# 附件：

## Gyp工具

GYP 简介:转载自：http://blog.xiaogaozi.org/2011/10/29/introduction-to-gyp/

说起项目构建工具，Linux 用户最熟悉的恐怕就是 [Autotools](http://en.wikipedia.org/wiki/GNU_build_system)，它将编译安装这个步骤大大简化。但对于项目作者来说，想要使用 Autotools 生成有效的配置文件着实需要下一番功夫，用现在流行的话来说就是用户体验不够友好。对 Unix shell 的依赖，也使得 Autotools 天生对于跨平台支持不佳。

与其类似的有 [CMake](http://www.cmake.org/)，CMake 使用 C++ 编写，原生支持跨平台，不需要像 Autotools 那样写一堆的配置文件，只需一个 CMakeLists.txt 文件即可。简洁的使用方式，强大的功能使得我立马对 CMake 情有独钟。在后来的使用过程中，虽然会遇到一些因为使用习惯带来的小困扰，但我对于 CMake 还是基本满意的。直到我发现了 GYP。

[GYP](http://code.google.com/p/gyp/)（Generate Your Projects）是由 Chromium 团队开发的跨平台自动化项目构建工具，Chromium 便是通过 GYP 进行项目构建管理。为什么我要选择 GYP，而放弃 CMake 呢？功能上 GYP 和 CMake 很是相似，在我看来，它们的最大区别在于配置文件的编写方式和其中蕴含的思想。

编写 CMake 配置文件相比 Autotools 来说已经简化很多，一个最简单的配置文件只需要写上源文件及生成类型（可执行文件、静态库、动态库等）即可。对分支语句和循环语句的支持也使得 CMake 更加灵活。但是，CMake 最大的问题也是在这个配置文件，请看下面这个示例文件：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | cmake\_minimum\_required(VERSION 2.8)  project(VP8 CXX)  add\_definitions(-Wall)  cmake\_policy(SET CMP0015 NEW)  include\_directories("include")  link\_directories("lib")  set(CMAKE\_ARCHIVE\_OUTPUT\_DIRECTORY "../lib")  set(VP8SRC VP8Encoder.cpp VP8Decoder.cpp)  if(X86)  set(CMAKE\_SYSTEM\_NAME Darwin)  set(CMAKE\_SYSTEM\_PROCESSOR i386)  set(CMAKE\_OSX\_ARCHITECTURES "i386")  add\_library(vp8 STATIC ${VP8SRC})  elseif(IPHONE)  if(SIMULATOR)  set(PLATFORM "iPhoneSimulator")  set(PROCESSOR i386)  set(ARCH "i386")  else()  set(PLATFORM "iPhoneOS")  set(PROCESSOR arm)  set(ARCH "armv7")  endif()  set(SDKVER "4.0")  set(DEVROOT "/Developer/Platforms/${PLATFORM}.platform/Developer")  set(SDKROOT "${DEVROOT}/SDKs/${PLATFORM}${SDKVER}.sdk")  set(CMAKE\_OSX\_SYSROOT "${SDKROOT}")  set(CMAKE\_SYSTEM\_NAME Generic)  set(CMAKE\_SYSTEM\_PROCESSOR ${PROCESSOR})  set(CMAKE\_CXX\_COMPILER "${DEVROOT}/usr/bin/g++")  set(CMAKE\_OSX\_ARCHITECTURES ${ARCH})  include\_directories(SYSTEM "${SDKROOT}/usr/include")  link\_directories(SYSTEM "${SDKROOT}/usr/lib")  add\_definitions(-D\_PHONE)  add\_library(vp8-armv7-darwin STATIC ${VP8SRC})  endif() |

你能一眼看出这个配置文件干了什么吗？其实这个配置文件想要产生的目标（target）只有一个，就是通过 ${VP8SRC} 编译生成的静态库，但因为加上了条件判断，及各种平台相关配置，使得这个配置文件看起来很是复杂。在我看来，编写 CMake 配置文件是一种线性思维，对于同一个目标的配置可能会零散分布在各个地方。而 GYP 则相当不同，GYP 的配置文件更多地强调模块化、结构化。看看下面这个示例文件：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | {  'targets': [  {  'target\_name': 'foo',  'type': '<(library)',  'dependencies': [  'bar',  ],  'defines': [  'DEFINE\_FOO',  'DEFINE\_A\_VALUE=value',  ],  'include\_dirs': [  '..',  ],  'sources': [  'file1.cc',  'file2.cc',  ],  'conditions': [  ['OS=="linux"', {  'defines': [  'LINUX\_DEFINE',  ],  'include\_dirs': [  'include/linux',  ],  }],  ['OS=="win"', {  'defines': [  'WINDOWS\_SPECIFIC\_DEFINE',  ],  }, { # OS != "win",  'defines': [  'NON\_WINDOWS\_DEFINE',  ],  }]  ],  }  ],  } |

我们可以立马看出上面这个配置文件的输出目标只有一个，也就是 foo，它是一个库文件（至于是静态的还是动态的这需要在生成项目时指定），它依赖的目标、宏定义、包含的头文件路径、源文件是什么，以及根据不同平台设定的不同配置等。这种定义配置文件的方式相比 CMake 来说，让我觉得更加舒服，也更加清晰，特别是当一个输出目标的配置越来越多时，使用 CMake 来管理可能会愈加混乱。

配置文件的编写方式是我区分 GYP 和 CMake 之间最大的不同点，当然 GYP 也有一些小细节值得注意，比如支持跨平台项目工程文件输出，Windows 平台默认是 Visual Studio，Linux 平台默认是 Makefile，Mac 平台默认是 Xcode，这个功能 CMake 也同样支持。Chromium 团队成员也撰文详细[比较](http://code.google.com/p/gyp/wiki/GypVsCMake)了 GYP 和 CMake 之间的优缺点，在开发 GYP 之前，他们也曾试图转到 [SCons](http://www.scons.org/)（这个我没用过，有经验的同学可以比较一下），但是失败了，于是 GYP 就诞生了。

当然 GYP 也不是没有缺点，相反，我觉得它的「缺点」一大堆：

文档不够完整，项目不够正式，某些地方还保留着 Chromium 的影子，看起来像是还没有完全独立出来。

大量的括号嵌套，很容易让人看晕，有过 Lisp 使用经验的同学可以对号入座。对于有括号恐惧症，或者不使用现代编辑器的同学基本可以绕行。

为了支持跨平台，有时不得不加入某些特定平台的配置信息，比如只适用于 Visual Studio 的 RuntimeLibrary 配置，这不利于跨平台配置文件的编写，也无形中增加了编写复杂度。

不支持 make clean，唯一的方法就是将输出目录整个删除或者手动删除其中的某些文件。

如果你已经打算尝试 GYP，那一定记得在生成项目工程文件时加上 --depth 参数，譬如：

$ gyp --depth=. foo.gyp

这也是一个从 Chromium 项目遗留下来的历史问题。

## Google test程序

[玩转Google开源C++单元测试框架Google Test系列(gtest)(总)](http://www.cnblogs.com/coderzh/archive/2009/04/06/1426755.html)

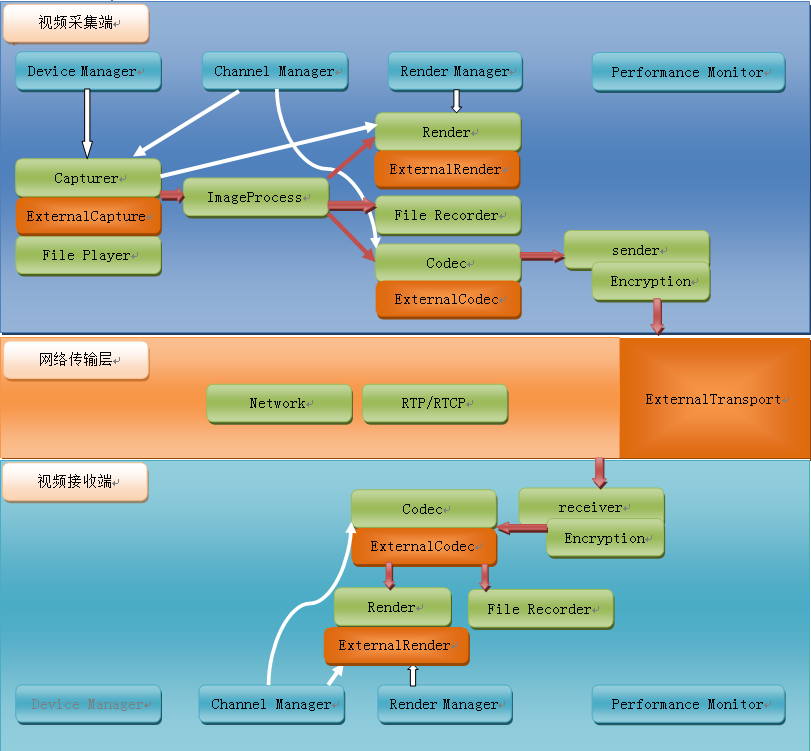
## Webrtc库介绍

trunk\webrtc\modules

**视频采集---video\_capture**

    源代码在webrtc/modules/video\_capture/main目录下，包含接口和各个平台的源代码。  
    在windows平台上，WebRTC采用的是dshow技术，来实现枚举视频的设备信息和视频数据的采集，这意味着可以支持大多数的视频采集设备；对那些需要单独驱动程序的视频采集卡（比如海康高清卡）就无能为力了。  
    视频采集支持多种媒体类型，比如I420、YUY2、RGB、UYUY等，并可以进行帧大小和帧率控制。  
  
**视频编解码---video\_coding**   
    源代码在webrtc/modules/video\_coding目录下。  
    WebRTC采用I420/VP8编解码技术。VP8是google收购ON2后的开源实现，并且也用在WebM项目中。VP8能以更少的数据提供更高质量的视频，特别适合视频会议这样的需求。  
  
**视频加密--video\_engine\_encryption**   
    视频加密是WebRTC的video\_engine一部分，相当于视频应用层面的功能，给点对点的视频双方提供了数据上的安全保证，可以防止在Web上视频数据的泄漏。  
    视频加密在发送端和接收端进行加解密视频数据，密钥由视频双方协商，代价是会影响视频数据处理的性能；也可以不使用视频加密功能，这样在性能上会好些。  
    视频加密的数据源可能是原始的数据流，也可能是编码后的数据流。估计是编码后的数据流，这样加密代价会小一些，需要进一步研究。  
  
**视频媒体文件--media\_file**   
    源代码在webrtc/modules/media\_file目录下。  
    该功能是可以用本地文件作为视频源，有点类似虚拟摄像头的功能；支持的格式有Avi。  
    另外，WebRTC还可以录制音视频到本地文件，比较实用的功能。  
  
**视频图像处理--video\_processing**   
    源代码在webrtc/modules/video\_processing目录下。  
    视频图像处理针对每一帧的图像进行处理，包括明暗度检测、颜色增强、降噪处理等功能，用来提升视频质量。  
  
**视频显示--video\_render**   
    源代码在webrtc/modules/video\_render目录下。  
    在windows平台，WebRTC采用direct3d9和directdraw的方式来显示视频，只能这样，必须这样。

**网络传输与流控**   
    对于网络视频来讲，数据的传输与控制是核心价值。WebRTC采用的是成熟的RTP/RTCP技术。



## [webrtc代码相关基础知识](http://blog.csdn.net/chenyufei1013/article/details/8156116)

<http://blog.csdn.net/chenyufei1013/article/category/1248211>

## STUN和TURN技术浅析

## 基于ICE的VoIP穿越NAT改进方案

1 引言

    近年来，随着数据网络通信逐渐融入传统的话音业务领域，VoIP技术越来越成为当前商业考虑的对象，并正在向一种正式的商业电话模式演进，而会话初始协议 (SIP，Session Initiation Protoc01)就是用来确保这种演进能够实现而需要的NGN(下一代网络)系列协议中重要的一员。SIP是一个用于建立，更改和终止多媒体会话的应用层控制协议。SIP因其简单、灵活、可扩展性强的特点，已经成为实现VolP系统的热点技术。

    随着计算机网络技术的不断发展，互联网规模飞速膨胀，大量企业和驻地网采用了私有网络通过NAT/防火墙出口来接入公共网络。而由于SIP包头中含有很多 对于路由、接续SIP信令和建立呼叫连接必不可少的地址信息，这样引发了业界对于SIP2穿越NAT/防火墙问题的研究。

    目前，IETF已经对该问题提出了多种解决方案。例如：ALCes(Application Layer Gateways)、MiddleboxControl Protocol、STUN Simple Traversal of UDPthrough NAT)、TURN(Traversal Using Relay NAT)、RSIP(Realm Specific IP)、Symmetric RTP等。然而，当这些技术应用于不同的网络拓扑时都有着显著的利弊，以至于只能根据不同的接入方式来应用不同的方案，所以，未能很好地解决A11- NATⅢ的问题，同时还会给系统引入许多复杂性和脆弱性因素。此外，由于NAT/防火墙已经大量应用，SIP设备也已经比较成熟，对它们进行升级来支持多 媒体通信穿越NAT/防火墙的代价将相当的大。因此，一种不需要升级任何现有网络设备，能够穿越各种NAT/防火墙并且方便在现有网络中实施的解决方案成 为迫切的需要。

    本文试图寻找一种能够穿越各种类型的NAT/防火墙，无需对现有NAT/防火墙没备做任何改动的解决方案——ICE解决方案，这种方式比以前的解决方案更加灵活，具有广阔的应用前景。

**2 现有NAT解决方案的比较分析**

    主流的NAT穿越解决方案包括STUN、TURN、Proxy及隧道穿越等，这几种方式各具优缺点，比较如下：

    (1)STUN(simple traversal of UDP over NAT)的原理是通过某种机制预先得到内部私有IP地址对应在出口NAT上的对外公网IP地址，然后在报文负载中所描述的地址信息就直接填写出口NAT上 的对外IP地址。其最大的优点是无需对现有NAT/防火墙设备做任何改动。局限性在于需要应用程序支持STUN CLIENT的功能，同时STUN并不适合支持TCP连接的穿越。

    (2)TURN即通过Relay方式穿越NAT，也是私网中的SIP终端通过某种机制预先得TURN SeI-ver上的公网地址，私网终端发出的报文都要经过TURN Serve：进行Relay转发。这种方式除了具有STUN方式的优点外，还解决了STUN应用无法穿透对称NAT(SymmetricNAT)以及类似 的Firewall设备的缺陷，局限性在于需要SIP终端支持TURN Client，并增大了包的延迟和丢包的可能性。

    (3)Proxy方式是指通过对私网内用户呼叫的信令和媒体||d时做Relay来实现对NAT/防火墙的穿越。由于不用对运营商和客户端的现有网络设备进行任何改造，具有很强的适应性，组网灵活，可满是NGN初期多样化的组网和用户接入。

    (4)隧道穿越技术的基本思想是通过把需要穿越的数据流封装征某种隧道中，从而绕过NAT/防火墙。它在很大程度上解决了对于不问应用协议需要开发不同穿越策略的办法，但是必须多媒体终端和服务器能够支持隧道，这是一个比较大的限制条件。

**3 穿越NAT/防火墙方案的实现**

**3．1 ICE方式**

    交互式连通建立方式ICE(Interactive ConnectivityEstablishment)并非一种新的协议，它不需要对STUN，TURN或RSIP进行扩展就可适用于各种NAT。ICE 是通过综合运用上面某几种协议，使之征最适合的情况下工作，以弥补单独使用其中任何一种所带来的固有缺陷。对于SIP来说，ICE只需要定义一些 SDP(Sessionescription Protoc01)附加属性即可，对于别的多媒体信令协议也需要制定一些相应的机制来实现。本文是针对SIP呼叫流程实现ICE的功能。

    这种方式的优点是可以根据通讯双方所处的网络环境，选取适合穿越NAT/防火墙的方式。首先，获取用户所征网络中NAT的类型，如果用户没有设置使用何种 方式连接，那么默队首先使用UDP连接，如果一定时间内没有连接成功，接着使用TCP连接，同样如果没有在一定时间内连接成功，那么将采用其他方式如 Upnp、Httptunnel。如果所有穿越方案都失败后，将结果返回给用户，由用户决定是否重试。

**3．2 ICE算法流程**

    ICE算法流程分为以F几个过程：

    (1)收集本地传输地址

    会话者从服务器上获得主机上一个物理(或虚拟)接口绑定一个端口的本地传输地址。

    (2)启动STUN

    与传统的STUN不同，ICE用户名和密码可以通过信令协议进行交换。

    (3)确定传输地址的优先级

    优先级反映了UA在该地址上接收媒体流的优先级别，取值范围0到1之间，按照被传输媒体流量来确定。

    (4)构建初始化信息(Initiate Message)

    初始化消息由一系列媒体流组成，每个媒体流的任意Peer之间实现最人连通可能性的传输地址是由公网L转发服务器(如TURN)提供的地址。

    (5)响应处理

    连通性检查和执行ICE算法中描述的地址收集过程。

    (6)生成接受信息(Accept Message)

    若接受则发送Accept消息，其构造过程与InitiateMessage类似。

    (7)接受信息处理

    接受过程需要发起者使用Send命令，由服务器转发至响应者。

    (8)附加ICE过程

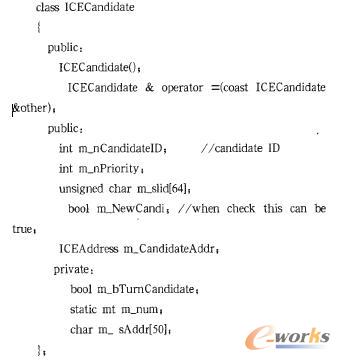
    Initiate或Accept消息交换过程结束后，双方可能仍将继续收集传输地址。

**3．3 ICE算法实现**

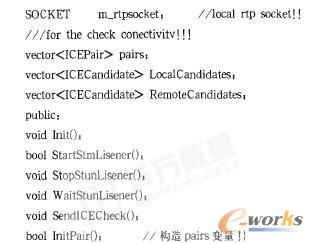
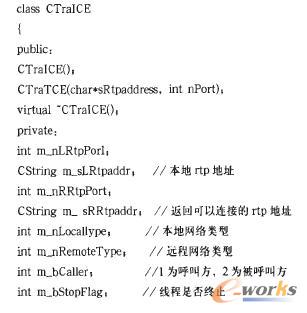
    当将ICE算法集成到SIP呼叫过程的时候，流程应该是：(1)SIP终端注册，并且访问STUN(STUNT)服务器，判断NAT/防火墙类型，以及 TCP时三种序列的包的过滤情况。(2)当发起呼叫信息(INVITE)或接收到呼叫信息回应(200 OK)之前根据NAT/防火墙类型进行对RTP进行地址收集(任非对称性NAT/防火墙后需要收集NAT映射地址，在对称性NAT/防火墙后还需要收集 TURN地址)。(3)在RTP的地址端口启动接收线程RSTUN服务程序。(4)发送SIP消息，收集的地址放列SDP消息中的alt属性中。 (5)SIP终端得到通讯双方地址后进行地址配对(将双方地址进行组合)，并且根据双方网络情况去掉无效的地址对。(6)根据地址对发

    送STUN check的包，其中STUN消息的用户名，密码从alt属性中得到，标识该地址对。(7)当检测到有效的地址对时(可以发送RTP媒体流的地址)，停止接收线程STUN服务)，开始传输RTP流。

    本文实现采用Winpcap API首先捕获TCP连接的SYN--out包，修改lP包头的TTL的值，用pcap—sendpacket()。然后使该socket调用listen函数。实现过程中对应于ICE收集地址的算法描述为：



    类中m\_nCandidateID对应地址序号，m\_nPriority表示地址优先级，m\_CandidateAddr表示地址(IP地址，端口)。实现ICE算法的实体算法描述为：



    实现ICE中会话发起者和接收者的步骤基本一样，仅任处理流程中先后次序稍微有些不同，本文中实现的会话流程如图l所示。

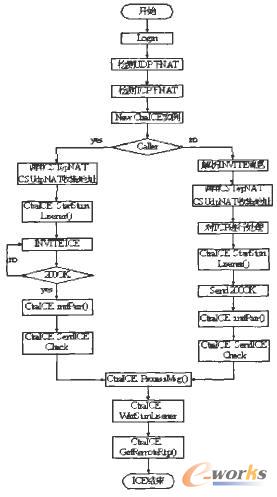


    图1会话流程

**4 测试**

    以安装了SIP软终端的双方都在Full ConeNATNAT/防火墙后为例，进行实例测试。测试过程：

    (1)将两台PC的IP的配置分别为公网59．64．148．187122和私网10．0．0．5/8l

    (2)从公网中的用户代理向私网内的用户代理呼叫，结果能够建立会话，无明显的延时，话音质量良好；

    (3)从私网内的用户代理向公网中的用户代理呼叫，结果能够建立会话，且话音质量良好；

    通过抓包分析可以确定，使用该算法可以成功地穿越NAT/防火墙设备。

**5 结论**

  ICE方式的优势是显而易见的，它消除了现有的机制的许多脆弱性。例如，传统的STUN有几个脆弱点，其中一个就是发现过程需要客户端自己去判断所在 NAT类型，这实际上不是一个可取的做法。而应用ICE之后，这个发现过程己经不需要了。另一点脆弱性在于STUN，TURN等机制都完全依赖于一个附加 的服务器，而ICE利用服务器分配单边地址的同时，还允许客户端直接相连，因此即使STUN或TRUN服务器中有任何一个失败了，ICE方式仍可让呼叫过 程继续下去。此外，传统的STUN最大的缺陷在于，它不能保证在所有网络拓扑结构中都正常工作，对于TURN或类似转发方式工作的协议来说，由于服务器的 负担过重，很容易出现丢包或者延迟情况。而ICE方式正好提供了一种负载均衡的解决方案，它将转发服务作为优先级最低的服务，从而在最大程度上保证了服务 的可靠性和灵活性。此外，ICE的优势还在于对IPv6的支持。由于广泛的适应能力以及对未来网络的支持，ICE作为一种综合的解决方案将有着非常广阔的 应用前景。

## ubuntu安装使用stuntman

1. 官网地址（要翻墙）：STUNTMAN：<http://www.stunprotocol.org/>

从<http://www.stunprotocol.org/stunserver-1.2.3.tgz> 下载源码

2. 编译依赖：

sudo apt-get install g++

sudo apt-get install make

sudo apt-get install libboost-dev # For Boost

sudo apt-get install libssl-dev # For OpenSSL

3. 编译 stunserver

cd stunserver

sudo make

4. 在stunserver目录下生成下面三个程序

stunclient, stunserver, stuntestcode

5. run the unit test. Should HAVE NO "FAIL" in the end of any line

./stuntestcode

Result of CTestDataStream: PASS

Result of CTestReader: PASS

Result of CTestBuilder: PASS

Result of CTestIntegrity: PASS

Result of CTestMessageHandler: PASS

Result of CTestCmdLineParser: PASS

Testing detection for DirectMapping

Testing detection for EndpointIndependent mapping

Testing detection for AddressDependentMapping

Testing detection for AddressAndPortDependentMapping

Testing detection for EndpointIndependentFiltering

Testing detection for AddressDependentFiltering

Testing detection for AddressAndPortDependentFiltering

Result of CTestClientLogic: PASS

Result of CTestRecvFromEx(IPV4): PASS

Result of CTestRecvFromEx(IPV6): PASS

Result of CTestFastHash: PASS

Result of CTestPolling: PASS

Result of CTestAtomicHelpers: PASS

6. start stun server......

./stunserver --help     # 使用说明。

nohup ./stunserver --mode full --primaryinterface eth0 --altinterface eth1 &

7. stunclient 检测地址端口映射及NAT类型

用法：./stunclient --mode full --localport 7777 [stun.sipgate.net](http://stun.sipgate.net/)

NOTE: stuntman只具有stun功能，没有转发功能。支持UDP，TCP。兼容RFC3489。

## 一个开源的ICE库——libnice介绍

原文地址：<http://blog.csdn.net/kl222/article/details/19336179>

libnice是一个ICE实现库。它实现了Interactive Connectivity Establishment (ICE) standard (RFC 5245) 和 the Session Traversal Utilities for NAT (STUN) standard (RFC 5389)。

官网地址：http://nice.freedesktop.org/wiki/

源码git库地址：http://cgit.collabora.com/git/libnice.git

1、下载源码：

git clone git://git.collabora.co.uk/git/libnice.git  
  
2、编译：

2.1、linux平台下：

它依赖：

 glib >= 2.10  
 pkg-config  
 gupnp-igd >= 0.1.2 (optional)  
 gstreamer-0.10 >= 0.10.0 (optional)  
gtk-doc-tools     #autogen.sh需要

2.2、编译

k@k-C410:/home/libnice$ ./autogen.sh   
  
k@k-C410:/home/libnice$ ./configure

k@k-C410:/home/libnice$ make

3、生成的程序和库

在nice/.libs目录下生成静态库libnice.a、动态库libnice.so   
在example目录下生成三个例子程序。

4、例子程序的使用

k@k-C410:/home/libnice/examples$ ./simple-example 0 stunserver.org  
Copy this line to remote client:  
  
  Tyyp 33oInvKVEn1Lo6LkVVy6P5 1,2013266431,192.168.10.17,47748,host  
  
Enter remote data (single line, no wrapping):  
>

红色部分表示提供给对等端协商时的验证用户名、密码、外网地址，以空格分隔。

启动二个实例，就可以开始IM对话了：

第一个控制台：

k@k-C410:/home/libnice/examples$ ./simple-example 0 stunserver.org  
Copy this line to remote client:  
  
  Tyyp 33oInvKVEn1Lo6LkVVy6P5 1,2013266431,192.168.10.17,47748,host  
  
Enter remote data (single line, no wrapping):  
> h4p1 7M8uL1928RzeRv6cWRDqG8 1,2013266431,192.168.10.17,47758,host  
  
Negotiation complete: ([192.168.10.17]:47748, [192.168.10.17]:47758)  
  
Send lines to remote (Ctrl-D to quit):  
> a  
>   
  
第二个控制台：  
  
k@k-C410:/home/libnice/examples$ ./simple-example 0 stunserver.org  
Copy this line to remote client:  
  
  h4p1 7M8uL1928RzeRv6cWRDqG8 1,2013266431,192.168.10.17,47758,host  
  
Enter remote data (single line, no wrapping):  
> Tyyp 33oInvKVEn1Lo6LkVVy6P5 1,2013266431,192.168.10.17,47748,host  
  
Negotiation complete: ([192.168.10.17]:47758, [192.168.10.17]:47748)  
  
Send lines to remote (Ctrl-D to quit):  
> a

存在的问题：在linux下，stun服务器地址不能通过域名解析到IP地址。解决方法是，直接用stun服务器的IP地址。本人已向项目提交了补丁包。