WebRTC 添加 H264 codec 支持

WebRTC 从 Chrome 50 开始支持 H264 Codec，在 52 版本的 Release Note 中宣布全功能支持 H264。但是通过调研我们发现 H264 Codec 只在特定平台的 Chrome 浏览器中启用，而在 WebRTC 内部并不可用。

本文总结了在 WebRTC 内部支持 H264 的工作流程。

1 WebRTC 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_52 branch-heads/52
git sync
```

2 编译

2.1 安装 VS2015

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 操作系统的系统语言为英文，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。

2.2 设置环境变量并生成工程文件

```
set GYP_DEFINES=include_tests=0 proprietary_codecs=1
                build_with_chromium=0
set GYP_GENERATORS=ninja,msvs
set GYP_MSVS_VERSION=2015
gclient runhooks
```

其中，proprietary_codecs 是为添加 H264 codec 做准备。上述执行完毕，即生成 VS2015 工程文件。打开 all.sln 即可进行编译。

3 解除对 Chromium 代码的依赖

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 本地目录下。chromium 源代码多达 12GB，而我们只关心符号链接的内容。因此，可以通过拷贝符号链接目标然后删除 chromium 源代码，来减小整个 WebRTC code base 的大小。

相关运行脚本添加到 src/remove_softlinks.sh。

4 WebRTC 支持 H264 Codec

我们本次更新 webrtc 52 的目的就是使用 H264 Codec，但是 WebRTC 默认不启用 H264。并且就算启用，也只用 encoder 而不用 decoder。Decoder 采用 ffmpeg 的 decoder，而且还有一些问题［3］。因此想要完全使用 H264 Codec，需要做一些工作。以下是任务列表：

4.1 解除 WebRTC 对 ffmpeg 的依赖

ffmpeg 由 H264decoder 使用，因此主要修改 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。

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。

具体改动细节请查看 Code Base 中该文件。

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。为了尽量不影响原 gyp 文件，新添加 openh264_assemble.gypi 文件。

具体改动细节请查看 Code Base 中相关文件。

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 的 decoder 库。该部分内容参考了文档[4]，根据 chrome 52 code base 做了适当修改。

4.5 修改 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。

5 测试

5.1 验证 SDP 协商 H264

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

5.2 Native VS Native 测试

两个 Native Client P2P 测试成功，使用 H264 编解码。

5.3 Native VS Chrome 测试

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

6 总结

本文总结了 WebRTC 代码更新到 Chrome 52 版本及默认启用 H264 Codec 的工作流程，难度不是很大。通过本任务，加深了对 WebRTC 代码结构的理解，对 openh264 的代码结果和 API 也有初步的认识。

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