WebRTC 源代码分析

第一章 WebRTC 概述

1.1 WebRTC 概述

1.2 WebRTC 源码下载

1.3 GYP 构建系统

1.4 WebRTC 编译

1.5 WebRTC 开发流程

1.6 总结

1.7 参考文献

第二章 WebRTC 总体架构及源码目录

第三章 PeerConnection Demo 示例程序

第四章 PeerConnection 及 API

第五章 信道建立过程及 SDP

第六章 WebRTC MediaEngine

第七章 Video Data Pipeline

第八章 Audio Data Pipeline

第九章 RTP、RTCP 和 QoS

第十章 网络传输

第五章 Javascript API

总结

第一章 WebRTC 概述

1.1 WebRTC 概述

WebRTC(Web Real Time Communication)是一项免费开源工程，旨在通过简单的 API 接口向浏览器和移动应用提供实时通信的能力。WebRTC 的组件经过充分优化以达成这个目标。WebRTC 的使命：为浏览器、移动平台和物联网设备开发功能丰富、高质量的实时通信应用，并通过一系列通用协议使这些应用互联通信。

第九章 RTP、RTCP 和 QoS

9.1 RTPRTCP

9.1.1 理解 RTP 报头中的时间戳

Timestamp in rtp header – RFC3550.Section 5.1

The timestamp reflects the sampling instant of the first octet in the RTP data packet. The sampling instant must be derived from a clock that increments monotonically and linearly in time to allow synchronization and jitter calculations (see Section 6.4.1). The resolution of the clock must be sufficient for the desired synchronization accuracy and for measuring packet arrival jitter (one tick per video frame is typically not sufficient). The clock frequency is dependent on the format of data carried as payload and is specified statically in the profile or payload format specification that defines the format, or may be specified dynamically for payload formats defined through non-RTP means. If RTP packets are generated periodically, the nominal sampling instant as determined from the sampling clock is to be used, not a reading of the system clock. As an example, for fixed-rate audio the timestamp clock would likely increment by one for each sampling period. If an audio application reads blocks covering 160 sampling periods from the input device, the timestamp would be increased by 160 for each such block, regardless of whether the block is transmitted in a packet or dropped as silent.

时间戳反映了RTP数据包中第一个字节的采样时间。采样时间必须从一个单调线性递增的时钟中获取，以便允许RTP同步和抖动计算。时钟必须足够精确以支持预期同步精度和测量数据包到达抖动，因此每个视频帧一个滴答显然不够。时钟频率依赖于负载数据的格式，数据格式可由档次或者负载格式规格静态确定，或者通过非RTP方法动态确定。如果RTP数据包周期性产生，那么采样时间由采样时钟确定，而不是由系统时钟确定。例如，对于固定速率的音频来说，时间戳时钟会在每个采样周期内增加一。如果一个音频应用每160个周期从输入设备中读取数据块，那么时间戳会每次增加160，而不管该数据块是通过数据包传输还是丢弃)。

The initial value of the timestamp should be random, as for the sequence number. Several consecutive RTP packets will have equal timestamps if they are (logically) generated at once, e.g., belong to the same video frame. Consecutive RTP packets may contain timestamps that are not monotonic if the data is not transmitted in the order it was sampled, as in the case of MPEG interpolated video frames. (The sequence numbers of the packets as transmitted will still be monotonic.)

时间戳初始值应该像序列号那样是一个随机值。一些连续的RTP数据包如果逻辑上在同一时刻产生，比如属于同一视频帧，那么它们应该具有相同的时间戳。如果数据包没有按照它们的采样时间先后顺序进行传输，那么连续的RTP数据包可能包含非单调递增的时间戳，比如在MPEG中差值产生的视频帧；然而数据包的序列号依然是线性递增的。

RTP timestamps from different media streams may advance at different rates and usually have independent, random offsets. Therefore, although these timestamps are sufficient to reconstruct the timing of a single stream, directly comparing RTP timestamps from different media is not effective for synchronization. Instead, for each medium the RTP timestamp is related to the sampling instant by pairing it with a timestamp from a reference clock (wallclock) that represents the time when the data corresponding to the RTP timestamp was sampled. The reference clock is shared by all media to be synchronized. The timestamp pairs are not transmitted in every data packet, but at a lower rate in RTCP SR packets as described in Section 6.4.

来自不同媒体流的RTP时间戳可能以不同的速率增长，因此它们有各自独立随机的偏移量。因此，尽管对于单个媒体流来说比较时间戳足以重建时间序列，但是对于不同媒体流来说通过比较RTP时间戳来进行同步不是有效的方法。相反，对于每一个媒体流来说，RTP时间戳和参考时钟时间戳是配对的：参考时钟时间戳用来表示和RTP时间戳相关的RTP数据的采样时刻，参考时钟由所有同步源共享。时间戳对 并不会在每个数据包中都传输，但是以一个较低频率在RTCP SR数据包中进行传输。

The sampling instant is chosen as the point of reference for the RTP timestamp because it is known to the transmitting endpoint and has a common definition for all media, independent of encoding delays or other processing. The purpose is to allow synchronized presentation of all media sampled at the same time.

采样时刻被用来选作RTP时间戳的参考点，因为它对另一个传输端是已知的，并且对所有媒体有一个公共定义，独立于编码延迟和其它处理。这样做的目的是允许对同一时刻采样的所有媒体进行同步表示。

Applications transmitting stored data rather than data sampled in real time typically use a virtual presentation timeline derived from wallclock time to determine when the next frame or other unit of each medium in the stored data should be presented. In this case, the RTP timestamp would reflect the presentation time for each unit. That is, the RTP timestamp for each unit would be related to the wallclock time at which the unit becomes current on the virtual presentation timeline. Actual presentation occurs some time later as determined by the receiver.

传输非实时数据的应用通常使用源于参考时钟的时间作为虚拟显示时间线，以决定何时显示下一帧数据或者其它数据单元。在这种情况下，RTP时间戳反映数据单元的显示时间，也就是说，每个数据单元的RTP时间戳和参考时钟时间相关联，在这个时间点上数据单元在虚拟显示时间线上变为当前显示单元。事实上显示时间相比于接收端确定的时间稍微延迟。

An example describing live audio narration of prerecorded video illustrates the significance of choosing the sampling instant as the reference point. In this scenario, the video would be presented locally for the narrator to view and would be simultaneously transmitted using RTP. The “sampling instant” of a video frame transmitted in RTP would be established by referencing its timestamp to the wallclock time when that video frame was presented to the narrator. The sampling instant for the audio RTP packets containing the narrator’s speech would be established by referencing the same wallclock time when the audio was sampled. The audio and video may even be transmitted by different hosts if the reference clocks on the two hosts are synchronized by some means such as NTP. A receiver can then synchronize presentation of the audio and video packets by relating their RTP timestamps using the timestamp pairs in RTCP SR packets.

一个描述 预先录制视频的直播音频解说的例子说明了选择采样时间作为RTP时间戳参考点的意义。在这种场景下，视频将在本地呈现给解说者观看，并同时使用RTP进行发送传输。在RTP中传输的视频帧的采样时间参考视频帧呈现给解说者的参考时钟时间进行建立。包含解说者声音的RTP音频包的采样时刻同样参考相同的参考时钟的时间进行建立。音频和视频甚至有可能在不同主机上传输，如果这两台主机的参考时钟采用像NTP这样的方式同步。接收者可以使用RTCP SR数据包中的时间戳对来同步音视频的显示时间。

RTCP SR报文中关于时间戳的定义

NTP timestamp: 64 bits

Indicates the wallclock time (see Section 4) when this report was sent so that it may be used in combination with timestamps returned in reception reports from other receivers to measure round-trip propagation to those receivers. Receivers should expect that the measurement accuracy of the timestamp may be limited to far less than the resolution of the NTP timestamp. The measurement uncertainty of the timestamp is not indicated as it may not be known. On a system that has no notion of wallclock time but does have some system-specific clock such as “system uptime”, a sender may use that clock as a reference to calculate relative NTP timestamps. It is important to choose a commonly used clock so that if separate implementations are used to produce the individual streams of a multimedia session, all implementations will use the same clock. Until the year 2036, relative and absolute timestamps will differ in the high bit so (invalid) comparisons will show a large difference; by then one hopes relative timestamps will no longer be needed. A sender that has no notion of wallclock or elapsed time may set the NTP timestamp to zero.

RTP timestamp: 32 bits

Corresponds to the same time as the NTP timestamp (above), but in the same units and with the same random offset as the RTP timestamps in data packets. This correspondence may be used for intra- and inter-media synchronization for sources whose NTP timestamps are synchronized, and may be used by media-independent receivers to estimate the nominal RTP clock frequency. Note that in most cases this timestamp will not be equal to the RTP timestamp in any adjacent data packet. Rather, it must be calculated from the corresponding NTP timestamp using the relationship between the RTP timestamp counter and real time as maintained by periodically checking the wallclock time at a sampling instant.

9.2 WebRTC PacedSending 工作原理

以30ms为操作间隔IntervalBudget，每个间隔有固定的media budget，该budget根据设定的target_bps计算的到。一个interval内buget如果没用完，那么下个interval就不能用；一个interval内的发送量如果超过budget，那么下个interval就不发或者少发，直到这个坑填平再开始发送。在某些情况下，还通过需要发送padding使数据量维持在min_bps以上，而实际上min_bps设定为0，也就是说不发送padding数据。

pacedsending的process()的调用间隔为每6~16ms调用一次。