 SendCompoundRTCP()函数中，首先通过 PrepareReport()确定将要发送何种类型的 RTCP 报文。然后针对每一种报文，调用其构造函数(如构造 SR 报文为 BuildSR()函数)，构造好的报文存储在 PacketContainer 容器中。在最后调用 PacketContainer::SendPackets()进行发送。

接下来每种 RTCP 报文都会调用各自的序列化函数，把报文序列化为网络字

节流。最后通过回调到达 PacketContainer::OnPacketReady()，最终把字节流发送到传输层模块：即通过 TransportAdapter 到达 BaseChannel，Network 线程调用传输层套接字 API 发送数据到网络。

RTCP 报文的构造和发送过程总体不是很复杂，最核心的操作就是获取数据源、构造报文、序列化和发送。相对来说构造报文和序列化比较繁琐，基于 RFC 定义的细节进行。

4.2 RTCP报文接收和解析

接收端的 RTCP 报文接收和解析过程如图 5 所示。

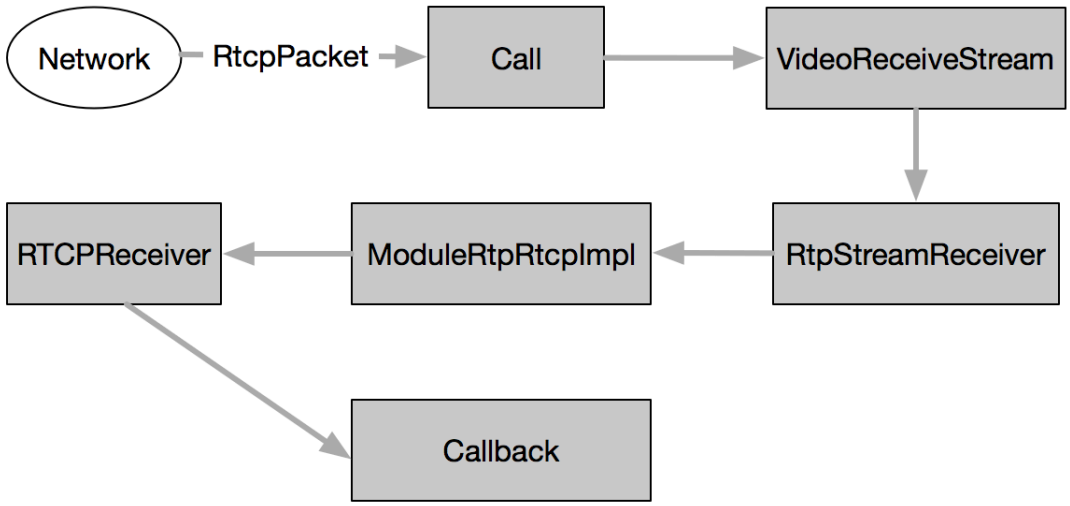

图4 RTCP报文接收和解析

在接收端，RTCP 报文的接收流程和 RTP 一样，都是经过网接收之后，到达 Call 对象，进而通过 SSRC 找到 VideoReceiveStream，继而到达 RtpStreamReceiver。接下来的解析和反馈操作都在 ModuleRtpRtcpImpl::IncomingRtcpPacket()中完成。该函数首先调用 RTCPReceiver::IncomingRtcpPacket()解析 RTCP 报文，得到

RTCPPacketInformation 对象，然后以该对象为参数调用RTCPReceiver::TriggerCallbacksFromRTCPPacket()，触发注册在此处的各路观察者执行回调操作。

RTCPReceiver::IncomingRtcpPacket()使用 RTCPParser 解析组合报文，针对每一种报文类型，调用对应的处理函数(如处理 SDES 的 HandleSDES 函数)，反序列化后拿到报文的描述结构。最后所有报文综合在一起形成 RTCPPacketInformation 对象。该对象接下来作为参数调用 TriggerCallbacksFromRTCPPacket()函数触发回调操作，如处理 NACK 的回调，处理 SLI 的回调，处理 REMB 的回调，等等。这些回调在各自模块控制流媒体数据的编码、发送、码率等服务质量保证，这也是 RTCP 报文最终起作用的地方。

至此，我们分析了 RTCP 报文发送和接收的整个流程。

五 总结

本文在深入分析 WebRTC 源代码的基础上，结合流程图描述出 RTP/RTCP 模块的实现流程，在关键问题上(如 RTCP 报文的数据来源)进行深入细致的研究。为进一步深入掌握 WebRTC 的实现原理和细节打下良好基础。

参考文献

[1] RFC3550 - RTP: A Transport Protocol for Real-Time Applications

https://www.ietf.org/rfc/rfc3550.txt

[2] 超越RFC3550 — RTP/RTCP协议族分析 :

http://www.jianshu.com/p/e5e21aeb219f