WebRTC支持H264编解码 - 简书

一、引言

众所周知，Chrome/WebRTC中的视频编解码器一直使用Google自己开发的VP8/VP9，而对于业界广泛使用的H264则支持有限。他们这么做除了推广自家产品外，还有一个很好的理由：专利。VP8/VP9是免专利费的，而H264则需要专利授权。因此，Google Chrome在2011年的时候甚至放弃对H264的支持。

但是，随着H264的发展，Chrome不得不再次考虑对H264的支持问题，特别是思科发布H264开源实现openh264后。最近，Google宣布从Chrome50开始支持H264视频编解码，用户在打开Chrome时通过如下命令行参数即可启用：

```
--enable-features=WebRTC-H264WithOpenH264FFmpeg
```

这无疑是一个好消息，尤其对研究WebRTC的开发者来说，在WebRTC中启用H264可以明显带来如下收益：

1. 浏览器互操作：除Chrome和Firefox外，目前微软Edge浏览器的ORTC也开始支持H264/AVC。

2. 移动设备支持：移动设备基本上都支持H264硬件编解码。

3. 遗留系统互连：遗留视频系统基本上都使用H264，绝大多数不支持VP8。使用H264可使得WebRTC和这些系统方便互通。

	Audio	Video
Google	G.711, Opus	VP8, VP9
mozilla	G.711, Opus	VP8, H.264
Microsoft	G.711, G.722, Opus	H.264
	AAC-ELD	H.264 / H.265
ERICSSON	G.711, Opus	VP8, H.264

source: BlogGeek.Me

H264可以极大提高WebRTC浏览器互操作

目前看起来形势一片大好，在WebRTC中启用H264收益多多。但是，通过研究Chromium/WebRTC源代码我们发现，目前H264只能够在Chromium浏览器中通过命令行参数启用，在WebRTC中则支持有限。

具体情况是：默认不启用H264；编码器使用openh264 Encoder；解码器使用ffmpeg Decoder；WebRTC代码中针对H264没有相应的支持和优化。

因此，要想在WebRTC中启用H264，还有很多事情要做。

二、规划

根据上节对情况的分析，要想在WebRTC中支持H264以获取其带来的收益，需要解决以下问题：

1. 选定Chrome 50以上版本的codebase为基础。
2. 编译WebRTC源代码，得到最新代码的库及Demo程序。
3. 解除WebRTC源代码对Chromium的依赖。（可选）
4. 替换FFmpeg Decoder为openh264 Decoder。
5. 添加WebRTC对H264的适配代码。
6. 测试

本文假定读者已经具有下载WebRTC源代码并编译成功的经验，并且对GYP构建系统也有一定的了解，以及如何设置环境变量，等等。另外，本文以Windows平台为例行文，WebRTC基于GYP构建系统，具有平台无关性。

三、实施

3.1 WebRTC codebase更新到Chrome 52

在WebRTC根目录下运行如下命令[1]：

```
gclient sync --with_branch_heads

git fetch origin

git checkout –b branch_52branch-heads/52

gclient sync
```

3.2 编译WebRTC

在Windows平台上目前需要Visual Studio 2015 Update 2以上的版本，官方推荐使用Professional版，但是Community版也可以。安装时要选择定制安装，选择Visual C++，通用Windows应用开发工具->工具，通用Windows应用开发工具->Windows 10 SDK(10.0.10586)这三个工具[2]。

另外，需要Windows操作系统为64位系统。且系统语言为英文，否则，编译时会产生’error C2220:警告被视为错误-没有生成“object”文件’错误。

接下来需要设置环境变量：

```
set GYP_DEFINES=proprietary_codecs=1 build_with_chromium=0

set GYP_GENERATORS=ninja,msvs

set GYP_MSVS_VERSION=2015

gclient runhooks
```

其中，proprietary_codecs表示要启用H264 codec。上述执行完毕，即在src目录下生成VS2015工程文件。用VS 2015打开all.sln即可进行编译。

3.3 解除对Chromium的依赖

本步骤对WebRTC支持H264本身没有关系。但是考虑到简化codebase，还是在这里叙述一下。

WebRTC依赖大量第三方库，这些库在chromium目录中，通过符号链接指向webrtc本地目录third_party下。chromium源代码多达12GB，而我们只关心符号链接指向的第三方库，这部分源代码只有1.6GB。因此，为减小源代码大小和方便管理，可以通过拷贝符号链接目标然后删除chromium目录。可通过脚本一键执行该任务。

3.4 适配H264Codec

3.4.1 解除WebRTC对FFmpeg的依赖

如上文所说，Chrome使用FFmpeg的Decoder，因此如果我们想启用H264 Decoder，需要解除对FFmpeg的依赖。主要修改webrtc/build/common.gypi和webrtc/modules/video_coding/codecs/h264/h264.gypi两个配置文件，在后者同时添加对openh264 decoder的依赖。

3.4.2 生成openh264 decoder工程

这部分主要修改third_party/openh264/openh264.gyp和openh264.gypi文件，仿照encoder工程文件的生成，添加对decoder工程文件生成的gypi定义。

3.4.3 启用openh264 codec汇编优化

third_party/openh264工程中的gyp文件默认不启用汇编优化，这对openh264的性能造成极大影响，为此开启汇编优化。这部分gyp文件编写较为复杂，参考了third_party/boringssl库中gyp编译汇编文件的写法。另外，为编译汇编代码，需要安装2.10.6+版本的nasm汇编器。

3.4.4 为openh264 decoder生成适配类

WebRTC中已经实现H264DecoderImpl类以适配对H264 decoder的调用，不过这里调用的是FFmpeg的decoder。为此重新定义H264DecoderImpl类，真正调用openh264 decoder的API。该部分内容参考了文档[4]。

3.4.5 修改SDP协商Codec优先顺序

WebRTC默认优先使用VP8进行编解码，SDP协商的时候VP8也是放在最前面。如果要默认使用H264，需要修改webrtc/media/engine/webrtcvideoengine2.cc中的DefaultVideoCodecList()函数，把H264部分代码提到函数开始处[3]。

至此，代码修改部分完成。下面即可进行编译测试工作。

4 测试

验证SDP offer和answer中关于video codec最终协商为H264。

验证两个Native Client P2P测试成功，使用H264编解码。

验证Native Client和Chrome P2P测试成功，使用H264编解码。

5 总结

本文根据Chrome/WebRTC的代码演化，以实际工作中的经验教训为基础，总结了在WebRTC中支持H264 Codec的工作流程。通过此次实践，我们可以在WebRTC中获得H264带给我们的优势，为我们的应用提供更好的性能，更大的应用场景和更多的可能性。

参考文档：

1. https://webrtc.org/native-code/development/

2. https://chromium.googlesource.com/chromium/src/+/master/docs/windows_build_instructions.md

3. http://blog.csdn.net/doitsjz/article/details/51787567

4. http://blog.csdn.net/xyblog/article/details/50433118