

FreeSWITCH

文 集

杜金房 著

INVITE sip:+861234567890@example.com SIP/2.0
Via: SIP/2.0/UDP 10.20.30.40:5060;rport;branch=z9hG4bKj9Nm7v3047Ha
Max-Forwards: 70 FreeSWITCH-CN 出品
From: "Seven Du" <sip:+861234567890@example.com>;tag=B0U9ZQZQ124SK
To: <sip:+861234567890@example.com>
Call-ID: eb9f724e-9fed-1233-88ac-525400272cd0
CSeq: 77777777 INVITE

FreeSWITCH 文集

杜金房

版权所有，侵权必究

图书不在版编目 (NCIP) 数据

FreeSWITCH 文集 / 杜金房 著 / 2016.7

ISBN 7-DU-777777-7

本书收集了曾经发表在 FreeSWITCH 网站以微信公众号上的一些文章，可能有助于你了解 FreeSWITCH。

本书主要是针对非技术人员的，如果你是一名技术人员并想更深入了解 FreeSWITCH，请阅读《FreeSWITCH 权威指南》。

FreeSWITCH 文集

作 者 杜金房
封面设计 杜金房
校 对 高斐
排 版 杜金房
责任编辑 安茂茹

开 本 1890 毫米 × 2360 毫米
印 张 7.5
印 数 7
版 数 2016 年 7 月第 1 版 2016 年 9 月第 3 次发布
电子邮箱 freeswitch@dujinfang.com

前 言

自《FreeSWITCH 权威指南》出版以来，已经过去两年多了。在这两年多的时间里，我也收到很多的反馈，有批评的，有赞扬的，大部分还是赞扬的。

我曾经把我的书送给我的一些朋友。他们有的不是做技术这一行的，很多都会真诚地抑或谦虚地说看不懂，但又盛情难却，把书收下，搬（因为书很厚，有点重）回家，或束之高阁，或垫桌子腿，总之，没有发挥出书本来的作用。也着实费了我不少成本（因为书很厚，有点贵；我自己买自己的书也得花钱，有点囧）。

为了免去这些尴尬，对于这些朋友，我准备再出一本电子版的书，里面只放一些非技术，或者非常科普的内容。

说到电子版，其实还是有很多优势的。一是出版周期短，二是可以随时更新。有人喜欢纸质书，但越来越多的人也开始喜欢电子书了。方便携带，在移动设备上可随时阅读。

本书的大部分来自 FreeSWITCH 中文网站上以及 FreeSWITCH-CN 微信公众号上发表的文章。

如果你喜欢我或者喜欢 FreeSWITCH，你大概也会喜欢本书。

本书内容会不断更新，请关注本书的网站：<http://book.dujinfang.com>。你也可以订阅『FreeSWITCH-CN』微信公众号以随时了解 FreeSWITCH 和本书动态。另外，也非常欢迎订阅笔者个人的公众号『七歌』。



图 1: 『FreeSWITCH-CN』和『七歌』微信公众号

借口北京有雾霾，2016 年初，笔者从北京撤回烟台，在[北京信悦通](#)名下注册了烟台小樱桃网络科技有限公司 (<http://x-y-t.cn>)，继续提供 FreeSWITCH 商业技术支持服务。

目 录

前 言	1
1 FreeSWITCH 文集	7
1.1 FreeSWITCH 的前世今生	7
1.2 我与 FreeSWITCH 的故事	8
1.3 如何学习 FreeSWITCH?	13
1.4 FreeSWITCH 开源社区指南	14
1.5 关于 FreeSWITCH 常用术语翻译的意见	16
1.6 FreeSWITCH 背后的故事(译)	18
1.7 FreeSWITCH 与 Asterisk	20
1.8 FreeSWITCH 的历史	25
1.9 The missing Link	31
1.10 在飞机上上网打电话	33
1.11 You pay what I know	34
1.12 180 还是 183?	34
1.13 从 3·15 看电话号码透传	39
1.14 FreeSWITCH 帮我找到了工作	42
2 ClueCon	45
2.1 Cluecon 2011 小记	45
2.2 ClueCon 2012	47
2.3 ClueCon 2014	55

2.4	为什么会是我?	57
2.5	为什么又是我?	58
2.6	ClueCon 2015	60
2.7	ClueCon 2016	61
3	FreeSWITCH 那些事	72
3.1	FreeSWITCH 是什么?	72
3.2	FreeSWITCH 能做什么?	73
3.2.1	功能列表	73
3.2.2	Features	74
3.2.3	Protocols	77
3.3	FreeSWITCH 的版本	77
3.3.1	FreeSWITCH 1.0	78
3.3.2	FreeSWITCH 1.2	78
3.3.3	FreeSWITCH 1.4	78
3.3.4	FreeSWITCH 1.6	78
3.3.5	FreeSWITCH 1.8	78
4	八卦江湖	79
4.1	Git 仓库迁移	79
4.2	FreeSWITCH 问题解决的几个例子	81
4.3	FreeSWITCH 代码最新变化	82
4.4	FreeSWITCH 与 FFmpeg	84
4.5	如何录制喜欢的电视节目	86
4.6	我看首届胶东开发者大会	89
4.7	与跟 FreeSWITCH 不大相关的故事	90
4.8	盘点《FreeSWITCH 权威指南》	91

5 科普解惑	99
5.1 PSTN 起源与发展	99
5.2 VoLTE 给我们带来了什么?	103
5.3 模拟信号与数字信号	104
5.4 PCM	104
5.5 我国电话网结构	105
5.6 电话号码的知识	106
5.6.1 固定电话号码	106
5.6.2 移动电话号码	107
5.6.3 短号码	107
5.6.4 800 和 400 号码	108
5.6.5 北美电话号码分类计划	108
5.6.6 电话号码的书写格式	109
5.7 时分复用与局间中继	109
5.7.1 时分复用	109
5.7.2 局间中继	109
5.8 信令 (Signaling)	109
5.8.1 信令分类	110
5.8.2 用户线信令	110
5.8.3 局间信令	111
5.8.4 七号信令	111
5.8.5 H323 与 SIP 信令	114
5.9 媒体 (Media)	115
5.10 电路交换与分组交换	116
5.10.1 电路交换	116
5.10.2 分组交换	116
5.11 VoIP	117
5.12 IMS	118
5.12.1 IMS 的特点	118
5.13 生活中的信令和媒体	120

6 小樱桃	124
6.1 小樱桃是怎么来的?	124
6.2 如何获得 FreeSWITCH 培训优惠?	125
6.3 烟台小樱桃成功参加 2016 年南京企业通信与呼叫中心技术发展论坛	126
6.4 FreeSWITCH 走进校园之烟大行	128
6.5 FreeSWITCH 高手速成培训 2016 春季班 (南京站) 圆满结束	129
6.6 FreeSWITCH 开发者沙龙 2016	130
6.7 FreeSWITCH&WebRTC 高手速成培训 2016 秋季班 (北京站) 圆满完成	136
7 不知所言	138
7.1 1024, 我也是一名程序员	138
7.2 待价而估, 技术服务如何定价	141
7.3 挂号费和飞机票	145
7.4 我在 Mac 上的写作工具链	146
7.5 有求必应	150
7.6 钱	154
7.7 这事不归我管	156
8 精英	158
8.1 FreeSWITCH 精英群上线了	158
8.2 FreeSWITCH 精英群第一次活动小记	158
8.3 FreeSWITCH 精英群第 N 次活动小记	160
8.4 FreeSWITCH 第 N+1 次活动小记	163
9 旧文	166
9.1 圣诞快乐	166
9.2 我 Donate, 我快乐	167
9.3 说说 NAT	168
9.4 要有光	169
9.5 如何得到我的帮助	171
9.6 无题	173

9.7 说说开源软件	173
9.8 穿越	174
9.9 屁大点事	175
9.10 那些故事	176
9.11 FreeSWITCH-CN 高手速成培训 2014 春季班（上海站）圆满结束	181
9.12 锤子	182
9.13 写在 2014 年最后一天	184
9.14 致 FreeSWITCH-CN 公众号所有订阅者	185
9.15 FreeSWITCH oTo 阿里巴巴站回顾	187
9.16 一个真实的 FreeSWITCH 问题	190
写在最后	193
作者简介	194
版权声明	195
广告	196
关于广告的广告	196
FreeSWITCH 第五届开发者沙龙将于 2016 年 8 月 27 日在京举行	196
FreeSWITCH 培训 2016 夏季班（北京站）将于 2016 年 8 月 28-30 日在京举行	196
烟台小樱桃网络科技有限公司提供商业 FreeSWITCH 及 OpenSIPS 技术支持	196
烟台小樱桃网络科技有限公司是潮流网络（GrandStream）山东总代理	196
FreeSWITCH 相关图书	197

第一章 FreeSWITCH 文集

本章收集了一些 FreeSWITCH 相关的文章，要了解 FreeSWITCH，先从这里开始吧。

1.1 FreeSWITCH 的前世今生



图 1.1: FreeSWITCH

话说混沌初开，那时还没有 FreeSWITCH，人类有史以来最早的通信方式基本上就是靠『吼』。当然，我们的老祖很是聪明，他们在生产劳动和相互协作中发明了历史上公认的最早的通信工具——『烽火台』，那都是史前的事了。

说话间就到了 19 世纪，苏格兰人亚历山大 · 贝尔发明了电话。从此，人们可以在被窝里窃窃私语，再也不用『吼』了。电话技术一直在进步，同时，计算机技术也一直在蓬勃发展，PC 机进入了千家万户。怎么在 PC 上打电话呢？Asterisk 以开源软件的姿态横空出世，解决了在 PC 机上打电话的问题，掀开了 21 世纪的新篇章。

2002 年，美国有一个叫 Anthony Minassale 的小伙子开始基于 Asterisk 编写一些呼叫中心类的应用。说起来，其实跟我们现在这些用 FreeSWITCH 写呼叫中心的码农差不多。Anthony 很有天分，对 Asterisk 研究地深了也经常发现一些 Bug、提交点补丁什么的。后来，他发现他写的补丁越来越

多，可根本问题还是解决不了，老是死。他就跟社区里的人说，咱们写个 2.0 版吧，大家说好啊，然后，就没有然后了。

Anthony 则不然，他回去后，盯着它的 Emacs 编辑器看了一个小时，敲入几行代码，他想重新写一个 Asterisk。总得起个名吧，第一个名叫 Choir，他希望该软件能像教堂里的唱诗班那样和谐。可是，没有人喜欢那个名字。他后来就改成 Pandora，也没有人喜欢。直到最后他改成 FreeSWITCH，大家都说，嗯，这个名高大上。立马注册商标，就挽起袖子开始干了。上面说的是 2005 年的事。

另外，2005 年 Anthony 还干了一件大事，那就是他搞了个网友见面会，把一批搞通信的技术大牛们聚在一起 Happy 了一把，这个网友见面会就叫做 ClueCon，从那年起每年一届都在芝加哥举行。

最初，FreeSWITCH 没什么功能，但是在 2006 年初的时候能跨平台了，至少能在 Linux、Mac 和 Windows 上原生的运行。后来，FreeSWITCH 还真能打电话了，经过努力，他们也有了第一批小白鼠用户，有不怕死的小白鼠还把 FreeSWITCH 应用到了生产系统上。

2008 年 5 月，FreeSWITCH 发布了 1.0 凤凰版。为什么叫凤凰？因为 Anthony 觉得开发的过程太痛苦了，1.0 发布有涅槃重生的感觉。

由于稳定、性能好，短短的时间，FreeSWITCH 收获了大批的用户。一晃到了 2012 年，Anthony 在 ClueCon 上发布了 FreeSWITCH 1.2 版。1.2 版是 FreeSWITCH 第一个稳定的发行版¹。

世界瞬息万变，Google 发布了 WebRTC 标准并在 Chrome 浏览器里实现一下子改变了互联网的通信方式。自然，很多人都想让 FreeSWITCH 与 WebRTC 互通。在一些公司的赞助下，这一功能终于在 FreeSWITCH 里落地生根，2014 年春，FreeSWITCH 发布了 1.4 版，这是第一个支持 SIP over Websocket 和 WebRTC 的版本。

时间进入了 2015 年，FreeSWITCH 1.6，带来了全新的 FreeSWITCH 视频转码和视频会议功能，WebRTC 高清视频能够支持高达 4K 的分辨率。

2016 年 ClueCon，Anthony 演示了同时用四种不同的浏览器（Microsoft Edge、Safari on Mac、FireFox、Google Chrome）同时呼入同一个 FreeSWITCH 会议，既将发布的 FreeSWITCH 1.8 又会给我们带来什么惊喜呢？让我们拭目以待。

1.2 我与 FreeSWITCH 的故事

杜金房 /2015.12.10

故事还得从几年前说起。

很久以前，我在电信（后来叫网通）做过程控交换机的维护，机型主要是上海贝尔的 S1240 和华为的 CC08，也了解西门子 EWS5、朗讯 5ESS 等机型以及各种接入网设备等。由于喜欢写些小程序，所以在工作中也写过一些与交换机交互的程序，但大都是一些执行命令创建和删除用户，收集和分析日志等的操作，从来没有涉及呼叫控制方面的。

¹之前的版本也稳定但是最稳定的版本永远都是开发分支的，也就是说发行版的发行速度远跟不上开发版。

有一年去北京，一次偶然的机会认识了 Jonathan Palley。他正在做英语在线教学的项目，从他那里我了解到了 Asterisk。当时，Asterisk 都已经七八岁的样子了，我却是第一次听说，可见处于体制内的我是多么的坐井观天。

在运营商工作的好处就是能接触到各种程控交换机、大型的服务器以及各种 UNIX 和各种数据库，虽然我的工作主要是维护，但我还是利用工作和业余时间开发了许多程序，开发语言主要是 C 和 PHP，把跟我工作相关的交换机和 OA 等系统全部融合起来。我的很多理念和想法以及具体实现都是比较超前的，系统也是很好用的。但是，在支撑系统全市统一、全省统一、全国统一的大环境下，我写的东西很显然推广不开。而且，后来，公司内部越来越轻运维，重市场，搞起了全员营销。说白了就是你必须发展多少小灵通用户，办多少宽带等。我越来越不喜欢那种工作状态，就准备去北京投奔 Jonathan Palley。

在北京之前，我研究了一阵 Asterisk，在淘宝上买了一个单口的语音卡，接上我的办公电话，有人打进来就先放 IVR，然后通过逃生口接上话机，我再跟他们通话，感觉很神奇。我甚至配置了路由，可以在家里用软电话通过网络利用办公室的电话线往外打『免费』的电话。

后来 Jonathan 说 Asterisk 有问题，他准备换 FreeSWITCH，问我有什么意见。我对 FreeSWITCH 其实没有什么研究，便赶紧上 FreeSWITCH 的官方网站看了看，下载、编译都很顺利，简单测试了一下，有些功能用起来竟然比 Asterisk 还方便。我便跟 Jonathan 说：值得换。那时候，FreeSWITCH 还没有出 1.0 版。

后来到了北京，我的职位是系统管理员，主要是搞定教学平台的各种安装和部署。我很欣慰，像我这种自学成才的 Linux 菜鸟在整个团队中却成了 Linux 专家。而且，更让我喜欢的是，我可以完全使用 Linux 桌面工作了，以前的各种 Windows 上专用的软件都见鬼去了（后来有了钱，电脑换成了 Mac）。在工作中，我学会了 Ruby、Ruby on Rails、SVN 以及后来的 Erlang 等技术。我很勤奋，分内的分外的事情我都做，以至于后来整个系统从最底层到最上层我都懂。

FreeSWITCH 在我们的系统里是一个重要的部分，在使用和维护过程中，我也会发现一些系统的 Bug，并渐渐跟 FreeSWITCH 社区有了交流。

其实，FreeSWITCH 最初相关的开发都是由 Jonathan 负责的，公司也一直想招一个专门做 VoIP 开发的人员，但就是一直没招到。后来 Jonathan 公务越来越忙，我又什么都会，FreeSWITCH 相关的开发任务也落在了我身上。当然，我后来也忙不过来，就又招了一名系统管理员给我打下手。这样，我的工作重点就转到了研发，以及系统架构。

外企的工作环境还是很宽松的。虽然工作不少，但我勤奋啊，在北京只有我一个人，因此，除了公司以及团队自发组织的各种业余活动，不管是晚上还是周末，我都抱着电脑。当然，有一件比较悲催的事情是，我跟公司几个客服（其它他们销售和客服什么都干）人员住在一起，因此，系统一有问题，便总能找到我。所以其实我大部分的业余时间是为公司奉献了。当他们不找我的时候，我就自己优化系统。

我也跟 Jonathan 开玩笑说，我其实大部分时间都是在修他写的 Bug。

Jonathan 是个天才，他学东西非常快，想做的东西很快就实现了。但如上面所说，他没有时间做精细的调试和打磨，因此，我正好是一个很好的补充。所以，我最初的 FreeSWITCH 技术全都

是跟他学的，包括混进 FreeSWITCH 社区。

我们是属于使用 FreeSWITCH 比较早的一批用户。在我们基于 FreeSWITCH 的系统上线后的几天内，FreeSWITCH 发布了 1.0.0 凤凰版。

当然，在使用开源软件的同时我们也有贡献。最初主要是报 Bug 和写 Wiki，后来，就开始提交补丁了。那时候，FreeSWITCH 的开发人员都很专注，因此，Bug 修得很快，基本上都是我们今天报上去，第二天上班一看就修好了（美国人在我们睡觉的时候工作）。有一阵，我接连不断的提交新特性的建议和补丁，在打完最后一个补丁后 Anthony 跟我说，我可以休息一下了吗？我说，可以了。

我基于一个 G729 库写了 FreeSWITCH 中最初的 G729 转码模块，后来网上流传的很多版本都是基于我的版本修改的（头文件里还有我的名字呢）。不过，写这个模块其实是一个错误。我后来才了解到，原来 FreeSWITCH 内部早就实现了 G729 的转码，只是，由于专利原因，无法对外发布。对此专利细节我不了解，但是，据我所知，在世界上绝大多数国家，即使你自己实现了 G729 的编码，也要交专利费。由于我的错误，FreeSWITCH 关闭了 Wiki 上 G729 页面的修改权限，也禁止在邮件列表里讨论 G729 转码。后来，FreeSWITCH 官方在 Linux 上实现了 G729 的 Licence 机制，10 美元买一个编码器和一个解码器，FreeSWITCH 最终才有了 G729 的解决方案。当然，那都是后话了。

当然，我们（外企）是很重视专利和版权问题的。所以，后来，我写的模块仅用于试验，但没用于生产。用于生产的，是我写的另外一个模块，叫 mod_recpld，是 Record Payload 的缩写。实现想法是，不经过转码，而把 G729 的内容直接录下来，后面，再用其它程序将 G729 转成 mp3 文件。而我们有这个『其它程序』的 G729 许可证。其实这种录音方法后来在 FreeSWITCH 核心中也实现了。另外，我们后来也发现 G729 编码在互联网上的声音质量不如 iLBC，因此，在我们决定不再使用一个仅提供 G729 编码的 PSTN 落地提供商的线路后，我们也不再使用 G729 了。

通过写上面两个模块，我深入了解到了 FreeSWITCH 内部的机理。以后再改代码，就容易得多了。

除了在线教育平台外，我们的销售和客服团队也逐渐壮大，因此，需要一个呼叫中心系统了。最初的呼叫中心是用 Trixbox 搭的，Trixbox 是一个集成 CentOS 和 Asterisk 的一个发行套件，使用起来很简单。我们使用很老的只有 700 多 M 内存的奔腾机器跑这个应用，竟也能带十来个座席。当时，使用的是『淘』来的 Asterisk 兼容卡连接模拟外线。我们甚至还用了 OSLEC 做回声消除。后来，随着需求越来越多，我们将它换成了 FreeSWITCH，我们自己实现了 CRM，跟我们的系统紧密集成。多紧密呢？我们用 Erlang 实现了 IVR 功能，当有电话打进来时，它便播放欢迎词，同时，根据主叫号码，异步到美国的服务器上去查询这个是谁的客户，进而分配座席和弹屏。去美国的服务器查询比较费时，但是我们用 Erlang 很优雅地通过异步操作实现了该功能。

另外，我们也购买了一个 Global IP Sound 的库，实现了自己的 SIP 客户端。没想到，后来 Global IP Sound 变成了 Global IP Solutions，后来又卖给了 Google，Google 又把那些代码开源了。

再回到我们的教学平台上。由于我们在美国的老师都是 Work-At-Home（在家工作）类型的，他们分布在美国各地，因此，实际上我们的教学平台就是一个分布式的呼叫中心系统。有一套很复杂的算法，根据老师上课的数量和质量给他们发钱。我们的平台用到了 Ruby on Rails，Erlang，Lua 和 FreeSWITCH，每个技术都有它们最擅长的地方。

同时，我们也在不断的招聘老师。我们做了一个自动面试程序，面试者在 Web 页面上注册，根据提示，输入自己的电话号码，我们呼叫他们的电话，阅读我们指定的文字并录音。这一切都是自动的，电话跟页面也是同步的。通过几轮自动面试之后，我们才会有相关的人员人工联系他们进行更深入的面试。这些，也都是用 FreeSWITCH 做的。

当然，这么多『分布式』的老师要开会啊，因而，我们又用 FreeSWITCH 实现了会议系统。

还有一个有意思的事情。当时，由于国内的网络条件和对 VoIP 的限制，我们在国内是直接使用 PSTN 电话跟学生连线的。有些学生反应他们的手机处于漫游状态（如到北京上新东方培训班），接电话很贵。因此我们又实现了 FreeSWITCH 跟 Skype 和 Google Talk 的互通。这样，我们就可以在 FreeSWITCH 中打电话到学生的 Skype 或 Google Talk 客户端上。

当然，事情也不是永远一帆风顺，最初跟 Skype 和 GTalk 互通，FreeSWITCH 总是崩溃，我们报了一些 Bug，打了一些补丁才解决。这也决定了我们的基础架构——我们专门起了几个 FreeSWITCH 进程，用于跟 Skype 或 GTalk 连，即使其中一个崩溃了，我们也可以迅速切换到另一个（为此，我们也注册了很多 Skype 账号：））。

总之，正是由于不断的『折腾』，我们的 FreeSWITCH 技术才不断进步。

花开两朵，各表一枝。再倒回去说说 FreeSWITCH 社区的事情。

一来二去，我在 FreeSWITCH 社区里混得熟了，便萌生了一个做 FreeSWITCH 中文社区的想法。我对开源很热爱的，我曾在 Linux Focus 上翻译文章，翻译 SQLite 的文档等。中国人也都很厉害，几乎任何东西都有了中文版，有了中文社区，但当时，FreeSWITCH 还没有。

我很快注册了域名，FreeSWITCH-CN 开张了。

在任何圈子里，总会找到志同道合的人，我发现了大熊。我发现他几乎在同一时间建立了 FreeSWITCH QQ 群，我甚至不记得我是怎么发现他的了。大熊人非常好，把我升级成了 QQ 群的管理员。他从来都很低调，不知道的还以为我是 QQ 群的群主呢。

对于 QQ，我以前不大用。最早注册的 QQ 号都忘记了，后来工作后一个同事让给我一个 QQ 号我使用至今。我前些年折腾 Linux，而 Linux 上的 QQ 总是各种不好用。后来在外企大家都用 MSN 和 Skype，而我的工作平台也由 Linux 换成了 Mac。离开外企之后，没有人用 MSN 了，而 Mac 版的 QQ 也开始争气了（比 Windows 上老是弹出垃圾广告好多了），所以我 QQ 就用得多了。

我很快取得了 FreeSWITCH SVN Contrib 代码库的提交权限。Contrib 就是 FreeSWITCH 外围的一些代码库，FreeSWITCH 扇丝们可以把自己实现的一些好玩的东西提交上去，全世界共享。我提交了一些我写的代码，后来，发现我在代码中『发明』的方法被大家广泛使用。甚至，有一次，在 ClueCon 上，一个保加利亚的哥们用非常不流利的英文跟我说那几个脚本真是太好用了，帮了他大忙。

再后来，由于我在 FreeSWITCH 社区表现突出，提交的补丁也得到了社区的认可，就取得了 FreeSWITCH 核心代码的提交权限。而这时，FreeSWITCH 的代码库也早已从 SVN 迁移到了 Git。

我最初建立社区的想法很简单，中文网站只是一个网站，用 Google Groups 做邮件列表，进行讨论，用 QQ 群做即时的讨论（我甚至也想过 IRC）。关于邮件列表，英文社区里也有很多讨论，到底

邮件列表好还是一个 BBS 好？社区的管理人员认为邮件列表好，因为你可以用任何你喜欢的邮件客户端参与讨论。而有不服气的人去建了 BBS，结果就是没有人气。我受他们的影响很大，因此，从来没有建 BBS 的想法。但很遗憾，Google Groups 很快就无法访问了。我好像是受了诅咒一样，我注册了 Twitter，Twitter 不好用了，我注册了 Facebook，用了几天不好用了，我用 DropBox，隔几天就连不上了，我建立了 FreeSWITCH-CN Google Groups 中文讨论区，也无法访问了。到今天，我偶尔还会在 Google Groups 里发邮件，但是，无论如何都没大有人气。

而关于建立 BBS，在国内也是很困难的。你必须备案，必须有人监管，必须……。总之，有很多人提出我们该有个 BBS，但我总是没有精力去做这件事，我也说过有人愿意捐赠服务器并搞定一切事情的话，我们也可以建立个 BBS，但到今天还是没有结果²。

我想，也许，大家有问题的话还是到专业的社区上去讨论吧，因此，我开了知乎专栏（zhuanlan.zhihu.com/freeswitch），并鼓励大家到那里去提问 FreeSWITCH 有关的问题。

QQ 群里的成员倒不少，目前也是最主要的交流平台了，群里风气很好，也没大有广告，我很高兴看到大家都在讨论技术。

最初，我在群里回答问题是很积极的，但是大家好像不怎么配合，总问些没法回答的问题。我便写了一系列的文章告诉大家应该怎么把问题描述清楚，也就是如何问到点子上，如何收集日志并贴到 Pastebin 上。后来，的确起到了一些效果，有些人已经学会了如何先把日志贴到 Pastebin 上再提问题了。

不可避免的，你需要一遍一遍的回答各种问题。因此，我想写一本书，给大家讲一些基础的知识。我的想法就是，再有人提问时，告诉他看书的第几页。但写一本书谈何容易，因此，最初我把写的东西以博客的形式发布出来，供大家免费阅读和挑刺。但我很讨厌把我的文章转载到自己的博客里去掉作者姓名当自己原创的人，也很讨厌那些把文档制成 Word 版上传到各种文档分享网站换积分的人。为此，我还专门跟百度干了一架。但得到的后果是在百度上搜 FreeSWITCH，再也到不了我们的网站了。

因此，后来，我的写作就不积极了，有时候写了也不愿意发。两年前，在我们国内 FreeSWITCH-CN 第一届沙龙前夕，我把书印了出来，大家竟然非常喜欢。也有读者告诉我，越看越爱看，每看都有收获，甚至在草原上放羊的时候也看。

但无论如何，小册子总有『非法出版物』之嫌。因此，能够出版，一直是我的一个夙愿。值得欣慰的是，经过四年多的积累，在华章公司的支持下，这本书《FreeSWITCH 权威指南》终于跟读者见面了。这是我的第一本书，也是国内第一本讲 FreeSWITCH 的书，但就在第一本书上出现了『权威』二字，我还是有点心虚的。我确实想把我所知道的都写进书里，但是，即使每个主题都没有深入进去，这本书也已经很厚了。无论取舍是否得当，总之，这是我真心想送给广大 FreeSWITCH 爱好者的一个礼物。同时，也欢迎各种批评。

2011 年，我第一次参加在美国芝加哥举办的 ClueCon，我不想错过任何机会，便找了个空子上台讲了几句。第一次用不流利的英文在 200 多个纯老外面前演讲，真的很紧张，不过，坚持下来了，反应还不错。

²后来，我们真的建立了一个 BBS，地址是：<http://bbs.freeswitch.org.cn>。

2012 年，我就有备上台了，英文也稍微好了一点点。我用了几个性感美女图片轰动全场。

2013 年，FreeSWITCH 三剑客 (Anthony、Brian、Mike) 都在下面认真听我的演讲（以前他们基本都在外面跟别人聊天），让我在台上小激动了一把。

当年，还有一个好玩的事情。一个加拿大的哥们跟我说，我写得东西真好，他有问题了就到我的网站上找答案，看不懂中文就用 Google Translate 翻译。还把他做的系统演示给我看（他确实做得很好）。

2014 年的 ClueCon 马上就要到了，我还会去。

2012 年开始，我们 FreeSWITCH-CN 中文社区也学着 ClueCon 的样子在国内搞活动，为避免版权问题，我们不叫 ClueCon，叫 FreeSWITCH 开发者沙龙。随着大家对 FreeSWITCH 的关注的提高，我们的活动竟也有模有样。

2013 年开始，我开始做公开的 FreeSWITCH 培训班，每批都有二、三十名学员，感觉也还不错。

花开两朵，再表一枝。大约就在我第一次参加 ClueCon 的时候，我们原先的公司因为各种原因，部分解散了，其中也包括我。

在老的没人要的时候，只能去创业了。

创业的过程是艰苦的，我们也学到了很多东西。在这些年里，我潜心研究 FreeSWITCH，实现了 MSRP、实现了类似 IMS 的架构、实现了 RTMP 基于 Flash 的视频（有大熊的支持）、实现了 RTSP、VLC 视频、实现了视频转发、视频转码、视频会议、视频录像、做了 N 个版本的指挥调度系统。这些系统有的是实验性地，有的还不很完善，也大部分没有开源，不过，我一直惦记着开源这回事呢。

关于做开源与赚钱的问题，我也一直在思考。但我想，我会一直做 FreeSWITCH 的培训与咨询，支持大家在 FreeSWITCH 上面做呼叫中心、做运营、做指挥调度等各种应用，而不是去做上层五花八门的应用，应该算是一种思路吧？

与诸君共勉。

1.3 如何学习 FreeSWITCH?

Seven/2016.08

好多同学问，如何开始学习 FreeSWITCH？在此，我就跟大家说道说道。希望不论是新手还是老手，从这里都能得到你想要的……

首先，我们官方网站上有很多资源。欢迎加入我们的社区，大家共同学习。

最好的中文学习资料就是《FreeSWITCH 权威指南》，每个学习 FreeSWITCH 的都应该好好读一读。除此之外，还有一些 FreeSWITCH 电子书，或许也值得一看 (<http://book.dujinfang.com>)。

如果有什么问题，你可以到 BBS 上讨论。

另外，我们有一个 FreeSWITCH-CN1 的 QQ 群，这个群目前小一点，只有几百人，这个是免费的（QQ 群号：487681577），欢迎大家加入交流；

我们还有一个 FreeSWITCH-CN 的 QQ 群，这里面有将近 2000 人，讨论问题也很多，剩余名额有限，属于稀缺资源，入群需要付费（200 元/位）。

我们还有一个 FreeSWITCH 精英群（QQ 群），群里不定期的会有一些技术交流活动，为了保证群的质量，这个也是需要付费的，目前优惠期已过，1024 元/年。

我们还有一个微信群（FreeSWITCH 群），名额有限，付费加入，200 元/人。

当然，也是最主要的是我们还有一个 FreeSWITCH 中文社区微信公众号（FreeSWITCH-CN）。

我们还不定期的有 FreeSWITCH 培训班³，如果你正好赶上，可以来参加我们的培训，最近的一次将在 8 月底，等着你来噢！如果赶不上，我们也提供“到公司里培训”的服务，请联系我们（邮件：info@x-y-t.cn）。

如果你小有所成，我们有一个 FreeSWITCH A 计划，如果你是自由职业者，或者有时间兼职做一些事情，可以考虑加入我们。

如果你们公司基于 FreeSWITCH 做一些产品服务，我们也提供完善的商业咨询服务，请给我们发邮件：info@x-y-t.cn。

当然，你也可以申请加入烟台小樱桃网络科技有限公司，我们是专门做 FreeSWITCH 的公司⁴。

我们有每年一度的 FreeSWITCH 开发者沙龙⁵，每个热爱 FreeSWITCH 的人都至少应该参加一次。你可以来跟大家学习交流，也可以来会上演讲，分享你的经验心得，以及你的创意。你也可以来赞助我们的活动，展示你的产品，让更多的人了解，喜欢你的产品。

ClueCon⁶是 FreeSWITCH 全球的年度盛会，每年一次在芝加哥举行，如果有条件的话，每个人都应该至少参加一次。

1.4 FreeSWITCH 开源社区指南

FreeSWITCH 是一个开源项目，也是一个开发者社区。我们欢迎所有人加入 FreeSWITCH 社区，一起学习、讨论、并贡献自己的力量，使我们的社区和我们的项目越办越好。

中文社区

FreeSWITCH 中文社区的名称是 FreeSWITCH-CN，网址是<http://www.freeswitch.org.cn>，社区的最新消息都会在这里发布。

我们有一个 BBS (<http://bbs.freeswitch.org.cn>)，一个 QQ 群（190435825）一个微信公众平台（FreeSWITCH-CN）。我们每年会不定期的举办中国区的 FreeSWITCH 沙龙线下活动。

³FreeSWITCH 培训：<http://x-y-t.cn/training.html>。

⁴小樱桃：<http://x-y-t.cn/jobs.html>

⁵FreeSWITCH 开发者沙龙：<http://www.freeswitch.org.cn/fscnfs/index.html>。

⁶ClueCon：<https://www.cluecon.com>。

鼓励大家在 BBS 上讨论。因为在 BBS 上提问和回答问题都是异步的，也就是说，想给你答案的人可以在任何时候回答，而不用像 QQ 群中那样你提问的问题被其它人冲走了。

另外，我们也建议大家到知乎 (<http://www.zhihu.com>) 上搜索『FreeSWITCH』相关话题并提问，也可以邀请社区里比较爱助人为乐的人回答。知乎是一个专业的问答社区，相信大家都能提出专业的问题，也能得到专业的解答。

在社区中（包括邮件列表、知乎和 QQ 群等）提问时，我们有以下建议：

- 我们是一个公共社区，社区成员来自不同的地方，有着不同的背景，对于同一个问题有着不同的理解。
- 礼貌回答，即使在别人不礼貌的时候也要保持冷静。我们是一个社区、不是战场。
- 还是礼貌，如果别人回答了你的问题，最好说声谢谢。如果别人的回答帮你解决了问题，也最好给一个反馈，让别人知道你的问题已经解决了。
- 如果在 QQ 等即时工具上，不用问『有人吗？』，或者『有人熟悉 XX 吗？』之类的问题，直接提问你自己的问题即可。
- 提问要问到点子上，要言之有物。不要提那些『我装上了 FreeSWITCH，怎么不好用啊』之类的别人无法回答的问题。
- 一般来说，提问时说明问题的现象，你使用的操作系统、版本号、FreeSWITCH 的版本号，贴上必要的日志等，有助于别人帮你快速定位并解决问题。
- 尽量不要使用截屏，有时候截屏很难辨认。
- 不要灌水。一个问题问多好遍也并不一定得到有效的回答，反而会影响大家对你的印象。一个例外是，如果你在 QQ 群中，你可以在一天中的不同时段或不同的日期重复提问，因为这时可能有不同的人上线。如果还没有人回答，而你又想知道答案，可以尝试将这些问题转到邮件列表和知乎上，得到回答的几率大一些。
- 大段的内容（如系统日志等）要贴到 pastebin 或 gist⁷上。
- 大家都是志愿者，都愿意为社区做贡献，但没有义务回答你的问题。所以，在问题得不到回答时，也不要耍小脾气。
- 要看新手指南、橡皮鸭子问题以及提问的智慧。多用 Google——其它的搜索引擎很难找到你想要的答案。
- 多做贡献。多为社区做贡献，就会越受到大家的爱戴，同时别人也越愿意帮助你。
- 不要贸然要求别人私下回答你的问题。有问题在公共邮件列表或 QQ 群中讨论，你得到回答的机会比向私人提问得到的机会更多，因为有那样的话会有更多的人看到你的问题。同时，别人也更乐意在公共场所回答你的问题，因为那样可以帮助更多与你遇到同样问题的人。
- 把以上这些再看一遍。

英文社区

英文社区的资源比中文社区的多。如果你想在一个国际化的环境中成长的话，建议从现在就开始

⁷参见<http://pastebin.freeswitch.org> 及<http://gist.github.com>。

做。如果英文表达不是很流利，那么也可以从现在开始就『潜水』。下面，简单列举一下英文社区中的一些资源。

- <http://www.freeswitch.org> 是 FreeSWITCH 的官方网站。
- <http://confluence.freeswitch.org> 是英文的 Wiki 网站，有无数的资料。
- <http://lists.freeswitch.org> 是一个邮件列表。大部分用户可以从 freeswitch-users 邮件列表开始。
- <http://freenode.net> 是一个 IRC 站点，其中有一个『#freeswitch』频道，大家都在那里交流，类似于中国的 QQ 群，但这个交流工具比 QQ 历史悠久得多。
- <http://jira.freeswitch.org> 是一个缺陷跟踪系统，你可以在那里汇报 Bug，或者提新需求。当然，如果希望你的新需求快速得到响应，也可以提一个 Bounty。一个 Bounty 的意思是说你希望实现一个新需求并愿意为之付费。
- <http://files.freeswitch.org> 上有 FreeSWITCH 各种资源的下载，包括源代码和已编译的程序等。
- <http://cluecon.com> 是 ClueCon 的主站。ClueCon 是一年一度的开发者技术交流会，每年 8 月份都在美国芝加哥举行。
- <https://freeswitch.org/confluence/display/FREESWITCH/ClueCon+Weekly+Conference+call> 列出了 FreeSWITCH 电话会议的参与方式。电话会议是 FreeSWITCH 社区成员使用语音交流的一种方式。基本上是每周三 UTC 时间下午 6 点开始，每周五不定时也有 Friday Free For All 的会议。支持使用 WebRTC、Flash、SIP、H323 以及 PSTN 等多种方式参加会议。该页面上也有历史会议的录音。

其它

总之，只要你学习或者使用 FreeSWITCH 就是为 FreeSWITCH 做贡献。不要不好意思说，有问题也不要不好意思问。众人捧柴火焰高嘛。

1.5 关于 FreeSWITCH 常用术语翻译的意见

本文所指翻译主要指英译汉。

众所周知，翻译的最高境界是『信、达、雅』，通俗来讲就是『正确、易懂、得体』，它实际上是从『内容、表达、风格』三方面来要求译作。但拘泥于以上标准往往受到很多局限。尤其对于科技作品来说，用上述观点来评价翻译作品则更难保证评价的正确性和客观性。

按照信息论的观点，翻译的本质就是通过语种转换把一种语言的言语所承载的信息转移到另一种语言的言语当中，用该种语言表达出来以传递给目标语言读者的过程，是把一种语言的言语转换成另一种语言的言语的活动。由此看来，译文是信息从原著传递到目标语言读者的中间载体。因此，翻译包括原著与目标语言读者两个对象。那么，我们在翻译时，除了应该忠实于原文外，更应该考虑的是目标读者。比方说，某件事情或动作，对于原著语言的读者来说可能是习以为常的，但对于目标语言

的读者来说可能很陌生或根本不知道是怎么回事。这时，我们就应该允许译者加入相关背景知识的脚注，或者直接用目标语言读者所熟悉的、类似的表达方式来表达。当然，我们这里的目的不是为了讨论翻译，在此，我仅就 FreeSWITCH 资料的翻译来讲一下笔者的观点：

FreeSWITCH 资料大部分来自其官方 Wiki 或者邮件列表，是来自全球众多网友的贡献。其作者大部分是程序员或技术人员，因此他们大都不太注重语法和修辞，并且这种网络媒体写起来本身就比较自由，再加上还有好多俚语和网络语言，有不少文章也是出自非英语母语的人之手，就更难保证语法的正确性了，因而翻译起来就非常困难。在此，我主张，遇到某些难以翻译的句子时，尽量把长句拆成短句，在充分了解原作的基础上甚至可以用自己的语言表达 (Anthony 最喜欢写长句了)。因为我们最重要的任务是读中文的人能理解它，而不是一味地『忠实』原文。

关于英文中的术语，可译可不译。如：没有人会把『HTTP』译成『超文本传输协议』，也没有人把 XML 译成『可扩展标记语言』。为什么？因为它们应该是众所周知的，至少在技术领域是应该是这样的。但对于某些特定领域的术语，有一些甚至在中文中没有对应的词，怎么办呢？最简单的答案是不译，这些即能最大程度地忠实原文，又不会让读者不知所云。当然，如果你的水平很高，你创造一个新词也可能成为国际标准，那就另当别论了。

对于某些缩写，最好能标注原文。有些书会在附录里列一个术语对照表，但我认为大多数情况下还是直接注在正文(用括号或脚注)中较好，而且只需在第一次出现时加注。如 PSTN(Public Switched Telephone Network，公共电话交换网)、ENUM(E.164 NUmber Mapping，E164 号码映射)。当然，对于非专业的读者来说，他们可能还是不知道是什么，如果你面向的是这样的读者，那最好在脚注里(或超文本里)加个指向维基百科的链接(如果有人问『维基百科是什么』该怎么办？)。举个例子，大部分人都不知道自己家电话所在的网络叫『公众电话交换网』，更不用说 PSTN 了。那么，在某些场合说到它与 VoIP 的区别时，你也可以称之为『普通电话网』或『传统电话网』。

关于标点。标点在文章中也是很重要的。对于括号，可以用英文的也可以用中文的，但必须统一。引号如果只表示一个名字时，在大多数情况下都可以不用。如上段中的『普通电话网』，如果不加引号，『你也可以称为普通电话网或传统电话网』一句是没有歧义的，作者加了引号是为了强调。当然，在本段中，你必须加引号，因为本段中是引用。

当然，如果你的译作不是发在正式的出版物中，而只是在论坛里灌水或贴在自己的 Blog 上，那就随便怎么写都可以了。

以下是作者对 FreeSWITCH 相关术语的译法，不当之处请广大读者批评指正。

- **Dialplan**: 拨号计划。虽然在 Asterisk 的书中译为『拨号方案』，但笔者还是认为『拨号计划』较好。
- **Endpoint**: 终点。专有名词，可不译。
- **ASR/TTS**: 自动语音识别/文本语音转换 (语音合成)。
- **Directory**: FreeSWITCH 中的用户目录，或者通用的目录服务 (如 LDAP)。
- **Events**: 事件。
- **Event handlers**: 事件句柄。
- **Event Socket**: 事件套接字，可不译。

- **Formats**: 格式。
- **Loggers**: 日志。
- **Languages**: 语言。
- **Say**: 说。可不译。
- **Phrase**: 短语。可不译。
- **Timers**: 定时器/计时器。
- **Applications**: 应用。
- **FSAPI**: FreeSWITCH 应用程序接口，也可译为 FreeSWITCH 命令接口。
- **CDR**: 呼叫详单/呼叫详细记录。
- **PSTN**: 公众电话交换网。
- **SIP**: 会话初始协议。
- **SIP UA**: SIP 用户代理。
- **UAC/UAS**: 用户代理客户端/服务器端。
- **Sofia Profile**: Sofia 配置文件，可不译。
- **VoIP**: 不译。

1.6 FreeSWITCH 背后的故事(译)

Anthony Minessale/文Seven Du/译

本文由 Anthony Minessale 写于 2007 年 5 月。来自[www.freeswitch.org⁸](http://www.freeswitch.org)。翻译它是因为我觉得永远都不会过时… — 译者注

我开发 FreeSWITCH 已经近两年的时间了。在我们第一个发行版即将发布的黎明之际，我想花一点时间来与大家分享一下软件背后的故事，并透露一点即将到来的消息。本故事也将会登在[OST Magazine](#)第一期上。呵，去下载一份吧，免费的！

写 FreeSWITCH 的想法诞生于 2005 年春的一次 Asterisk 开发者大会上。当时，我为 Asterisk 1.2 版贡献了大量的代码和新特性，并为其以后的发展提了好多思路。我们都认为在当时的 Asterisk 代码树中存在很严重的限制，并且面临着一些问题 修正一些问题时不仅会牺牲很多特性，而且还会花大量的时间。一种思路是建立新的代码分支，通过将大家已经熟悉的代码分开，新的分支就不会影响 Asterisk 用户的使用，从而能减轻好多开发的压力。问题是开发者都不希望将精力分散到同一份代码的两个版本上，因为那样他们就不得不将 BUG 修正和代码修改在两个分支间拷来拷去。我的建议是：在一个单独的代码库上从头创建一个 Asterisk 2.0 的分支，等一切就绪后再对外发布。我想那个想法确实打动了一些开发人员，但现实是，由于讨论的时间过长，最终大家都失去了兴趣。不过，我没有。

很明显，我是唯一一个认真考虑这一问题的。接下来我用了几天的时间在做一个白日梦——如何设计一个新的电话系统。之后，我再也无法抑制，便创建了一个新的目录开始从头写 choir.c。是的，Choir 是我为该工程选的第一个名字。我希望不同的通信部件能够步调一致地协同工作，就像教堂的

⁸<http://www.freeswitch.org/node/60>

唱诗班那样优雅和谐 (perfect harmony)。我又用了 5 天时间把我想到的点子都组织到几个文件里。最初的努力并不能装配成一个电话交换机。我知道，我需要创建一个稳定的核心，并应该能跨平台。所以最简单的是使用 Apache APR 库来搭建一个基本的子系统，让它能够动态装载共享模块并悠闲地停在屏幕上等待 shutdown 命令。实现这些代码用了另外 6 个月的时间。

在那 5 天里，我知道了写一个达到那种要求的可伸缩性程序并不是一件简单的事情。我最初的目标是捡起那些 Asterisk 丢掉的东西并加以改善。但随着思考的深入，我越来越觉得，实际上我想改进的是最基础的设计和功能。最终我得出结论，Asterisk 不能实现一些我期望的功能是因为它不是我所需要的软件。也就在那时，我知道我不是要编写另外一个 PBX，而是要开发一个完全不同级别的应用程序。我用了后续的几个月时间组织了第一届 ClueCon 年会来讨论如何合理的设计 Choir。我希望在继续写任何代码之前能确信我有正确的计划。从这一点上说 I 仍在做相当的一部分 Asterisk 开发，并且我也在我写的一些第三方模块中来实验我的想法。另外，我也让我的同事们花了相当多的时间来争论在电话领域里如何『正确』地做事情，这也算是一种前瞻性吧。

在那年的 ClueCon 会议上，我有幸遇到一些在电话领域里很有影响力的开发者，并把很多灵感带回了家。同时，Asterisk 阵营中的关系开始有些紧张，因为有几个开发者也打算建立新的分支。新的项目称为 OpenPBX (现在叫 Call Weaver)，从某种意义上讲它引起了 Asterisk 社区的一次严重地分裂。最初，OpenPBX 把我的许多第三方 Asterisk 模块加入到他们的新代码中，并就如何增强稳定性方面咨询我的建议。其中有一点，甚至他们还来看我是如何在我的新项目中实现的。可能我们能再打起精神来重写那些老的计划，但最终没有实现。不过，我想，最意义深远的一刻是 当有人问我：『多长时间以后它才能打电话呢？』我不知道，所以我决定搞清楚。简短的回答是一星期。

与我想达到的目标相比，能打电话只是一个小小的胜利。我还有很多要做的事情。这一点不起眼的业绩得不到太多关注。好消息是，这一项目最终上路了。我深居简出，用了三个月时间试图能做出点能吸引公众眼球的东西。那段时间，项目的名字曾改为 Pandora 并最终成为 FreeSWITCH。2006 年 1 月我们公开的 SVN 仓库和一个邮件列表上线了。那时仅有几个模块和一个很短的特性列表。但我们有了一个可以工作的内核，它能够在包括 Mac OS X, Linux 和 BSD 的几个 UNIX 变种上编译和运行，并能在 Microsoft Windows 上以一个控制台程序运行。

随着时间的推移，我们也越加有动力。有时也停下来犯几个错误，然后继续前进，写几个新的模块。在试验了四个不同的 SIP 终点模块后，我们决定使用 Sofia SIP。也曾试验过 5 个不同的 RTP 协议栈，最终决定还是我们自己写。同时我也开发了 mod_dingaling 来做与 Google Talk 的接口，以及一个多特性的会议桥。在第一年里，我主要关注如何在使核心尽量稳定的基础上，提供几个外部接口以便于其他模块的开发。如用于 IVR 的嵌入式的 Javascript，一个 XML-RPC 接口和一个用于远程控制和事件监控的基于 TCP 的 Socket 的接口。

第二年的 ClueCon 大会，那一天仿佛就是我白日梦醒来的日子。我第一次向我的同行们演示了 FreeSWITCH。近九个月后，在距离第一个想法两年之际，第三届 ClueCon 一个月前，我们达到了开发的 BETA 阶段 FreeSWITCH 1.0 发布在际了。我们也吸引了一些勇敢的开发者一道。他们已经把 FreeSWITCH 用于生产系统，并给我们提供了保证我们第一个发布版本成功的很重要的反馈。

在发布之前，最后一点工作是我新写的一个叫做 OpenZAP 的开源的 TDM 抽象库。使用 BSD 协议的'mod_openzap'模块将取代当时特定于 Sangoma 的'mod_wanpipe'，并提供 Sangoma 及其他

几种 TDM 硬件（只需要开发对应的模块）支持。OpenZAP 也将为模拟和 ISDN 信令提供一个简单的接口。OpenZAP 的指导思想是 应用程序能使用同样的 API 去控制任何它所支持的 TDM 硬件。它提供一种方式能将所有不同的特性规范化。如果一种卡缺少某种特性，那么它就能以软件的形式实现，不管是在 OpenZAP 库中还是在与生产商硬件 API 通信的接口程序中。

我们在如此短的时间内做这么多事听起来好像是不可能的，但我想，最终，还是像格言里说的：『需要是发明之母。』接下来的路还很长。但我想就此机会感谢所有曾经帮助我们走了这么远的人。下面列表中是所有在我们 AUTHORS 文件中的人：

- Anthony Minessale II (就是我!)
- Michael Jerris (我们极具价值的编译专家和跨平台专家 cross-platformologist, [呵，这个词是我创造的])
- Brian K. West (我们挚爱的 Mac 权威，没有他的帮助，我们将寸步难行)
- Joshua Colp (帮助我们做了第一个 SIP 模块，虽然现在我们已经不用了)
- Michal “cypromis” Bielicki (他从第一天就加入进来了，感谢信任！)
- James Martelletti (把 mono 集成进了 FreeSWITCH.)
- Johny Kadarisman (帮我们弄好了 python 模块)
- Yossi Neiman (写了'mod_cdr'收集通话详单)
- Stefan Knoblich (在我们的 SIP 之旅上帮助甚多)
- Justin Unger (找到很多 BUG)
- Paul D. Tinsley (SIP presence 以及其他好的建议)
- Ken Rice (为什么有这个名字？它给了我们做了很多测试和补丁)
- Neal Horman (在会议模块上有巨大贡献)
- Michael Murdock (我们 CopperCom 的朋友，有大量反馈和补丁)
- Matt Klein (大量 SIP 帮助，将帮我们确保 FreeSWITCH 运行于 FreeBSD.)
- Justin Cassidy (幕后工作者，确保一切正常)
- Bret McDanel (敢吃螃蟹的人，试验了绝大多数的功能，最早发现了很多隐藏的 BUG，我指的是复活节彩蛋！)

1.7 FreeSWITCH 与 Asterisk

Anthony Minessale/文 Seven Du/译

VoIP 通信，与传统的电话技术相比，不仅仅在于绝对的资费优势，更重要的是很容易地通过开发相应的软件，使其与企业的业务逻辑紧密集成。Asterisk 作为开源 VoIP 软件的代表，以其强大的功能及相对低廉的建设成本，受到了全世界开发者的青睐。而 FreeSWITCH 作为 VoIP 领域的新秀，在性能、稳定性及可伸缩性等方面则更胜一筹。本文原文在<http://www.freeswitch.org/node/117>，发表于 2008 年 4 月，相对日新月异的技术来讲，似乎有点过时。但本文作为 FreeSWITCH 背后的故事，仍很有翻译的必要。因此，本人不揣鄙陋，希望与大家共读此文，请不吝批评指正。— 译者注

FreeSWITCH 与 Asterisk 两者有何不同？为什么又重新开发一个新的应用程序呢？最近，我听到很多这样的疑问。为此，我想对所有在该问题上有疑问的电话专家和爱好者们解释一下。我曾有大约三年的时间用在开发 Asterisk 上，并最终成为了 FreeSWITCH 的作者。因此，我对两者都有相当丰富的经验。首先，我想先讲一点历史以及我在 Asterisk 上的经验；然后，再来解释我开发 FreeSWITCH 的动机以及我是如何以另一种方式实现的。

我从 2003 年开始接触 Asterisk，当时它还不到 1.0 版。那时对我来讲，VoIP 还是很新的东西。我下载并安装了它，几分钟后，从插在我电脑后面的电话机里传出了电话拨号音，这令我非常兴奋。接下来，我花了几分钟的时间研究拨号计划，绞尽脑汁的想能否能在连接到我的 Linux PC 上的电话上实现一些好玩的东西。由于做过许多 Web 开发，因此我积累了好多新鲜的点子，比如说根据来电显示号码与客户电话号码的对应关系来猜想他们为什么事情打电话等。我也想根据模式匹配来做我的拨号计划，并着手编写我的第一个模块。最初，我做的第一个模块是 `app_perl`，现在叫做 `res_perl`，当时曾用它在 Asterisk 中嵌入了一个 Perl5 的解释器。现在我已经把它从我的系统中去掉了。

后来我开始开发一个 Asterisk 驱动的系统架构，用于管理我们的呼入电话队列。我用 `app_queue` 和现在叫做 AMI（大写字母总是看起来比较酷）的管理接口开发了一个原型。它确实非常强大。你可以从一个 T1 线路的 PSTN 号码呼入，并进入一个呼叫队列，坐席代表也呼入该队列，从而可以对客户进行服务。非常酷！我一边想一边看着我的可爱的 Web 页显示着所有的队列以及他们的登录情况。并且它还能周期性的自动刷新。令人奇怪的是，有一次我浏览器一角上的小图标在过了好长时间后仍在旋转。那是我第一次听说一个词，一个令我永远无法忘记的词——死锁。

那是第一次，但决不是最后一次。那一天，我几乎学到了所有关于 GNU 调试器的东西，而那只是许多问题的开始。队列程序的死锁，管理器的死锁。控制台的死锁开始还比较少，后来却成了一个永无休止的过程。现在，我非常熟悉『段错误（Segmentation Fault）』这个词，它真是一个计算机开发者的玩笑。经过一年的辛勤排错，我发现我已出乎意料的非常精通 C 语言并且有绝地战士般的调试技巧。我有了一个分布于七台服务器、运行于 DS3 TDM 信道的服务平台。与此同时，我也为这一项目贡献了大量的代码，其中有好多是我具有明确版权的完整文件⁹。

到了 2005 年，我已经俨然成了非常有名的 Asterisk 开发者。他们甚至在 CREDITS 文件以及《Asterisk，电话未来之路》这本书中感谢我。在 Asterisk 代码树中我不仅有大量的程序，而且还有一些他们不需要或者不想要的代码，我把它们收集到了我的网站上(至今仍在 <http://www.freeswitch.org/node/50>)。

Asterisk 使用模块化的设计方式。一个中央核心调入称为模块的共享目标文件以扩展功能。模块用于实现特定的协议（如 SIP）、程序（如个性化的 IVR）和其他外部接口（如管理接口）等。Asterisk 的核心是多线程的，但它非常保守。仅仅用于初始化的信道以及执行一个程序的信道才有线程。任何呼叫的 B 端都与 A 端都处于同一线程。当某些事件发生时（如一次转移呼叫必须首先转移到一个称作伪信道的线程模式），该操作把一个信道所有内部数据从一个动态内存对象中分离出来，放入另一个信道中。它的实现在代码注释中被注明是『肮脏的』¹⁰。反向操作也是如此，当销毁一个

⁹ <http://www.cluecon.com/anthm.html>

¹⁰ /* XXX This is a seriously wacked out operation. We're essentially putting the guts of the clone channel into the original channel. Start by killing off the original channel's backend. I'm not sure we're going to keep this function, because while the features are nice, the cost is very high in terms of pure nastiness. XXX */

信道时，需要先克隆一个新信道，才能挂断原信道。同时也需要修改 CDR 的结构以避免将它视为一个新的呼叫。因此，对于一个呼叫，在呼叫转移时经常会看到 3 或 4 个信道同时存在。

这种操作成了从另一个线程中取出一个信道事实上的方法，同时它也正是开发者许许多多头痛的源头。这种不确定的线程模式是我决定着手重写这一应用程序的原因之一。

Asterisk 使用线性链表管理活动的信道。链表通过一种结构体将一系列动态内存串在一起，这种结构体本身就是链表中的一个成员，并有一个指针指向它自己，以使它能链接无限的对象并能随时访问它们。这确实是一项非常有用的编程技术，但是，在多线程应用中它非常难于管理。在线程中必须使用一个信号量（互斥体，一种类似交通灯的东西）来确保在同一时刻只有一个线程可以对链表进行写操作，否则当一个线程遍历链表时，另一个线程可能会将元素移出。甚至还有比这更严重的问题当一个线程正在销毁或监听一个信道的同时，若有另外一个线程访问该链表时，会出现『段错误』。『段错误』在程序里是一种非常严重的错误，它会造成进程立即终止，这就意味着在绝大多数情况下会中断所有通话。我们所有人都看到过『防止初始死锁』¹¹这样一个不太为人所知的信息，它试图锁定一个信道，在 10 次不成功之后，就会继续往下执行。

管理接口（或 AMI）有一个概念，它将用于连接客户端的套接字(socket)传给程序，从而使你的模块可以直接访问它。或者说，更重要的是你可以写入任何你想写入的东西，只要你所写入的东西符合 Manager Events 所规定的格式（协议）。但遗憾的是，这种格式没有很好的结构，因而很难解析。

Asterisk 的核心与某些模块有密切的联系。由于核心使用了一些模块中的二进制代码，当它所依赖的某个模块出现问题，Asterisk 就根本无法启动。如果你想打一个电话，至少在 Asterisk 1.2 中，除使用 `app_dial` 和 `res_features` 外你别无选择，这是因为建立一个呼叫的代码和逻辑实际上是在 `app_dial` 中，而不是在核心里。同时，桥接语音的顶层函数实际上包含在 `res_features` 中。

Asterisk 的 API 没有保护，大多数的函数和数据结构都是公有的，极易导致误用或被绕过。其核心非常混乱，它假设每个信道都必须有一个文件描述符，尽管实际上某些情况下并不需要。许多看起来是一模一样的操作，却使用不同的算法和截然不同的方式来实现，这种重复在代码中随处可见。

这仅仅是我对 Asterisk 中遇到的最多的问题一个简要的概括。作为一个程序员，我贡献了大量的时间，并贡献了我的服务器来作为 CVS 代码仓库和 Bug 跟踪管理服务器。我曾负责组织每周电话会议来计划下一步的发展，并试图解决我在上面提到过的问题。问题是，当你对着长长的问题列表，思考着需要花多少时间和精力来删除或重写多少代码时，解决这些问题的动力就渐渐的没有了。值得一提的是，没有几个人同意我的提议并愿意同我一道做一个 2.0 的分支来重写这些代码。所以在 2005 年夏天我决定自己来。

在开始写 FreeSWITCH 时，我主要专注于一个核心系统，它包含所有的通用函数，即受到保护又能提供给高层的应用。像 Asterisk 一样，我从 Apache Web 服务器上得到很多启发，并选择了一种模块化的设计。第一天，我做的最基本的工作就是让每一个信道有自己的线程，而不管它要做什么。该线程会通过一个状态机与核心交互。这种设计能保证每一个信道都有同样的、可预测的路径和状态钩子，同时可以通过覆盖向系统增加重要的功能。这一点也类似其他面向对象的语言中的类继承。

做到这点其实不容易，容我慢慢讲。在开发 FreeSWITCH 的过程中我也遇到了段错误和死锁

¹¹ Avoiding initial deadlock

(在前面遇到的多，后来就少了)。但是，我从核心开始做起，并从中走了出来。由于所有信道都有它们自己的线程，有时候你需要与它们进行交互。我通过使用一个读、写锁，使得可以从一个散列表(哈希)中查找信道而不必遍历一个线性链表，并且能绝对保证当一个外部线程引用到它时，一个信道无法被访问也不能消失。这就保证了它的稳定，也不需要像 Asterisk 中『Channel Masquerades』之类的东西了。

FreeSWITCH 核心提供的的大多数函数和对象都是有保护的，这通过强制它们按照设计的方式运行来实现。任何可扩展的或者由一个模块来提供方法或函数都有一个特定的接口，从而避免了核心对模块的依赖性。

整个系统采用清晰分层的结构，最核心的函数在最底层，其他函数分布在各层并随着层数和功能的增加而逐渐减少。

例如，我们可以写一个大的函数，打开一个任意格式的声音文件向一个信道中播放声音。而其上层的 API 只需用一个简单的函数向一个信道中播放文件，这样就可以将其作为一个精减的应用接口函数扩展到拨号计划模块。因此，你可以从你的拨号计划中，也可以在你个性化的 C 程序中执行同样的 playback 函数，甚至你也可以自己写一个模块，手工打开文件，并使用模块的文件格式类服务而无需关注它的代码。

FreeSWITCH 由几个模块接口组成，列表如下：

- 拨号计划(Dialplan): 实现呼叫状态，获取呼叫数据并进行路由。
- 终点(Endpoint): 为不同协议实现的接口，如 SIP, TDM 等。
- 自动语音识别/文本语音转换(ASR/TTS): 语音识别及合成。
- 目录服务(Directory): LDAP 类型的数据库查询。
- 事件(Events): 模块可以触发核心事件，也可以注册自己的个性事件。这些事件可以在以后由事件消费者解析。
- 事件句柄(Event handlers): 远程访问事件和 CDR。
- 格式(Formats): 文件格式如 wav。
- 日志(Loggers): 控制台或文件日志。
- 语言(Languages): 嵌入式语言，如 Python 和 JavaScript。
- 语音(Say): 从声音文件中组织话语的特定的语言模块。
- 计时器(Timers): 可靠的计时器，用于间隔计时。
- 应用(Applications): 可以在一次呼叫中执行的程序，如语音信箱(Voicemail)。
- FSAPI(FreeSWITCH 应用程序接口)：命令行程序，XML RPC 函数，CGI 类型的函数，带输入输出原型的拨号计划函数变量。
- XML：到核心 XML 的钩子可用于实时地查询和创建基于 XML 的 CDR。

所有的 FreeSWITCH 模块都协同工作并仅仅通过核心 API 或内部事件相互通信。我们非常小心地实现它以保证它能正常工作，并避免其他外部模块引起不期望的问题。

FreeSWITCH 的事件系统用于记录尽可能多的信息。在设计时，我假设大多数的用户会通过一个个性化的模块远程接入 FreeSWITCH 来收集数据。所以，在 FreeSWITCH 中发生的每一个重要事情都会触发一个事件。事件的格式非常类似于一个电子邮件，它具有一个事件头和一个事件主体。事件可被序列化为一个标准的 Text 格式或 XML 格式。任何数量的模块均可以连接到事件系统上接收在线状态，呼叫状态及失败等事件。事件树内部的**mod_event_socket** 可提供一个 TCP 连接，事件可以通过它被消费或记入日志。另外，还可以通过此接口发送呼叫控制命令及双向的音频流。该套接字可以通过一个正在进行的呼叫进行向外连接(Outbound)或从一个远程机器进行向内 (Inbound)连接。

FreeSWITCH 中另一个重要的概念是中心化的 XML 注册表。当 FreeSWITCH 装载时，它打开一个最高层的 XML 文件，并将其送入一个预处理器。预处理器可以解析特殊的指令来包含其他小的 XML 文件以及设置全局变量等。在此处设置的全局变量可以在后续的配置文件中引用。

如，你可以这样用预处理指令设置全局变量：

```
<X-PRE-PROCESS cmd="set" data="moh_uri=local_stream://moh"/>
```

现在，在文件中的下一行开始你就可以使用`$$ {moh_uri}`，它将在后续的输出中被替换为`local_stream://moh`。处理完成后 XML 注册表将装入内存，以供其他模块及核心访问。它有以下几个重要部分：

- 配置文件：配置数据用于控制程序的行为。
- 拨号计划：一个拨号计划的 XML 表示可以用于**mod_dialplan_xml**，用以路由呼叫和执行程序。
- 短语：可标记的 IVR 分词是一些可以『说』多种语言的宏。
- 目录：域及用户的集合，用于注册及账户管理。

通过使用 XML 钩子模块，你可以绑定你的模块来实时地查询 XML 注册表，收集必要的信息，以及返回到呼叫者的静态文件中。这样你可以像一个 WEB 浏览器和一个 CGI 程序一样，通过同一个模型来控制动态的 SIP 注册，动态语音邮件及动态配置集群。

通过使用嵌入式语言，如 Javascript、Java、Python 和 Perl 等，可以使用一个简单的高级接口来控制底层的应用。

FreeSWITCH 工程的第一步是建立一个稳定的核心，在其上可以建立可扩展性的应用。我很高兴的告诉大家在 2008 年 5 月 26 日将完成 FreeSWITCH 1.0 PHOENIX 版。有两位敢吃螃蟹的人已经把还没到 1.0 版的 FreeSWITCH 用于他们的生产系统。根据他们的使用情况来看，我们在同样的配置下能提供 Asterisk 10 倍的性能。

我希望这些解释能足够概括 FreeSWITCH 和 Asterisk 的不同之处以及我为何决定开始 FreeSWITCH 项目。我将永远是一个 Asterisk 开发者，因为我已深深的投入进去。并且，我也希望他们在以后的

Asterisk 开发方面有新的突破。我甚至还收集了很多过去曾经以为已经丢失的代码，放到我个人的网站上供大家使用，也算是作为我对引导我进入电话领域的这一工程的感激和美好祝愿吧。

Asterisk 是一个开源的 PBX，而 FreeSWITCH 则是一个开源的软交换机。与其他伟大的软件如 Call Weaver、Bayonne、sipX、OpenSER 以及更多其他开源电话程序相比，两者还有很大发展空间。我每年都期望能参加在芝加哥召开的 ClueCon 大会，并向其他开发者展示和交流这些项目 (<http://www.cluecon.com>)。

我们所有人都可以相互激发和鼓励以推进电话系统的发展。你可以问的最重要的问题是：『它是完成该功能的最合适的工具吗？』

1.8 FreeSWITCH 的历史¹²

Anthony Minessale/文 杜金房/译

为了恰当地介绍 FreeSWITCH 的起源，我们必须回到从前，那时，我们甚至还没想到要实现它。VoIP 革命真正的开始与成型是在世纪之交，以开源的 Asterisk PBX 和 OpenH323 项目为主要标志。这两个软件的革命先驱使得很多开发者得以访问 VoIP 的资源而无需付出高昂的商业解决方案的费用。这两个项目后来又导致了很多创新，真正的可能的 IP 电话通信是真实存在的被迅速传播开来。

我在 2002 年第一次进入这一行业。当时我们公司的业务是对外技术支持外包，我们需要一种方式管理电话呼叫，并把呼叫送到一个线下的地方（原文是 an off-site location）。当时我们用的是的解决方案，但是那种方案不仅部署费用太贵，而且我们还得支付不菲的每座席的费用。作为一个 Web 平台的架构师，我在过去的工作中曾经基于开源项目如 Apache 和 MySQL 做过很多开发，所以我决定研究一下在电话方面有没有相关的开源解决方案。很自然地我找到了 Asterisk。

当我第一次下载了 Asterisk 的时候，我惊呆了。为了使它工作，我找来了好多模拟电话板卡，然后中，在我家里，装在我的 Linux PC 机上，当在后面插上电话线并在话机上听到第一声拨号音之后，我叫到：哇塞！简直是帅呆了！我很快就深入代码中，试着研究出它是怎么工作的。我很快的学到，类似 Apache，该软件竟然也可以以可加载模块的方式扩展它的功能以做其它的有用的事情。这比以前任何的东西都要好。现在，我不仅可以使自己的电话可以与计算机通信，我还可以让它在拨打特定号码的情况下执行我自己写的代码。

我尝试并验证了几种想法之后，忽然，我产生了一个新想法：嘿，我非常喜欢 Perl¹³，并且这些电话系统非常的酷，如果我把它们结合起来会怎么样？我研究了如何把 Perl 嵌入 C 语言程序的文档，很快我就有了一个 `app_perl.so`，它是一个 Asterisk 的可加载模块，通过该模块可以在电话路由到我的模块以后执行我的 Perl 代码。当时它并不完美，然后我开始很快地学习将 Perl 嵌入一个多线程的程序中的各种挑战。但至少是对我的想法的可行性的一种概念验证，在经过几天的修修补补之后能够得以运行也算是一个不小的成熟了。

¹²本文译自《FreeSWITCH 1.2》(PACK 出版社，2013)，附录 C: The History of FreeSWITCH。翻译得到作者授权。

——译者注

¹³一种编程语言。——译者注。

随后，我深入了 Asterisk 在线社区。在这些代码上玩了几周的之后，我用 Asterisk 作为电话引擎开始做一个呼叫中心解决方案，以及一个简单的 Web 前端程序。在实现的过程中，我遇到了 Asterisk 中的几个 Bug，然后我就把它们提交到了 Asterisk 开发分支的缺陷跟踪系统上。该过程重复地越多，我参与该项目的成长和发展就越深。除了零散的 Bug 修复之外，我也开始写代码对它进行改进。到 2004 年的时候，我除了修复我报告的 Bug 之外，也在修复其它人报告的 Bug 了。这是我感觉在我找到我自己的问题的免费解决方案以后，我所能做到的回报方式。如果我的问题解决了，大家也都能看到。

事件升级

当我在测试我的程序的时候，我会往系统中打很多电话并看着一个 Web 页面在更新、控制呼叫队列，以及看着各种状态统计信息。然而，我没有注意到的一件事就是并发的呼叫数以及呼叫量本身。确实我在测试的时候也就一般同时打一两个电话，而没有全面的测试我的程序。当我第一次将程序放到生产系统上的时候，也是我第一次看到一个多线程的软件遇到无法解决的锁冲突的时候，这种锁冲突就是众所周知的死锁。我非常熟悉段错误¹⁴，因为我在开发自己的模块的时候遇到过无数次。但是，使我不解的是，我有时看到在某些非常无法解释的情况下也出现这种错误。

段错误是由于一个程序在运行时进行了不正当的内存访问，如多次释放同一段内存或者试图越界访问内存地址或者访问根本不存在的内存。由于你可以直接访问底层的操作系统，并且除非你非常自律，系统无法防止你犯错误，所以，在 C 语言编写的程序中这种错误是很常见的。但我不是轻言放弃的人，那样的话你会认为是一个诅咒抑或是恩赐。所以，当我遇到问题时，我已经准备好了奋斗到底。我花了无数的时间研究 GNU 调试器的输出，并尝试模拟出大量的呼叫以便重现我遇到的问题。经过一些试错（原文是 trail-and-error），我成功了。我终于通过一个呼叫发生器的帮助找到了导致系统崩溃方法。当时的感觉非常好，只是好的感觉太短暂。就在那天下午，我又在代码的另一处发现了另外一个类似的新问题。

我尝试小心的去掉我的程序中的一些可能导致死锁或崩溃的功能，但是我无法去掉全部。最终我发现导致我的不幸的是 `app_queue` 模块，这对我来说可不是个好消息，因为在我的呼叫中心程序中我主要就用了那个模块。我修复了那个模块中的问题，但一些修改对原有的模块影响太大，因而无法包含到主流的发布代码中。最后，我还是自己维护了我的模块代码，并继续更新 Asterisk 中其它的部分。这总算令它稳定下来了，但这种稳定只是在找到另一种解决方案前相对稳定的方案。

到那时为止，我往 Asterisk 里加入了很多的新特性，并对开发一些性功能有了很好的想法。我创建了一个新的概念，叫做功能变量（function variables），它允许模块可以对外提供一个接口，通过该接口，能从拨号计划（Dialplan）中扩展模块的功能（如果你读了本书，你会感觉本书中同样实现了类似的想法）。与此同时，我仍然在纠结那个队列死锁问题。所以后来我与 Asterisk 社区中的另一个成员开始计划一个新的队列模块——`mod_icd`。

ICD 的代表智能呼叫分配（Intelligent Call Distribution），与首字母缩写的自动电话分配（ACD，Automatic Call Distribution）相对。我们找出了 `app_queue` 模块所有的问题，我们对做一些新的稳定的，再也不会导致无休止的死锁和崩溃的模块同样感兴趣。我们使用状机以及更高级的基于内存池的内存管理抽象以及其它一些在标准的 Asterisk 中不存在的创造性地概念。但问题是，我觉得我们在

¹⁴即 Segmentation Falult，是程序运行期间由于共享的内存遭到破坏而引起的程序崩溃。——译者注。

那个模块上做过了头，看起来几乎是我们把 Asterisk 核心的一些功能也边缘化了。当然，那只是其中的一个可载模块，完全边缘化 Asterisk 的核心是不可能的。

我们始终没有完全完成`mod_icd`。在 2004 年底，我参与呼叫中心解决方案的机会被那些不可饶恕的段错误和死锁的深海击碎、涤荡殆尽。我们开始关注另外的与队列无关的电话服务。我使用了我添加到主流的 Asterisk 中的几个新特性以及我的几个未被批准的不太流行的小模块开发了一个新的被叫付费业务（类似中国的 800 电话）以及传真转电子邮件服务。我建立了一个由 7 台 Asterisk 主机组的集群，将它们与电信部门提供的线路对接。这种部署 Asterisk 的方式并不是不会出问题，只不过比较美好的一点是，如果某台机器崩溃，就会有另一台顶上去，然后我们就有机会重启那台崩溃的机器。

新点子和新项目

在这一点上我积累了一些新想法——有的测试过、有的没有，还有一些需要对 Asterisk 做一些大的改动。我跟我的队友 Brain West 以及 Michael Jerris 在 Asterisk 项目上贡献了很多时间。我们帮助管理缺陷跟踪系统（Issue Tracker），我们修复了很多 Bug，并且我们每周都主办开发者的电话会议，甚至我们还在我们的服务器上做了一个代码库镜像站点。我们参与的太多以至于我们的一些新想法在 Asterisk 社区中引起了一些政治骚乱，起因在于一场不同的开发者之间的一场没必要的竞争——每一个 Asterisk 的贡献者必须签署一个表格以声明他们写的所有日后可能用于 Asterisk 的代码都自动对 Digium 公司（Asterisk 的拥有者）有一个免版税的授权，以便他们可以用你的代码做任何他们喜欢的事。如，通过这种方式，他们可以将这种无限的许可证以高的多的价格卖给他们的潜在客户。这完全背离了开源精神，但这就是另一个故事了。我认为这种差异化引起了一些跟我一样的志愿开发者与 Digium 雇佣的一些 Asterisk 开发者之间的一些冲突。

即使在这么紧张的环境下，我们还是全力以赴地支持该项目真正希望它能成功。我们继续召开每周的电话会议，他们也确实采取一些措施开始帮助开发者们增加动力。我们认为我们应该有一场现场的见面会，以便我们所有人都可以聚到一起分享我们的电话技术知识，并一起玩几天。我们不知道我们要干什么，但我们决定要做，并把该聚会称为 ClueCon。有一个 Clue¹⁵意味着你知道你要做什么，所以 ClueCon 的意思就是帮每一个人找到一个『Clue』。我承认，即我刚刚说过我们也不知道我们要做什么。看起来非常有意思，一群没有 Clue 的人开了一个有 Clue 的会——ClueCon。不过，事实证明非常幸运，这好像根本不是个问题。前面我们所指的 Clue 都是指电话技术，而不是做会议。

因此，在第一届 ClueCon 之前的几个月中，即 2005 年春天，我们在一次例行的电话会议中开始专门深入讨论 Asterisk 中的几个缺点。这非常正常，因为我们主要的目标就是找出这些问题并找到解决方案。在那个时期，有一大群不守规矩的、受够了他们在 Asterisk 上遇到的无穷无尽的问题的人。那人中的很多人都参加了那次周会，希望能说服我们帮忙看一看他们遇到的问题。我想得越多，越觉得解开折磨我们的核心架构问题越是任重道远。Asterisk 中的很多核心都具有单一的性质而无法扩展（Scale），其中的很多我发都有很多用户依赖于它们，任何企图改进他们的动作都有可能会引发功能上的退化。有些问题看起来是无解的，除非用一把大锤把那些旧代码砸个稀巴烂，并从核心的代码深入重写。但这种方法看似不可行，因为它将会使得 Asterisk 在几个月内甚至一年或更长的时间内都不可用。也就在那时候，我有了一个想法——我们做一个 2.0 版吧。

¹⁵线索，后面的 Con 指会议（Conference）。——译者注。

做一个 2.0 版并不是一个最坏的想法。我知道它将有很多挑战，但是，我想我们可以与旧的代码并行启动一个新的代码库，那样我们就可以删除那些有问题引起问题最多的部分旧代码并替换成新的，并仍然维护用户依赖的一些仍然可用的代码。有了这个想法后我感到很兴奋，同时也在我提出该想法后看到该项目的领导人的反映时得到了同样地震惊。他似乎也对我竟然提出这么一个想法同样震惊。总之，简单来讲，我们没有做 2.0 版。那时，我有无数的想法，我也非常清楚地知道我喜欢以及不喜欢 Asterisk 的地方，但没地方写。

我盯着那个在一个空目录中打开的空的编辑器缓冲区¹⁶，盯了足足一个小时。我知道我想要做什么，但不知道怎么写出来。在我在编辑器中增加了一些奇怪的单词以及一些标点符号之后，我才知道该如何开始。那些单词并不是你常用的单词，而是一些符号以及变量声明——我是在写 C 语言代码。几天后，我用 C 写了一个基本的程序，试验了几个我在过去编程中喜欢用的一些编程工具。我有 Apache 可移植性运行库（或称 APR），有 Perl 编程语言，以及一些其它的程序包。我建立了一个核心，以及一个可加载模块的结构，一些助手函数用于内存池管理，并且我有了一个简单的命令提示符，你可以在命令行上键入 `help`，如果你愿意看到命令行上显示一个尖刻的提示，提示你根据没有帮助信息的话，或者也可以键入 `exit` 以终止程序。我还写了一个示例模块，允许你使用 telnet 连接到一个特定的 TCP 端口上，你输入的任何字符都将原来的回显回来。另外我还实现了一个简单的状态机。我把这个程序叫做 Choir。因为我希望我的一系统的想法都可以像教堂里的唱诗班一样发出和谐的声音。在那些最初的代码之后，我放了一段时间。因为 ClueCon 快要来了，我还有许多东西要准备，而没有想到时候太仓促。

第一届 ClueCon

2005 年 8 月的 ClueCon 是第一届。当时参加的有几个 VoIP 项目的领军人物，包括 OpenH323 的作者之一 Craig Southeren 以及 Asterisk 的创建者 Mark Spencer，当然，不喜欢我的 Asterisk 2.0 的想法的也是他（Mark），但无论如何，将这些人聚到同一间屋子里还是一件非常酷的事情。我们整天都有演示，以及反复地交流，并且，我们真正使得每人都开始想问题。那一届 ClueCon 非常成功，会议圆满结束以后我动力十足，并准备好继续写我的 Choir 代码。但事实是，我并没有立即开始行动，而是在我们的电话会议上讨论了几个月，同时挣扎在时时刻都有倾覆危险的 Asterisk 平台上。秋天马上就到了，Asterisk 社区的混乱最终导致了一场苦命——社区中占相当比例的人 Fork¹⁷了 Asterisk，新的项目叫做 OpenPBX。

我完全理解他们为什么那么做，并尽我所能地支持他们。我贡献了我所有为 Asterisk 所写的代码，他们可以根据自己的喜好随时取用。如果有闲暇时间，我也会帮助他们，但我最终还是没有完全融入他们。因为我仍然有同样的问题没有解决——我认为有些问题必须从最底层解决，而这一新项目（指 OpenPBX）的创始人更倾向于解决那些 Asterisk 团队没有能够及时解决的现实的问题。我们仍然开电话会议，但大多数情况下 Asterisk 项目的人都不再参加了，因为我们为 Asterisk 社区的革命欢呼使得他们不高兴。由于在不将 Asterisk 核心完全推倒重来的情况下我无法修复任何问题，有一天我为此事道了歉。也就在那时候有人（如 Tootsie Pop commercials 的 Owl 先生）问我：『你觉得让你的新代码能打电话需要多长时间呢？』我不知道，所以我决定试一下——不管是一周、两周还是三周。

¹⁶ 用于编写程序代码的编辑器。作者使用 Emacs 编辑器，在 Emacs 中，每一个文件（甚至 Shell 等）都称为一个缓冲区（Buffer）。——译者注。

¹⁷ 专业术语，指在原来代码库的基础上重新建了一个分支，然后两个分支分别独立发展。

我写的第一个能够发出声音的模块是**mod_woomera**, 该模块是一个 Endpoint 模块, 它使用了 Craig Southeren (与我在 ClueCon 上遇见的是同一个人) 写的 Woomera 协议。我也为 Asterisk 写了一个类似的模块。Woomera 协议非常简单, 它并不需要编解码或其它复杂的东西。它的实现思想是它屏蔽了 H323 协议的复杂性并允许应用程序使用该简单的协议与它通信以便更容易地集成到 VoIP 程序中。所以, 从它开始好像是一个正确的选择。当我开始工作的时候, 我意识到在我的核心代码中需要更多的元素, 然后我就慢慢的添加, 并最终将这些代码融合到一处, 我终于可以给以 Woomera 协议武装的 H323 监听进程打电话了, 同时, 我也可以在我的 Pandora 代码里获取通话的状态。是的, 我把我的项目名称改成了『Pandora』, 因为大家都不喜欢 Choir 这个名字。我非常愉快的听着 Alan Parsons Project hit Sirius stream 第一次从我的扬声器里流出来。这一次比我第一次使用 Asterisk 打电话时更加兴奋, 因为我白手起家从头开始写的代码现在开始工作了。

现在, 我已经有所进展了。我研究出了如何让两个 Channel 桥接到一起、如何支持更多的其它协议以及做一些其它的基础的事情, 而不是仅仅打印一条尖刻的帮助信息并退出。有人建议把这些代码叫做 OpenPBX 2, 有人也建议其它名字。在经历了足够的命名争论后, 我知道了 (并永远决定了) 我应该叫它什么: FreeSWITCH。我终于有了一个我将为之坚持不懈的名字、一些可以工作的代码, 以及很多野心。我埋下头继续工作。所有地方都有工作要做, 多得甚至你都没时间去想。所以我就不停地写代码。时间很快到了 2006 年 1 月, 那时我有足够的代码可以与公众分享了。我们向开发者们开放了我们的代码库。通过让他们注册一个开发者账号才能获取代码的访问权限, 我们确保只有非常严肃的开发者才愿意完成整个注册过程。有一些人下载了¹⁸源代码并提供了一些反馈, 我们当时真觉得我们有了一个真正的项目。

我们有了一个可以桥接电话的模块、一个可以放音的模块、一些编解码模块、一些作为例子的拨号计划 (Dialplan) 模块, 以及一些其它的模块。哦, 我有没有说过它在 Windows 上也可以运行¹⁹?

虽然页面上所有的链接都失效了, 但我们原来的站点仍然保留着: http://www.freeswitch.org.old_index.html。

FreeSWITCH 诞生

在实现我们计划的过程中, 我们确实写出了可以运行于 Windows、Linux 以及 Mac OS X 上的代码。我早期团队的伙伴——Michael 和 Brian, 从一开始就跟在我一起。Mike²⁰在 Windows 平台上很有经验, 他确保了我们的代码能在 MSVC 里编译和运行。最初这确实是一个痛苦的过程, 但在修复了无数情况下无数的编译错误之后, 我第一次开始学习如何编写跨平台的代码。光阴似箭, 下一次 ClueCon 的时间到了。在那一年, 我进行了我的第一次 FreeSWITCH 演讲, 演示了在本书开篇²¹中所描述的核心设计和基础架构。我们看到了非常令人兴奋的模块, 如一个可以与 Google Talk 通信的模块。在演讲过程中, 我也通过**mod_exosip**模块现场演示了在有几千个并发呼叫的情况下呼叫的实时建立和释放。那是一个很好的演示, 但我们并不满足。

Exosip 仅仅是在 Osip 基础上进行了一些上层的封装, 而原来的 Osip 库则是一个开源的 SIP 协议栈, 它提供了大多数的 SIP 功能。Exosip 使得开发一个 Endpoint 模块更简单一些, 所以, 我们决

¹⁸原文是 Check out, 即从 SVN 仓库检出代码。

¹⁹FreeSWITCH 可以在 Windows 上原生的运行, 而 Asterisk 不能, 或者只能通过 Cygwin 等模拟环境运行。

²⁰Michael 的简称。——译者注。

²¹该书第一章是 Architecture of FreeSWITCH, 即 FreeSWITCH 的架构与设计。——译者注。

定基于它来开发我们的 SIP 模块。但后来，我们还是遇到了几个灾难性的问题，使得我开始感觉我们陷入了与当时让 Asterisk 正常工作一样的境地。因而，我们开始寻找一个 Exosip 的替代者。我们寻找替代者的另一个原因是 Exosip 选择了 GPL 许可证，而不顾原始的 Osip 库本来是 LGPL（就我个人感觉，应该继续选择 LGPL 更合理一些），这就导致了潜在的许可证冲突。由于我们在我们的项目中使用的是 MPL 许可，而 GPL 协议不允许使用 GPL 许可的代码嵌入 MPL 许可的程序中。有关许可证的争论很有趣，也经常能令人兴奋，但当时我们没有时间参与这些。

由于在 FreeSWITCH 中对 SIP 功能有很高的要求，我们上天入地，找遍了开源界的每一寸土地，希望能找到一个可以用的新的 SIP 协议栈以及 RTP 协议栈。我们针对这两点评估了几个库，最终几乎每种协议都试用了至少 5 个库，但还是没有找到一个令我满意的 RTP 协议栈，所以最终我决定自己实现。当然，我并不至于傻到也自己去实现 SIP。在我看到 Asterisk 曾试图从头到尾写一个 SIP 协议栈并最终失败而转投 Exosip 之后，我继续在开源领域中寻找 SIP 协议栈，直到最后发现了诺基亚 (Nokia) 开发的 Sofia-SIP。我们写了一个可用的 mod_sofia 模块来进行测试，效果非常不错。我们继续打磨该模块直到它可以完全替代 mod_exosip，然后 mod_sofia 就成了我们系统中首要的 SIP 模块。不过，那仅仅是开始。因为到后来，即使是现在我还经常要往 mod_sofia 中添加代码。SIP 是一个非常复杂并且令信恐惧的协议，它带来了许多令人不愉快的思想，而不像它的名字看起来那样简洁。但现在不是讨论这个的时候。

我们在 ClueCon 2007 上再一次演示了 FreeSWITCH，那一次，有了一个新的 SIP 模块以及更多的代码。另外，还有 OpenZAP，它使用一个 TDM 库将 FreeSWITCH 连接到电话硬件上。OpenZAP 后来被 FreeTDM 替代，现在由 Sangoma 公司负责维护。我经历了使用同样的板卡，很久以前让它在 Asterisk 上工作，现在又让它在 FreeSWITCH 上工作的愉快过程。我们曾经很快地宣布我们将推出 FreeSWITCH 1.0.0 版。很多看过我们原来的主页的人可能会注意到当时我宣布了一个官方的新版本将『很快发布 (oming soon)』，的消息的发布时间是 2006 年 1 月，当时我们拼命的想让所有事情按我们希望的那种方式工作。我们非常希望专注于在添加任何其它功能前做一个稳定的核心，并且我们也有了很大的进展，但是我们还是没有准备好发布 1.0 版。

2008 年春天我们有了稳定的 SIP、有了 Event Socket 用于远程控制 FreeSWITCH、有了一个模块可以通过 HTTP 执行 FreeSWITCH 命令、有了 XML curl 等等一系列的新功能和特性。我们最终觉得可以发布一个版本了。所以我们就一举发布了 FreeSWITCH 1.0 凤凰版 (Phoenix)。我之所以将该版本取名为凤凰，是因为感觉到我们所有的辛苦的工作成果都是从先前的失败的骨灰中来的，并且这个名字也被很多其它人使用，包括 NASA 在同一时刻将『凤凰号』送上了火星。总之我认为那是一个很合适的名字。

在 ClueCon 2008 上，我们又一次宣布了曾于当年 5 月份发布的 1.0 版。当时还有另外一些与 FreeSWITCH 有关的演讲，以及与 Asterisk 有关的演讲，因为当前也发布了 Asterisk 1.6。在当前接下来的时间里，我们用了所有的时间专注于宽带（高清）语音的支持以及其它的一些功能，例如即时的进行采样率转换。此外，我们还增加了一些新的 SIP 功能，如状态呈现（SIP Presence）以及其他的一些简单通话以外的功能，并于后来发布了 1.0.1 版。

2009 年，我们发布了 1.0.2 至 1.0.4 并于第 5 届 ClueCon 年度会议上又一次演示了 FreeSWITCH。到那时候，我们早期的一些创新变成了现实。因为我们可以演示使用 Polycom 话机新的 Siren 编码进

行高清语音的通话，并且我们还支持了与 Skype 互通。当年的 FreeSWITCH 演示概括了一些你可能根本就意识不到你可以做的事情，除非你具有四维空间的想象力。而这，正是 Emmett Brown 博士（来自电影《回到未来 (Back To The Future)》）都想做的。FreeSWITCH 有一些与 Asterisk 类似的行为，但是我们同时也有了一个新的词汇表，该词汇表新像为你打开一个通向无限可能的空间的大门，使你可以通过一个 PC 和一个电话就可以做任何无法想象的事情。

2012 年的 ClueCon 是在 Trump Tower²²举行的。它是一届有史以来最值得怀念的 ClueCon。当年我们出版了本书的第一版，我们还发了几本做为奖品。我们详细演示了 FreeSWITCH 以及它的性能。当年我们发布了 1.0.5 和 1.0.6。那年的主题好像是 Erlang，好像每个人都赶上的事件驱动 (Event Driven) 架构的时尚。所以，我们绝对是在合适的时候处在了合适的位置。

2011 年是该项目大跨跃的一年。受我们一年前自己关于性能的演讲的启发，我们对 Sofia SIP 模块进行了大刀阔斧地修改，将原来串行化的消息处理改为并行的处理，每个路电话的消息都会被推到它自己的线程中处理。这次改变产生了很多并行的操作，避免了由于单一的 Channel 发生问题时阻塞整个 SIP 协议栈的可能。那年的 ClueCon 是在壮丽的 Sofitel 酒店举行的。我们演示了很多新特性，包括一些用于帮助开发者进行开发的特性，如变量数组的概念以及范围变量——你可以使用它来设置一个通道变量，该变量仅在某一特定 App 执行的时候才有效。

2012 年我们宣布了一个新的计划，在 FreeSWITCH 代码库中支持稳定版的分支 (stable branch)。这是一个令人望而却步的任务，因为你必须将成熟的代码与新的代码分离，并且在每次修改时都需要在不同的分支上做额外的检查以确保所有东西都平稳运行。我们为此非常努力地工作。并且本次新版的书将包含 FreeSWITCH 1.2 稳定版中最初版本的内容。我们在 Hyatt 酒店举行了一次很成功的 ClueCon，并演示了一些新的特性如支持 Hylafax 的软件模拟器以及一个新的 mod_httapi 模块，该模块也在本书前面的章节中讲到了。

在写本书的时候，已是临近 2013 年了。在活过了玛雅人的预言²³之后，我们开始研究消除传统的电话与 HTML5 以及 WebRTC 之间的间隙以及第一个 FreeSWITCH 1.4 Alpha 版本。ClueCon 将继续在 Hyatt 酒店举行²⁴，他们为给我们打造更温馨的环境重新进行了装修。我们希望在那儿见到你们所有人，并希望你通过本书多学到了一丁点儿的 FreeSWITCH 的知识以及了解为我为什么决定在那个空白的文本编辑器上开始打上那几个字符²⁵并最终变成 FreeSWITCH 核心组件的近 50 万行代码的。

1.9 The missing Link

本文主要讨论应用程序如果依赖其它库的话，应该静态链接还是动态链接。在这一点上，我们看看 FreeSWITCH 是怎么做的。

曾经，在 2007 年有个叫 Roman Shaposhnik 的哥们发了一篇博客文章—«[What does dynamic linking and communism have got in common?](#)»，上面讲到，在一个理想的世界中，所有的库都

²²一个国际性酒店。——译者注。

²³传说玛雅人的历法中预言 2012 年世界将灭绝。——译者注。

²⁴本文写于 ClueCon 2013 之前，后来，ClueCon 2013 如期在 Hyatt 举行，当年的主题好像是 WebRTC。译者曾在现场。
——译者注

²⁵指作者最初写 FreeSWITCH 代码的时候。——译者注。

应该是动态链接的，这样可以最大限度的节约磁盘空间和内存，程序也更有效率，少出 Bug。

但是，FreeSWITCH 的作者 Anthony Minessale 却有不同的话说。该文章来自<http://www.freeswitch.org/node/56>，原文翻译如下：

最近，我读了 Roman Shaposhnik 的博客文章，是关于静态连接的：https://blogs.oracle.com/rvs/entry/what_does_dynamic_linking_and。

作为 FreeSWITCH 阵营的代表，我想，我应该说点什么。

对于整个问题，我可以提供一个完全不同的视角。我们的项目（FreeSWITCH）我很多我们自己的代码，同时也有一系列的依赖库，这些依赖库大多数在一些外部模块中使用。现在，Roman 在它的文章中说，在一个理想的世界里，应该只有一个操作系统，并且只有一个唯一实体管理着相同的环境。为了能在你的系统上运行我们的软件，我只需要向该实体请求我需要的环境，包括我们的软件依赖的各种程序库。

我们遇到两个主要问题。一是我们的目标是跨平台的，而『跨平台』在我们心目中的含义是它能尽可能地运行在任何操作系统以及任何硬件上。实际上，这本身就是个恶梦。与 Roman 所描述的理想世界相反，由于这个世界上有 Windows、Mac OS X，数十种 Linux 发行版、以及 Solaris 等各种不同的操作系统，它们又各有不同的要求，将这个世界带入极端的无政府主义的混乱状态。当我试图在任何一种平台上编译我们的软件时，我几乎必然会遇到问题。令人痛苦的事实是，所有这些平台或厂商的维护者以及我们所依赖的库的开发者们都没有像我们一样想快一点使所有东西都完美无缺。我甚至怀疑他们是否跟我们一样对『跨平台』支持执着。这让我想到第二个问题——即使我们非常幸运地在某个平台上找到了我们依赖的库，我们又如何知道它是按我们所需的方式配置（configured）的呢？很多软件包都有数十种编译时的配置参数，其中有很多参数是我们必须的，或者说，少了某些参数我们的软件也就无法实现某些功能，这不是我们愿意见到的。另外，为了在所有不同的系统上礼貌地要求我们所需的依赖，我们应该如何打包我们的软件呢？只有一种选择，那就是，我们必须维护我们自己依赖的代码并编译我们所需要的版本的库。这么做也是因为我们并不想野蛮的将我们版本的库装到某人的系统上，并且，我们也绝对不想安装动态库，那样做通常会引起混乱（应用程序可能会找到错误的动态库）。所以，我们只是简单的将我们依赖的代码编译成静态库，并静态的连接到我们的可加载模块或我们的代码中而不会影响到别人。通过静态编译和连接，我们知道我们在运行时所用的代码就是我们开发时同样的代码，因此我们也能睡的安稳。所有这些决定跟运行效率或节约磁盘空间、节省内存等没有半毛钱关系，我的想法很简单，我就是想让我的软件能正常地运行。

必须指出，我们确实曾花了一些时间去研究是否有这样的代码存在——它是众所周知的、容易安装的或通常都是已经在操作系统上默认安装的。如果有这样的软件存在的话，并且我们能在所有目标系统上进行测试确认没有问题的话，我们是非常愿意用这样的系统提供的库的。这么说吧，迄今为止，似乎只有一个库符合这个标准，那就是 ODBC。它能入选的原因是它有一个定义很好的而且永久不变的二进制的 API 以及无数的实现。

1.10 在飞机上上网打电话

本文写于 2014 年 8 月在 ClueCon 完毕后回中国的飞机上。

在飞机上上网、打电话一直是人们的一个愿望。而前些日子看到有些关于在飞机上打电话的讨论。没想到，今天，我也用上了。

我乘坐的是美联航从芝加哥飞旧金山的航班，在芝加哥奥黑尔机场有貌似有免费 Wifi，不过，费了半天劲没登录上。航班延误 2 个小时，有些不爽，再加上昨天晚上睡得很晚，特别困，但也得坚持着。

终于登机了，在飞机门边上看到了 Wifi 的标志，小兴奋了一把。不过，后来想起大家的讨论，在飞机上上网可不是免费的，好像还比较贵。跟乘务员确认了一下 Wifi 是否可用，得到的答复是—Hopefully。

落座后，用我的 Nexus 平板试了一下，能连上 Wifi，但要飞机起飞后才能连互联网。困，就直接睡了。

一觉醒来，乘务员过来倒水。喝完后，试了一下 Wifi，果然可以购买了，\$1 和 \$2 两种选择。前者只能支持 Email 或 App，后者可以浏览网页，但两者都不支持流媒体。我选择了前者，一块钱（美元）可以上网一小时，真心不贵。

用信用卡支付后，互联网马上就开通了。上微信、QQ，都没什么障碍。Google Maps 也能正常打开，有点慢，但比起在国内翻墙上 Google Maps 的速度，感觉还是快。

Idapted (我的旧公司) 竟然都有北美微信群了，正好有个朋友在，便聊了聊，商量接下来怎么玩。

收发了几封邮件，一切正常。我使用 Gmail 客户端，比起在国内家里上网，一点都不感觉慢。

微信、QQ 发图也很快，只不过，当时国内的朋友都在睡觉，也没什么能聊的。

试了一下 FreeSWITCH 官方的 Verto Demo (WebRTC)，给 Brian 打电话，他正在通话，进入语音留言，便留了一段。本来再想给别人打电话，想了半天没想起来该打给谁，就算了。测了一下视频通话，第一帧看起来不错，后面的就花了。该 Demo 默认是使用 720p 的，需要的带宽比较高，所以，花屏可以理解，我估计如果改成 CIF 尺寸应该会有戏。

当然，不管视频是否有戏，语音通话是没有问题了（当然，是基于 VoIP/WebRTC 而不是基于手机的）。看来，虽然提示不支持互联网流媒体，但 WebRTC 还是通的。只是，我忘了上 Youtube 确认一下是否传统的流媒体真的就不行。

给 Brian 发邮件问他收到语音留言没，很快就得到了回复，说一切正常。我在准备给他回邮件的时候，一个小时的互联网时间结束了。

这是第一次在飞机上体验上网和打电话，希望所有飞机上都能以合理的价格尽快开通这项服务啊（美联航好像也还是 Beta 版）。当然，既然 VoIP 解决了打电话问题，用传统手机打电话的需求就不那么迫切了吧？

最后还想说一下，飞机座位下面是有电源的，要不然我也没法在飞机上用电脑写完这篇文章了。

1.11 You pay what I know

虽然一直是这个想法，可是听到 Brian 说出来，就感觉那么有哲理。

『You pay what I know』是说你要为我的知识付钱，而与之相对的『You pay what I do』则是为我的工作付钱。很容易理解，这绝对不是一个境界。

近几年，我一直做 FreeSWITCH 有关的咨询工作。为学习和使用 FreeSWITCH 的个人和企业提供一个强有力商业支持，一直是我坚持做的。不过，很惭愧，我的大部分工作还是『You pay what I do』的，看来，接下来，我需要向更高的目标前进了。

其实，对于真正 Pay 的人来说，很多人都着眼于你是真正干了多少活，而不是你作的工作对他们有多少价值。而我认为，这是不对的。对他们来说，你是否干活或干了多少本不是他们应该关注的，他们更应该关注的是你为他们解决了什么问题，帮他们省了或多赚了多少钱，进而决定应该支付你多少是否值得。

当然，要每个公司每个人都做到这一点还真是不容易的，这不仅在中国，在全世界都一样。而我们唯一能够做的，除了提高自身的能力外，便是不断的跟他们讲下面这个耳熟能详的故事 (Brian 也给我讲到这个)：

20世纪初，美国福特公司正处于高速发展时期，一个个车间一片片厂房迅速建成并投入使用。客户的订单快把福特公司销售处的办公室塞满了。每一辆刚刚下线的福特汽车都有许多人等着购买。突然，福特公司一台电机出了毛病，几乎整个车间都不能运转了，相关的生产工作也被迫停了下来。公司调来大批检修工人反复检修，又请了许多专家来察看，可怎么也找不到问题出在哪儿，更谈不上维修了。福特公司的领导真是火冒三丈，别说停一天，就是停一分钟，对福特来讲也是巨大的经济损失。这时有人提议去请著名的物理学家、电机专家斯坦门茨帮助，大家一听有理，急忙派专人把斯坦门茨请来。

斯坦门茨仔细检查了电机，然后用粉笔在电机外壳画了一条线，对工作人员说：『打开电机，在记号处把里面的线圈减少 16 圈。』人们照办了，令人惊异的是，故障竟然排除了！生产立刻恢复了！

福特公司经理问斯坦门茨要多少酬金，斯坦门茨说：『不多，只需要 1 万美元。』1 万美元？就只简简单单画了一条线！当时福特公司最著名的薪酬口号就是『月薪 5 美元』，这在当时是很高的工资待遇，以至于全美国许许多多经验丰富的技术工人和优秀的工程师为了这 5 美元月薪从各地纷纷涌来。1 条线，1 万美元，一个普通职员 100 多年的收入总和！斯坦门茨看大家迷惑不解，转身开了个清单：画一条线，1 美元；知道在哪儿画线，9999 美元。福特公司经理看了之后，不仅照价付酬，还重金聘用了斯坦门茨。

1.12 180 还是 183？

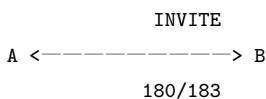
在 FreeSWITCH 中怎么配置给客户端回 180 还是 183，是一个经常被问到的问题。然而，答案却远没有你想象中的那么简单。

要明白怎么配置，首先你需要明白 180 和 183 的来龙去脉。另外，你自己还要知道你要干什么。

『什么？我提的问题我当然知道我要干什么！』也许你会这样咆哮，也许你真的知道你要干什么，但是，你得让我知道你要干什么，我才能回答你的问题啊。

好了，我们先不争论这个，我们来说说什么是 180 和 183。

在 SIP 通信中，所有 1 开头的响应叫临时响应，常见的有 100，180 和 183。这些响应一般是对 INVITE 请求的响应。使用场景是这样的：主叫用户（A）在发起一个呼叫时，会向被叫用户（B）发起一个 INVITE 请求，被叫用户在收到这个请求后，会给主叫用户一个响应，一般的响应流程是回 100，紧接着回 180，或（和）183。



其中，100 主要是信令层的，它的作用是 B 告诉 A 它收到了 A 的 INVITE 请求。关于它的作用实际也可以讲一讲的，但那就到信令的底层了，大家一般可以不用关注。

180/183 的作用是 B 告诉 A，你可以听回铃音了。

什么是回铃音？这要从更早的模拟电话时代讲起。A 给 B 打电话，B 的话机会振铃，同时 A 的话机回放回铃音（嘟嘟声），这样，A 就知道 B 的话机振铃了。

所以，回铃音是模拟电话时代的事情，它的作用就是给 A 一个提示，对方的电话正在振铃，这样，A 就可以放心地等待 B 接电话。当然，随着技术的进步和时代的发展，回铃音也在进步，典型地，除了嘟嘟声以外，电信运营商也会利用这段等待的时间给用户放一些音乐，甚至是广告，反正闲着也是闲着。这些音乐或广告就称为彩铃，这些是在 A 与 B 正式通话前播放的，因此不对 A 收费。但是由于彩铃也是资源，因此运营商可能会对 B 收费（B 放音乐提高自己的逼格，或者放广告提升自己的形象甚至获取商业利益，收费也是有道理的）。

到了 SIP 时代，就需要在 SIP 中描述这些回铃音或彩铃，这些回铃音或彩铃是真正的声音数据，称为媒体（Media）。但为了将它与真正的媒体（即 A 与 B 真正的通话数据）相区别，将其为早期媒体，即 Early Media。

SIP 的全称是（Session Initiation Protocol，即会话初始协议），它仅仅是完成会话的协商，但是实际的媒体如何传输却需要另外一个协议来协商，负责描述媒体的协议称为 SDP（Session Description Protocol），即会话描述协议。不过，SDP 寄生在 SIP 中，典型地，它寄生在 INVITE 消息和 183 消息中。当 A 向 B 发起呼叫时，它需要把自己的 SDP 放到 INVITE 消息中发给 B，同时，B 在 183 消息中放入自己的 SDP，回送给 A。当双方都知道对方的 SDP 后，真正的媒体数据就可以传输了，这时，A 才听到 B 的 Early Media。

为了帮助大家理解 SDP，我们再进一步。假设 B 接听了电话，B 会响应 200 OK 消息，该消息也带了 SDP（可能跟 183 中的相同也可能不同），用于建立 A 与 B 真正的通话，毕竟 A 给 B 打电话是为了跟 B 通话，而不是为了听 Early Media 中的音乐或广告。

所以，不管是 183 中的 SDP，还是 200 OK 中的 SDP，都是为了建立媒体服务的。区别只是一个叫 Early Media，另一个叫 Media。其实 Early Media 和 Media 差别除了一个是广告一个是真正的 B 的声音外，差别也不大，如果说差别的话，那就是运营商的收费方面有差别，因为 Early Media 是不收费的。

也许有人开始嫌我罗嗦了，但是，我确信，有些人真的不是从根本上理解我上面说的这些，包括一些开始嫌我罗嗦的人。不信，耐心点接着往下看。

好吧，183 和 200 好像有点懂了，那 180 和 183 又有什么区别呢？

180 和 183 之间就差个 3。你说我开玩笑？好吧，这些狗屁都是 RFC 定义的，RFC 就没定义他们的区别！

不过，RFC 划出道来大家就要走啊，因此，大家都按照自己的理解实现了 180 和 183。

现在最流行的实现方式是：180 不带 SDP，183 带 SDP。FreeSWITCH 也遵守这种约定。

所以，180 与 183 的区别不是 3，也不是其它的，关键是看它们带不带 SDP，即 180 也可以带 SDP，183 也可以不带 SDP。为了防止思维混乱，我们下面认为 180 不带 SDP，而 183 带 SDP。如上面所说，这些符合绝大多数人的思维。

那么，带 SDP 的 183 我们上面讲过了，180 不带 SDP 有什么用呢？

我们上面不是说到时代进步了吗？SIP 终端属于智能终端，比以前的模拟电话可先进多了。其中的一点就是它能区别 180 和 183。如果 A 的 SIP 终端收到 183，它就协商媒体，将 B 端发过来的 Early Media 在自己的扬声器里放出来；但如果收到的是 180，没有 SDP 就没法协商媒体，因此，B 就没法给 A 发 Early Media 了。怎么办，总不能让主叫用户干等着啊，所以，A 的话机在这种情况下能自己产生一个回铃音，或任何用户在 A 话机上设置的音乐。

好吧，到这里，不管你有没有理解，反正我把该说的都说了。

那么，言归正传，如何在 FreeSWITCH 中配置回 180 还是 183 呢？

FreeSWITCH 是一个多功能的 SIP 服务器，在此，为了简单起见，我们先把它当成一个普通的 SIP UA。比方说，它就是 B。

在 FreeSWITCH 内部，FreeSWITCH 的行为靠一些称为 Application 的功能函数控制的。当一个呼叫到来时，FreeSWITCH 会查找拨号计划（Dialplan）来决定执行哪些 Application。如，下面的 Dialplan，当一个呼叫到来时，它首先执行 Answer，给对方回 200 OK，然后执行 playback 给对方放一段声音（从声音文件中读取），然后挂机。

```
<action application="answer"/>
<action application="playback" data="/tmp/hello.wav"/>
<action application="hangup"/>
```

如果你自己测试这段 Dialplan，就会发现，FreeSWITCH 不会回 180 也不会回 183，而是直接回 200。这里，answer 相当于 B 摘机应答，在 SIP 中就直接回 200。

当然，『纸上得来终觉浅，绝知此事要躬行。』至于真是这样你还得自己去试，学习 SIP 最好的办法不是看 RFC，而是照着我说的这些（以及《FreeSWITCH 权威指南上的例子》）自己抓包去看。如果试着不对的话，也不要怪我，也许是你的 Dialplan 里边东西太多，在这几个 Application 前执行了你不知道的其它的 Application。

好吧，那怎么让它回 183 呢？

在上面的 Dialplan 中把 answer 那一行去掉，就回 183 了。

为什么呢？

因为'playback'的作用是向 A 播放一段声音，但，在 B 向 A 发送声音前要建立媒体通道。如果有'answer'，FreeSWITCH 会发送 200 OK，带 SDP 建立媒体通道。如果没有'answer'，那么 FreeSWITCH 就会发送 183，带 SDP 建立媒体通道，而这时，hello.wav 的媒体内容就成了 Early Media。

所以，送不送 183 就看你在 answer 前还是 answer 后执行 playback。

那么 180 呢？也很简单，那就是在发送 180 前执行一个'ring_ready'，即：

```
<action application="ring_ready"/>
<action application="answer"/>
<action application="playback" data="/tmp/hello.wav"/>
<action application="hangup"/>
```

在上面的例子中，如果你抓包，就可以看到 180，但你很可能听不到回铃音。原因很简单，'answer'执行的太快了。尝试在'ring_ready'和'answer'之间停顿一下，就可以听到回铃音了。下面例子中的'sleep'可以在发送完 180 后暂停 2 秒钟（2000 毫秒）再发送 200 OK：

```
<action application="ring_ready"/>
<action application="sleep" data="2000"/>
<action application="answer"/>
```

我们已经知道怎么让 FreeSWITCH 发 180 还是 183 了，问题解决了吗？

显然没有，我知道你在实际应用中没这么简单。我们在这里把 FreeSWITCH 当成了 B，但实际你希望 FreeSWITCH 是 FreeSWITCH，B 是 B。什么意思呢？FreeSWITCH 实际上是一个 B2BUA，它是一个中间人，你希望的拓扑结构是这样的：

A <-----> FreeSWITCH <-----> B

哎呀呀，你看你看，我第一句话说中了吧，针对你提的问题，答案远没你想象的简单。现在，我们需要列出（猜出）你想要的各种情况了。

好吧，就算这回我猜中了，继续往下说。

中间人也是不好当的，因为，他可以有 N 种不同的策略。至于使用哪种策略，看 B 的心情，也要看 FreeSWITCH 的心情。啊，说白了，又是没有标准答案。所以你还是要把你想要的告诉我。如果你不告诉我，我就得猜，或者把所有 N 种可能的情况都告诉你。

首先，我们先看一种熟悉的情况。FreeSWITCH 可以假装它就是 B，这样，配置方法跟上面讲的基本一样，只是它在假装后还要假戏真做，要用'bridge'这个 Application 再去呼叫 B，并把电话接通。

```
<action application="ring_ready"/>
<action application="sleep" data="2000"/>
<action application="answer"/>
<action application="playback" data="/tmp/hello.wav"/>
<action application="bridge" data="user/B"/>
```

所以在上面的配置中，至于是回 180 还是 183，配置方式跟上面讲的一模一样，就没必要多说了。

其次，FreeSWITCH 心情好，想听听 B 的意见。如果它即不执行`ring_ready`，也不执行`answer`，而是直接用`bridge`去呼叫 B。

```
<action application="bridge" data="user/B"/>
```

这种情况其实也简单，那就是，如果 B 向 FreeSWITCH 回复 180，FreeSWITCH 就向 A 回 180；如果 B 回 183，FreeSWITCH 就向 A 回 183。这种情况其实就相当于 FreeSWITCH 不存在，所有消息都是透明的。（不过，要记住：FreeSWITCH 是一个 B2BUA，即，它是一个中间人，它不会直接拿 B 回给它的 180 或 183 消息『转』给 A，而是自己新产生了一个 180 或 183 消息回给 A。当然，也许你不关心这个，但你说得越不清楚，我越累啊，要不然人家还会说我的回答不严谨呢。或者，万一我猜错你问的意思呢？）

再次，FreeSWITCH 跟 B 这两天不大对付，什么事情都拧把。B 回 180，FreeSWITCH 就回 183，B 回 183，FreeSWITCH 就回 180。

好吧，看起来是越来越复杂了。又是两种情况。

先看 B 回 180 的情况。FreeSWITCH 要想给 A 回一个 183，由于 B 的 180 中不带媒体，FreeSWITCH 就要『造』一个媒体出来，因此，它想了一种办法，在 bridge 之前造一个媒体：

```
<action application="set" data="ringback=/tmp/ring.wav"/>
<action application="bridge" data="user/B"/>
```

由于在执行**bridge**之前还没有 B，因此 FreeSWITCH 不知道什么时候 B 回 180 还是 183。通过在 bridge 之前使用 set 设置一个变量 (ringback)，实际上相当于 FreeSWITCH 给 bridge 下了一个套，到了 bridge 阶段，不管你什么时候 B 回 180，FreeSWITCH 都会向 A 播放事先『造』好的回铃音 ring.wav。当然，FreeSWITCH 要向 A 发送媒体前要先用 183 建立媒体通道，这就完成了 180 到 183 的转换。

所以，这也是 FreeSWITCH 设计精巧之处——同是一个 bridge，通过一个 ringback 变量改变了它的行为。

再看 183 变 180 的情况。

如果 B 向 FreeSWITCH 回了 183，FreeSWITCH 要向 A 回 180，那就不能把媒体信息送给 A。所以，实现也很简单，还是一个简单的 bridge，只是，把 B 送来的 Early Media 忽略掉就行了：

```
<action application="ring_ready"/>
<action application="bridge" data="{ignore_early_media=true}user/B"/>
```

跟 set 不同。set 是一个 Application，它作用于当前的 Channel，即 A 那一个 Channel (那时候还没有 B)。而`{ignore_early_media=true}`这种语法，在建立 B 端的 Channel 的同时，将`ignore_early_media`作用于 B。再强调一次，FreeSWITCH 是一个 B2BUA，因此 A 跟 B 间的通话要产生两个 Channel，即，所谓的 a-leg 和 b-leg。

在建立 B 通道的时候，`ignore_early_media`也是给**bridge**下了一个套。即，不管什么时候 B 回了 183，忽略它。由于我们选择了忽略，因此，为了让 A 仍能听到回铃音，我们用**ring_ready**在**bridge**前送一个 180。严格来说，它不是 183 变 180，因为 FreeSWITCH 收到 183 前就已经送出了 180，但是，如果你不趴在 FreeSWITCH 内部看，谁知道什么时候变得呢？

N 种情况讲了 N 种了，永远都会有 N+1。既然 FreeSWITCH 位于中间，那它能不能把 B 发过来的广告（彩铃）换成它自己的广告呢？能是能，但我不教你怎么做。不过，不幸的是，如果你不是特别笨的话，我上面已经教会你了……

读到这里，你认为这个问题好答吗？

1.13 从 3·15 看电话号码透传

杜金房/2015.03.17

2015 年的 3·15 晚会，报出了好多通信业违规群呼和透传电话号码的事。关于 3·15 的政治问题我们就不说了，说了也没意思，我们聊点技术方面的。

『透传』这个大众并不熟悉的专业术语，仿佛随着 3·15 一夜之间就普及了。

新闻稿上说：『为了让骚扰电话更加具有迷惑性，增加用户识别难度，金伦科技还会使出更厉害的手段，帮助呼叫中心随意显示主叫号码，这种技术，在这一行业被称作「透传」……』

『XX 公司展示了这一神奇的技术，轻而易举地在记者手机上显示了一个实际并不存在的电话号码』。

『同样，XX 也将记者随意提出的一个号码，轻松地拨打在了手机上。』

『……』

这都是些好高深的技术啊！

不懂的人打眼一看，哟，这么厉害的一个技术，如果掌握在坏人手里，这不天下大乱了吗？

其实『透传』真不是什么厉害技术，更不是什么洪水猛兽。透传就是在被叫的电话或手机上显示任意的主叫号码，包括所谓的 110。这一点，其实谁都可以做到。有合适的落地网关，在 FreeSWITCH 中就可以轻而易举地做到，如：

```
originate {origination_caller_id_number=110}sofia/gateway/gw1/138xxxxxxxx &playback(test.wav)
```

那么，如此说来，不真天下大乱了吗？

当然不是。其实主叫号码的显示跟写信是一样的。这些年写信少了，但大部分人至少还都写过信吧？（没写过信的人想想快递也行。）写信的时候，上面是收件人地址，下面是发件人地址。收件人地址当然要写正确，那发件人地址是不是想写什么就写什么？其实这里的发件人地址就相当于电话中的主叫号码，你想送什么就送什么。

电子邮件也一样。在发送电子邮件的时候，发件人的邮箱地址叫做 Envelope Address，也就是相当于信封上的地址，也是随便你想写什么就写什么的。所以，垃圾邮件泛滥就是这个原因。

普通信件要经过邮局，投递到收件人手中。一般来说，邮局是不会审核你的发件人地址的。而且你随便把信投入哪一个邮筒也没人管。所以，写匿名信或者伪造信件是很容易的事。虽然收件人可能通过笔迹等看出是伪造的信件，但追查发件人却是几乎不可能的。

电子邮件相对来说管得要严一点，所有的电子邮件提供商都不希望你往外发送垃圾邮件，所以，一般在你发送时要对发件人进行权限验证（一般是密码验证）。当然，对发件人验证是近十来年的事了，在十几年前，发邮件几乎没有验证的，随便发。

光在发件的服务器上验证也不行，有的人可以直接将电子邮件发送到收件人的服务器上。如果收件人的服务器不想被垃圾邮件骚扰，那就要采用一些手段如检查发件人的 IP 或域名。但这个也很难防止垃圾邮件。

电话系统也类似，你拿起电话，可以打通几乎任何人的电话。所以，骚扰电话是畅通无阻的。现在，有些手机里有一些功能，能使用黑名单过滤掉一些已知的骚扰电话。因此，打骚扰电话的人一个最简单的策略就是经常换电话号码，以提高电话的接通率。

电话从主叫用户到被叫用户，要经过很多中间的交换机。在被叫端交换机上检查主叫是否合法一般来说是不现实的。因此，为了防止恶意显示主叫号码，对主叫号码的合法性检查就需要在主叫侧进行，一般来说，是在最靠近主叫用户的交换机上。

对于普通的模拟电话来讲，是无法改变主叫号码的，因为电话线就唯一决定了电话号码。而当电话进入数字时代，尤其是一些大的呼叫中心都使用数字中继或 IP 中继（SIP 线路），随意显示电话号码就变得容易了。主叫用户只需要在发起呼叫时把电话信令中的主叫号码填成自己心中想要的号码即可。是的，所谓的透传就是这么简单！

当然，为了防止天下大乱，在主叫侧的交换机上是对主叫号码进行检查的，也就是说，运营商的交换机上实际上有一个检查员。如果检查员睁一只眼闭一只眼，你的主叫号码就过去了，送到其它的交换机上以后从此畅通无阻（少数时候也有阻，即所谓的主叫号码传送规范，我们就不深入讨论了）。如果检查员比较负责，那么，你发的主叫号码就无效了，检查员会把它替换成你真正的电话号码。由于绝大多数检查员都很认真负责，因此到现在没有天下大乱。

所以，检查员相当于一个开关，能否『透传』完全是检查员说了算，而不是什么厂家开发了什么先进的技术。简单是一派胡言！

那么，这个开关是怎么打开的呢？其实这个谁都知道。提供技术支持的厂商固然有利益驱动，但如果不是跟检查员勾结，再厉害的技术能过得了第一道坎吗？

那么，是不是就应该所有人都遵守规则，永远都不需要透传呢？当然不是。透传有透传的用处。在需要的时候，就应该有透传。如下图：

客户来电 ——PSTN——> 公司总机 ——PSTN——> 转出到个人手机

在上图这种情况下，如果客户的来电到了公司的总机上。总机发现被叫不在办公室，再通过 PSTN 转到被叫的手机上。在这种情况下，如果电话号码不能透传，那么，在被叫的手机上就显示公司总机号，但是被叫真正想看到的是客户实际的电话号码。

所以，这是一个实实在在的合法的需求。但是，95%（瞎猜的数字）以上的公司都做不到这一点。为什么？运营商不让你透传。

所以，问题的实质是，需要并且合法使用这种功能的人得不到应有的服务。坏人却总是轻而易举地能做到。而且，由于这些非法的使用，使得守法的人越来越难得到这种合法的业务。

无论如何，我今天想说的是-不要把『透传』变成一个贬义词。另外，媒体还说：『某某科技专门研发了一套系统，……，每天群呼上千万个。』技术无罪，不要误导了群众，把重点放到那些随意贩卖个人信息的那些人吧。

回到我们的技术原点，在 FreeSWITCH 中，默认是禁止透传的。但如果你想让『检查员』退休，『透传』你想要的电话号码，可以把用户配置文件中类似如下的两行删掉或注释掉：

```
<variable name="effective_caller_id_name" value="Extension 1009"/>
<variable name="effective_caller_id_number" value="1009"/>
```

好了，放过技术吧，技术无罪。看你怎么用。

最后，学会了但不要做坏事哟。《FreeSWITCH 权威指南》上还有对主叫号码更详细的讨论。

1.14 FreeSWITCH 帮我找到了工作

杜金房/2010.10

今天，跟 FreeSWITCH 邮件列表中的一位网友聊天，忽然想到了它曾发到列表中的一则故事。感悟颇多。

该网友是一个盲人，靠屏幕阅读软件看电脑，用语音识别软件转换成文字跟我聊天 (Google Talk)。我一直有个疑问，我用 FreeSWITCH 毛算也有三年了，在学习和使用中经常遇到一些问题，我手眼健全都需要很长时间解决，作为一个盲人，他是怎么做到的呢？

从聊天的语言来看，跟正常人别无二致，而且，可能是他使用语音识别的缘故，反应起来甚至比正常人都快。

当然，跟他聊天并没能完全解答我的疑问。我想，如果有机会去阿尔及利亚，我一定找机会就见见他。

在工作的生活中，每当忙了、累了，我就想到这则故事。今天，我使得他同意，把全文翻译成中文，放到这里。需要指出，由于原文可能是用语音识别写的，因此有些句子不太顺，我尽量按我的理解翻译了。

邮件列表中的同学们，大家好，

我叫 Meftah Tayeb，来自阿尔及利亚。我双目失明，但自从 1.0.1 时代（版本）就开始使用 FreeSWITCH。

我最开始用的是 Asterisk，在我的生命中，它是一款非常奇怪的 VoIP 软件。感谢 Miconda 在 IRC 上的 #openser 频道，把我从 Asterisk『重定向』到 FreeSWITCH，那是 2008 年的事情。

我于 2008 年 12 月开始学习 VoIP 的基础知识。感谢 IRC 上 #freeswitch 频道上的伙伴们对我的帮助。最开始有 anthm，Michael S Collin，brian (BKW)，感谢你们以及其它所有的好人。

在 2009 年 8 月，阿尔及利亚电信以及政府开始屏蔽 SIP 流。我并不使用 SIP 来做 VoIP 生意，但是，老实说，我只是想用 SIP 连接到每周一次的 FreeSWITCH 公共电话会议，在会议里，大家都讨论 VoIP。我开始向我当地政府（或电信，原文是 local city）抱怨此事，但没人理我。我想直接

进入总经理办公室，因为没有 SIP 我无法使用我的 PC。我找到了总经理秘书，他接待了我，并听取了我的抱怨：『你们为什么屏蔽 SIP？』。然后他问我：『那你为什么一定需要 SIP？你想无证经营 VoIP 业务吗？』我把我的实际情况告诉他。他说：『好吧，没问题，我会给你开通 SIP，但你下周一定要到这里来。』。我说：『OK，那没问题』。

然后，我回到家，看到了新的东西：

1. 一个静态 IP 地址，绑定在我的 ADSL 帐户上
2. 完全开放的 SIP

OK，在下个周一我到了总经理办公室，见到了总经理秘书。使我奇怪的是，他问我：『你有工作吗？』我说没有。他说：『你肯定有，告诉我』，我还是说没有。然后他说：『你在为阿尔及利亚电信工作』！；）

他给了我一个惊喜！现在，他给了我一个好工作、免费的房子、以及免费的护理，还有司机。

所以，在这里谢谢 FreeSWITCH 项目，特别是 FreeSWITCH 的主人以及所有贡献者。

谢谢。

邮件原文在此 (<http://lists.freeswitch.org/pipermail/freeswitch-users/2010-June/058789.html>)，以下是全文拷贝：

```
[Freeswitch-users] get your job aguinst freeswitch

Meftah Tayeb tayeb.meftah at gmail.com
Sun Jun 6 11:34:09 PDT 2010
Previous message: [Freeswitch-users] Hi
Next message: [Freeswitch-users] get your job aguinst freeswitch
Messages sorted by: [ date ] [ thread ] [ subject ] [ author ]

hello list,
i am meftah tayeb, a blind person from algeria that was using freeswitch
sunse 1.0.1 release
i started firstly with asterisk, that was the very strangett voip
application in my life
and thank to miconda that redirected me to freeswitch, from asterisk in
#openser in 2008
i started learning voip basic in dec 2008
thank to the #freeswitch folk that teached me all this, including
firstly anthm, Michael S Collin, brian (BKW), sekil the nice GUI and all
other
in ogust 2009, algeria telecom and the algeria gouvernmant started
blocking sip traffic
i was not using it for voip business, but, honestly, just to connect to
the public freeswitch conference and the weekly voip users conference
i start a complain in my local city, no reply from AT
```

i decided to go to the general office, because i can't use my PC without SIP
i got the general directore secrutary
ok, he receyved me and heare me saying why you are blocking sip?
so he asked me
why you need sip?
do you do voip business without autorisation?
i explaned to him my actual situation and he say: ok, no problem i will
open the sip for you, but conditionaly
the next mondey you will be here
i say ok no problem
so, i returned to my home and i see something new:
1. a static ip address linked to my ADSL account
2. sip completly open
ok, next monday i was in the general office and i meet the gebnral
directotore surprisingly
he asked me:
do you have a job?
i say no
but he say yes, you have, tel me
i say no aguin
and he say you work for algeria telecom
;)
he surprised me with this
now, he gave me a:
good job
free house
free care with driver
so please say thank to the freeswitch project especiaiy the owner and
try to donate to him a much a pocible
thank you

--
Meftah Tayeb
algérie télécom SPA
phone: +2132176XXXX
phone (INUM): +88351000128XXXX
mobile : +21366034XXXX
mobile (INUM: +88351000128XXXX
<http://www.algerietelecom.dz>

第二章 ClueCon

ClueCon 是 VoIP 开发者的年度盛会，每年一届在美国芝加哥举行。

Cluecon 是由 FreeSWITCH 核心开发者主办的，好多人都以为是先有 FreeSWITCH 再有 ClueCon，其实不然。

2.1 Cluecon 2011 小记

杜金房/2011.8

搞 FreeSWITCH 好几年了，一直盼望能去 ClueCon 见识见识。今年，终于有了一个机会。

准备材料，办签证，虽然罗嗦，但还是顺利地通过了。保险起见，提前两天出发。所以，7号下午从北京出发，到了芝加哥还是7号下午，到酒店已经是晚上了。8号随便走了走，玩了一天。由于时差的关系，第二个晚上还是睡地比较凌乱，因此9号早上起的有些晚。又因为被人指错了路，所以到 ClueCon 会场的时候已经是九点多了。

在登记的时候遇到了 Michael Collions，他是大会的主要组织者，长得很高大也很帅。而且他的发音很纯正，因此听起来没有任何困难。

进到会场后发现几百人的会场坐无虚席，我在最后面靠墙的地方找了个位置坐下。正好没耽误 Anthony Minessale 的演讲。

他讲到 FreeSWITCH 的一些新特性，及最近做的一些优化。由于 sofia 是单线程的，所以做并发一直有问题。最近的更新通过使用 FreeSWITCH 的内部信令队列使得 sofia 已经能承受 sipp 1000 cps 的话务，并能达到 30,000 个 channel (call legs)。Anthony 还幽默地说：『Dont try this at home.』。由于我坐在最后排，没看清他用了一个什么服务器，只是隐约看到 status 命令的输出数字都很长¹。另外，与 Michael 比起来，听他说话就比较费劲了。一是他说得比较快，声音不洪亮；二是他好像也不照顾我这从中国来的，讲得慢一点：(。

¹后来，Ken Rice 在 2012 在邮件列表中证实了：Tony demonstrated FreeSWITCH running 1000cps 30sec call duration with media for a total of 30K concurrent calls @ ClueCon last year (just a few weeks shy of a year ago)。参见：<http://lists.freeswitch.org/pipermail/freeswitch-users/2012-July/086156.html> 。

偷眼看了一下会场，所有桌子上都放了好多插排，有的人也自带了各种各样的插座转换器，总之所有笔记本（除了像我坐在最靠墙一排的）都能得到电源。Mac 居多，使用其它本本的少数人用 Linux，有的用 Windows。

午餐是自助的，看起来应该挺贵，但没什么可选的，只好将就着选了点菜叶和牛排。不过看到外国人吃的都很 High。

中午休息的时候我看到 Giovanni，他是 mod_skypopen 模块的作者，意大利人。虽然第一次见，但由于以前在电子邮件和 IRC 上交流很多，因此一见如故。正好他旁边有个位子，我就坐到第一排了。

大牛们的演讲都很精彩（当然也有不精彩的），确实学到不少东西。比如 Mathieu René 他们除了做了 mod_rtmp，还把 FreeSWITCH 移植到 iPhone 上了；2600hz whistel 的新发展等等。

上新浪微博给大家直播了一下。由于现场人比较多，再加上手机等需要上网，因此也经常有网络不通的情况。另外，国内是晚上，因此，也没大有人回复，直播的热情自然就少了好多。

晚上是一个欢迎的 Party，大部分人都去了。我跟美国、印度、南非人共四个人打了两局保龄球，成绩还不错，每次都是第三名，要知道，以前仅仅在烟台打过一次啊。当然最后累得手都疼了。

遇到一个中国人，在美国上大学毕业后在纽约一个小公司工作。好几天没说中国话了，因此猛聊了一阵子。他老板是华裔，但不大会说中国话。

时间总是过得很快，半夜回去上上网很快就睡了。

噢，差点忘了说，由于我没有预订酒店房间，Michael 帮忙找了个室友，William Dale。他自己有公司，做了 labortimetracker，一个网上考勤打卡系统，在美国市场还不错。因此我们俩还就要不要把他们服务拓展到中国聊了好一阵子。

第二天的会议没什么值得多说的。通过昨天晚上的 Party，大家也都比较熟悉了，因此打打招呼，随便聊聊什么的。

好不容易找 Anthony 聊了半个多小时。找他聊天真是比较困难，因为大家都在找他说话，而且我想独占他一点时间，因为怕跟他交流有困难，所以想慢慢聊把问题说得清楚些。还不错，我们聊了一些过去的事情，请教了一些技术问题，也聊了些 Google Talk、Skype 以及 Microsoft，还有 FreeSWITCH 在中国的发展等等。有大约 10% 的东西听不懂，都哈哈过去了。

晚上跟 Michael 等 7、8 个人一块吃了晚餐。AA 制，花了 30 多刀，吃了个不怎么好吃的意大利面。话说在这边吃东西还是挺贵的，而且还得交税，还得给小费。

吃完饭以后又回去跟他们聊了会，Anthony 一伙人则弄了吉他什么的一堆乐器在那儿娱乐。后来我累得实在撑不住了。回房间后倒头就睡了。

第三天，也是最后一天了。早上四点多就醒了，再也睡不着，准备演讲的幻灯片。因为之前跟 Michael 打过招呼，午饭后他给安排了一个表现的机会。第一次用英语上去讲，而且台下都是界的大牛们，好紧张。心里默念了好多遍的台词都忘了说。不过还好吧，比我预想的讲得长了点。完了后看大家反应应该还算不错。尤其是 Giovanni，说了好几个 Very Good。也有人过来要名片，大概是因为我提到中国有 1,339,724,852 这么多人口吧。

然后跟 2600Hz 的 Darren 简单聊了会，他说我做得很东西跟他们都很像。我是啊，英雄所见略同嘛。由于他要赶飞机，只好以后再联系。

时间过得很快，三天的会议马上就结束了。不爽的是，会上抽奖有一个 MacPro，两个 iPad，若干 snom 及其它电话，始终没有叫到我的号。

会后大家都走了，我不急着赶路，磨蹭了一会。看到 Mike Jerris，请教了一些 GDB 的问题，感觉豁然开朗。好多年，没有人手把手给我讲东西了。

后来闲着没事帮他们收拾了一下东西，就打道回府了。

当然期间也跟 Brian West 打了招呼，不知道怎么形容，反正感觉他很帅很有意思。

很遗憾的是光顾聊天和兴奋了，忘了拿出手机跟他们挨个合个影。

回想起来整个会议还是安排得挺不错的。他们组织和服务的没几个人，却秩序井然。大家也随意把笔记本什么的丢在桌上出去说话，好像也不怕丢失。另外感觉这几天英语也有所进步，至少是说得比以前快了:)，而且，感觉有些在台上演讲的人发音还不如我好呢。尤其像我搞了这么多年英语教育，感受还是很深的。其实学习语言真的不难，当你听的说的甚至想的都是英语时，自然就进步了。所以，环境是很重要的。而我认为第二重要的是别人的鼓励—在国内，觉得自己说的不好就不敢说；而在美国，甚至你只会说几个单词别人就说『哇，你英语说的不错』。是吧？想想是一样的道理，我们在中国遇到有老外说几个『谢谢』，『你好』之类的也会夸奖他们。

好像扯远了。在飞机上写下这些，算是个小结吧。

2.2 ClueCon 2012

杜金房/2012.8 原文名称《八月旅行记》

2012，8月初，借 ClueCon 会议出去转了一圈，小有所感。

本来从很早就打算，但，还是在机票上出了问题，定好的行程只好相应改变。

从北京出发，第一站是华盛顿。安检、过海关都很顺利，比去年快好多。在机场碰到奥巴马，还一块合了个影。

然而应该在两小时后出发去芝加哥的下一班班机始终不能到港。延迟、取消，美联航也用同样的招数招待旅客。时已至午夜，大部分人顺从地改签了第二天晚上的班机。但我等不了那么久，因为 ClueCon 在会议开始前一天晚上要举行一个热身晚宴，我不想错过。与其他几个同行的中国人商量，他们则更急，第二天一早就有会。后来经一再与航空公司客服磨蹭，我们 5 个人中有两个人转签了第二天最早的航班，其余三人则给了 Stand By 的登机牌。Stand By 的意思就是，如果有人在午夜到早上 6 点的时间突然取消机票，我们就有机会登机，或者，等待有的人早上睡过了头。不管怎么讲，这都算是最好的结果。



图 2.1: 华盛顿机场跟奥巴马合影

由于航空公司以天气原因为由（实际上两边天气都很好，我始终觉得是上一班飞机不知出了什么问题）拒绝提供住宿，又由于我们离登机也只有 6 个小时时间，几个人一商量决定有两个留在机场，其他三人打车去市里转一圈。

司机是巴基斯坦人，说话跟印度人口音差不多，很多都听不懂，当然，他听我们的英语也有困难，呵呵。机场离市区也不算近，转了三个来小时，路过国会大厦，白宫和五角大楼，其实晚上没什么好看的，但也算是『到此一游』，『不虚此行』吧。

值得一提的是，就在白宫正对面有一个小帐篷，有一个人在看守着，旁边的牌子上写着：『WHITE HOUSE 24 A DAY ANTINUCLEAR PEACE VIGIL SINCE 1981 MAINTAINED BY CONCEPCION THOMAS …』。大意是反核武器。与小伙简单聊几句，他倒是很健谈，口音很纯正，只是大部分都是哲学和逻辑，我也不记得多少。印象最深的就是他们有 10 多个人轮流到这里值班，在白宫正对面，坚持了 30 多年了²。

回到机场很快就到登机的时间了，又多过了一次安检。还好，最后我们三个 Stand By 全部转正，路上一顿好睡，飞到芝加哥是早上 8 点。乘地铁到 Hosteling International Chicago，由于去年到过一次，因此很顺利，只是 Check In 的时候被告知前一晚由于我没到，订单已被取消，下一次 Check In 的时间是下午 3 点半。天啊，我告诉他们我必须睡觉，后来他们给我一个临时卡，我可以上二楼大厅里休息。

一觉醒来，已是下午 5 点半了，没想到在沙发上睡了一整天！匆匆办理了 Check In，便急忙赶去晚宴。虽说有地图，但还是有点迷路，茫然间遇到了 mod_skypopen 和 mod_gsmopen 的作者 Giovanni

² 后来查了一个还真有那么回事，参见：https://en.wikipedia.org/wiki/White_House_Peace_Vigil。



图 2.2: White House Vigil

大侠，顿时感觉找到了组织。赶到酒店，正赶上集合。

其实晚宴也不怎么隆重，就是大家吃饭，随便聊一下。其间跟 OpenSIPS 的几个哥们聊到 OpenSER 及 Kamailio，对于历史他们则只字不提。从聊天中才知道他们的核心团队原来是来自罗马尼亚的，我之前光知道是欧洲，呵呵。

吃完饭回青旅，写演示文稿—由于最近事情有些杂乱，飞机也不正点，因此所有这些工作就压到了最后。

期间看到邮件，Anthony 还提交了一些代码，并且打上了一个我几个月前写的补丁，感觉到他甚是勤奋。

第二天按时赶到 ClueCon 会场，与几个熟悉的简单聊几句，正式的演讲便开始了。形式大约就是主题演讲，大家提问，茶余饭后大家也可以相互交流。

大家的演讲都很精彩，尤其是 Anthony 在演讲中说到『FreeSWITCH 1.2.0 版已经在一分钟前正式发布了』，真是非常有意思。怪不得他在前一晚还在孜孜不倦地提交代码。

另外，这次看到有中国人上去演讲了—来自 OpenVox 的 Belief Mo。午餐倒是没什么好吃的，随便吃了一些。晚上 OpenVox 的哥们请客，也对他们的产品加深了一些了解，聊到中国的开源软件，唏嘘不已。

晚上是 Anthony 的生日，有人准备了蛋糕，Party 就在我们开会的会场上举行。我把我的书送给他，他虽不懂中文，但看起图来很在行，指着一幅 X-Lite 的图兴奋地说，哈哈，我到现在还在用这个，新版的有 xxxx 问题……Brian 同志好像喝的不少，一直在唱卡拉OK。我就在后面写明天的演示文稿。很高兴遇到 William King(mod_vlc 作者)，学习了好多东西。

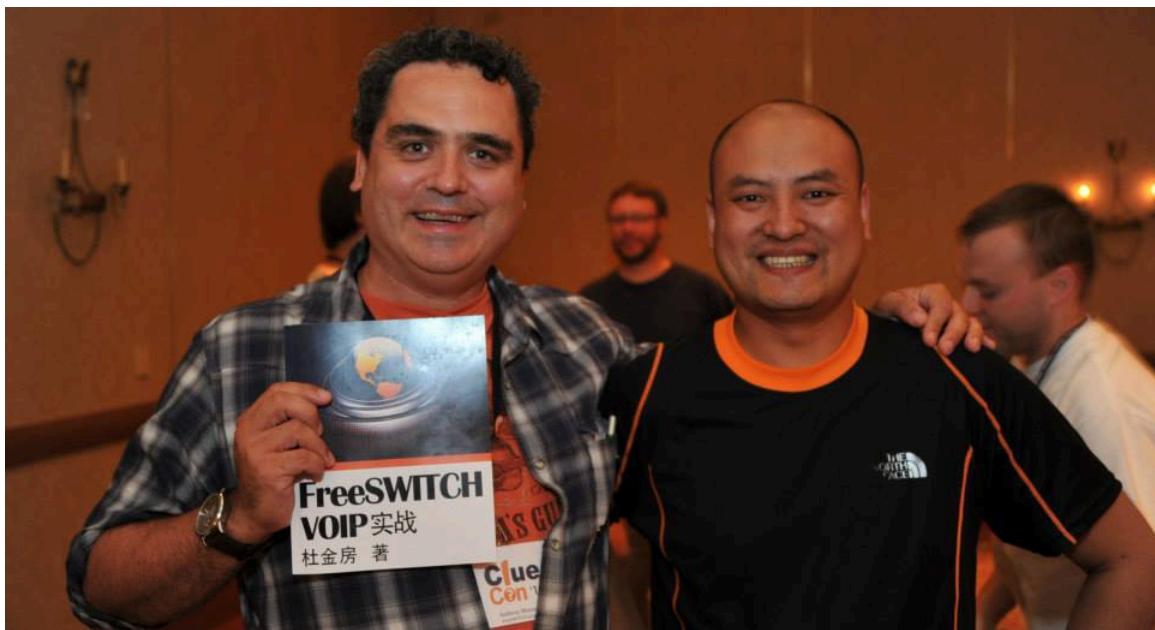


图 2.3: 本书作者与 Anthony 合影

晚上就住在开会的酒店，后来我回房间写到 2 点多，总算完成差不多。由于时差的关系，因此睡得也不算晚，第二天一早起来又修改了一下，匆匆赶去会场。就在我出去 711 买了点早点的工夫，据同桌说抽奖号码可能念到我了，后悔不已。

今天第一场演讲就是我的，过程还算顺利，英语也比去年感觉顺畅一些，只是还是自己注意到了有好多 Ur… 和 Ah…。由于演示文稿使用了 Prezi，讲起来思路很清晰，事先准备的几个 Live Demo 也都很顺利。讲完，还剩下一分半钟的提问时间，算是刚刚好。当然，今年没忘让同桌帮我照几张照片。

后来过来找我聊天的就比较多了。有一个人说他用到我用 Lua 写的 Dialer，我说我写完后就一直没用，很高兴对他能有帮助；也有人关心我们在视频方面的研究问能不能开源；Michael Jerris 说他虽然看不懂中文字，但他都知道我写的什么，因为他自己对 FreeSWITCH 太了解了；Brian 也想要一本书，问我能不能寄给他…。后来晚上有一个夏威夷的华裔哥们请我吃饭，他做 ISP，占有一部分市场，以前用 Asterisk，现在想改 FreeSWITCH，便来到大会听听。

晚饭后大家都去了一个玩的地方，我也不知道叫什么，但发现原来是管饭的，由于我们已经吃完，便直接上二楼去玩游戏，说实话，我除了上高中时打过街机的三国志外，现代的 Video Games 基本都不会玩，便选了一个叫 Terminitor 的游戏打枪（机关枪！）打到手抽筋。

说话间就到第三天了，我坚持到最后终于中了一个 Andriod Pad 大奖，今年的奖品没有 Macbook 了。

散会后我又呆了一会，大家都走的差不多了，抓住 Ken Rice（负责维护 FreeSWITCH 的分支及稳定性）给我讲了半天 FreeSWITCH 代码管理的东西，受益菲浅。

总之，大会就这么结束了。应到 200 多人，实到 200 多人。OpenVox 的兄弟们去了，亿联话机的也去了，这次更多了一些中国人的面孔，让我也不感到那么孤单……

晚上又回到 Hostelling International，省钱嘛，同时这里会遇到好多人，不会那么闷。当然第一先睡一觉，这几天实在是太累了。3 小时后起来随便吃了点饼干，煮了包从国内带去的方便面。不像去年，这次对西餐倒有些适应了，剩下的一包一直到香港到了最后一晚才解决掉，那是后话。

第二天报名参加了一个林肯公园 Walking Tour。一个 70 多岁的老奶奶带路（志愿者）。还顺路带我们去了她家，三层楼，据说都是她的，一层租出去做店铺，我们去了二楼。屋子倒不小，但是堆满了东西，花花绿绿的，显得很狭窄。她好像以前挺爱跑步或骑自行车的，家里有好多的奖牌。

后来带我们在街上转转，中午随便找小店吃饭，下午去了林肯公园，免费，这里的空气很清新。噢，对了，忘了说，我们团队有来自香港、法国、LA、德国、墨西哥、加拿大等约 10 个人，很好玩。不过，逛完公园后就剩下好像 5 个人了，我们随后一直走到海边（密歇根湖边），好大的风。然后又上到 John Hancock 中心 96 层喝了杯啤酒。高处鸟瞰芝加哥别是一番风味。

晚上回青旅一直写代码，视频录像 mp4 取得了较大的进展，录的像基本上能放出来了。又收发了些邮件，在米国用 Gmail 是相当的爽啊，除了没有墙外，速度也相当快。Clone 个 FreeSWITCH 代码也就一两分钟的事。另外在 ClueCon 上听一哥们说他们是做接入的，直接用天线架 WIFI，网速百兆，可覆盖 30 英哩。不过万事都没有完美的，上新浪微博就相当的不爽。

翌日启程回国，由香港转机。我之前一直以为北京和香港到芝加哥距离是差不多的，后来才明白地球是圆的，从北京走飞机沿北极圈飞要近好多，大约差三个小时，我还得再用三个小时从香港回京，下次记着可不这么绕了。

由于又遇到流量控制，起飞晚了，到香港已是下午 6 点。之前没能订上 Hostelling International 连锁的青旅，只好在 Hostelworld 订了。乘地铁到市区（香港站）再转公交，到旅舍已是 8 点多了。因为很难找，转了好几圈。后来发现自己竟然是在铜锣湾，离铜锣湾地铁也就几步远。

出去找饭吃，进了一家小店。点完了才发现公仔面竟然就是方便面。不过，猪扒和煎鸡蛋还是非常的好吃的，也可能是我太饿了。

回来，在路上慢慢走走，可以仔细看一看。这里街道很窄，一般也就两个车道，单行道比较多。再加上两边的楼都特别高，显得格外拥挤。后来第二天的时候我到其它地方看到也大抵如此，最宽的也不过双向 4 车道，不过，路上的车都井然有序，没看到有北京那种大堵车的感觉。

街上大都是汉字的招牌，对于刚从米国回来的人来说，看起来真有些亲切。再好好看旅舍，原来就相当于是家庭旅馆，这里并没有小区，就在路边，一楼二楼是店铺。楼道非常窄小，电梯虽然标明荷重 6 人，我看我一人在里面还差不多，最多的时候也就 5 个人，因为电梯是正方形的，怎么乘 6 个人呢？

后来看到其它地方的房子也差不多是这种格局，当然是否有富人住的小区我就知道了，总之，我住的房间是很狭小的，根本没有南北通透一说。

同屋的舍友晚上 1 点多才到，是新加坡的，刚从北京到香港。男男女女几个人一直吵到半夜，我也没睡好。不过还幸亏他们有一个人带了个万能插座适配器，我的 Mac 和 iPhone 才能充电。之前只想着去美国插座都是没问题的，就没想到香港的不一样。上了会儿网，简单查了一下攻略，才发现我乘的地铁并不是最省钱的交通方式，如果乘 A11 路公交可直达我的地方，只要 40 多元（以下都是港币），而我坐地铁单程就是 100 元。

第二天早晨起来已是 10 点了，吃饭就当午饭了。去了据说很经典的『池记云吞』，确实挺好吃，也不贵。

到地铁站买了个一日任意坐的地铁卡，第一站到旺角，不知道为什么，只是因为以前听说过这个名字。刚出地铁站就下雨，只好钻进一家商场瞎逛，什么也没买—衣服什么的暂时不需要，瓶瓶罐罐里的东西我也不感兴趣。看了一下，大概除了奢侈品外，其它的还是比大陆贵，而且后来我去超市还特意看了一下酱油的价格，去香港打酱油一说我认为有点不靠谱。

我是后来才知道机场有免费地图的，当时不知道，所以没有地图，也不敢瞎走，又乘地铁到了油麻地。在街上随便走走，冷不防走进了一个农贸市场，里面什么都有，模式跟国内相似，又有差别。到底差在哪里，也说不上来，只是记得有一家卖肉的招牌上写着『保證足秤』，当然是用繁体字写的。

虽然听不懂也不会说广东话，但听起来还是很好听的，就像年青时喜欢的粤语歌一样。另外，以前也了解一些广东话的，粤语有 6 个音调，因此变化比较多，抑扬顿挫，比较好听。而且，旧时兵家必争之地都在中原，南方的战乱比较少，因此语言文化受到的冲击比较少，便保留了好多古音（如入声字在普通话里已经没有了），对于我这个古代文化的粉丝来说，还是很感兴趣的。另外香港和台

湾也一直使用繁体字，所以保留古代的韵味都比较浓。有趣的是，虽然他们近代受到西方文化的冲击非常大，但该保留的还是顽强的保留着。甚至，海外唐人街的主要语言都是广东话，新加坡、马来西亚的汉语报纸书籍行文也都类似港台的。在海外，西人更多的把广东话 Cantonese 与普通话的 Mandarin 放到同等重要的位置。我想，一国两制绝对是对的，它是香港繁荣的基础，它包容，宏大，每年来自世界各地的游客带来了世界各地的文化，又把这里的文化传播到海外。

香港使用繁体字，如同台湾。中国大陆推行简化字已有几十年，当然，它简化了笔划，易于书写，在一定程度上减少了文盲，带来了经济的发展。但同时，字体在简化的过程中也失去了原来的意义，失去了汉字的美。繁体字是美的，这也是大多数书法家都爱写繁体书的原因。

除简繁分别外，行文也大有不同。我看到一处店铺上写着『偷窃送官究治』感到非常有意思。在大陆，虽然『官』这个词也会常常说起，但当官的自己不用，小民用到时就多含贬义。而这个牌子上，没有用到公安，没有用到警署，没有用到 Police，看起来非常古朴，庄重。也许香港人看起来稀松平常。

说起普通话，在香港人眼里看来还是有一定份量的。书店里，就有专门讲香港语言与大陆语言的区别（之所以没说广东话，我觉得这里说的是书面语，而广东话本身更是一种口语方言，跟书面话还是有很大区别的）。在现在，遇到不会普通话的人，有会说英语的会让你说英语，英语也不会的，会让你慢慢说，以便他能听懂。从我的感觉来看，完全没必要因为不会广东话而感到有问题，相反，他们反而会因为不会普通话而觉得不安。而且，我见过一个人，普通话已经说的很不错了（还有英语），还老是跟我说她的普通话不够好。

从书店出来，随便上了一个公交车，竟意想不到的来到了尖沙咀码头，也就是著名的维多利亚岗。实在是太漂亮了，宠辱皆忘，只能这么形容了。陶醉半晌，才想起来拍照片。顺着星光大道轻轻走过，每一步都像是不同的角度不同的景色。去之前也听说过，但对一个住在海边的人来讲本不是怎么向往的。而来到之后，才知道这不一样的感受。烟台也有海，但没这么深，这么蓝。尤其，没有这漂亮的云！

好东西不可多用，美景再好，不能贪图享受。时间有限，看到香港艺术博物馆就在路边，我也不想错过机会。

门票 20 港币，一点都不贵。看了好多古董，走马观花而已。明显的感觉就是这里的瓷器都比较新，都比较细腻有光泽，大约出自唐汉，宋元明清等，而很少有远古时代的物件。书法字画也看了不少，大都一带而过。最使我驻足的，是丰子凯的画，有好几个展厅，我也看了一个来小时。以前也知道他的名字，但作品没什么印象，这次一看，确实非常深刻。作品以漫画为主，草草几笔，却把人物、故事勾勒的非常生动，把主题表现的非常鲜明，看起来令人有一种深切的共鸣。印象最深的是解放战争后，丰回家后看到原先的大宅只剩残垣而作的一幅。有空一定买一本画谱来看，虽与真迹相差太远，但总可以仰读俯思，在忙碌的生活中添一份宁静。

不敢久留，因为答应了朋友去苹果店看看。乘地铁倒没多远，只是去了两家苹果店，均没有最新的 Macpro retina 版，只好空手而归。

大商场里晚饭没找到合适的地方，有个地方似乎卖北京、上海等各地小吃的，然而对我没吸引力，排队的人也多，我索性吃了麦当劳，华灯初上，站在四楼某天台看维多利亚港。后来，为了不虚



图 2.4: 丰子恺作品之一

此行，还是花了 2.5 元坐了一把天星小轮（Star Ferry），很不错，但维港的夜景没想象中的美。因此，又回到星光大道发了会儿呆以后，便打道回府了。

回到铜锣湾地区，到时代广场，已经快到晚上 10 点关门的时刻了，只是匆匆给儿子买了件玩具。

又到另外一个商场，忘了名字，好像是刚开业或搞什么活动，12: 00 才关门。溜达半天也没看到什么可以买的，最后进了一家叫『诚品』的书店，看了会书。多是中文繁体的，也有部分英文书。看到了英文版的《Steve Jobs》，而中文版的译作《贾伯斯传》。英文版要贵好多，具体价格忘记了。但繁体中文版也比大陆的简体中文版贵好几倍。

最后看到一本台湾出版的繁体中文竖排版《郎咸平說—中國即將面臨的 14 場經濟戰爭》，郎咸平说得不一定对，但是开卷有益嘛，不知道大陆有没有出版，便花了 100 大元买了回来。

第二天，6 点乘地铁赶 8 点班机回京，一切顺利。飞机上香港《星島日報》的信息量还挺大，有各种大小案件，有钓鱼岛，有王立军，有薄熙来，有 MM 公寓……

回到首都北京，虽然空气依然不好，但毕竟是家啊。

另外，附小词一首，是以记：

诉衷情 · 维多利亚港

初至维港，但见天高水阔，风清海蓝。登高一望，码头半壁连海，两岸高楼擎天，直教人流连忘返。
观熙攘旅客，穿梭游船，又有远处青山环黛，绿树含烟。海上清风拂面，天上云舒云卷，风景无边。
至若华灯初上，更有繁星点点，灯火斑斓。走星光大道，乘天星小轮，移步换景，千般妩媚，万种
风情。一时诗情涌动，因有词曰：

高楼两岸都接天，
绿树伴云眠。
一时宠辱皆忘，
信步看远山。

云卷卷，
风清清，
海蓝蓝。
沉吟之处，
即是仙境，
又是人间。

2.3 ClueCon 2014

杜金房/2014.09.28

光阴似箭，转眼，ClueCon 已经到了第十个年头了。

为了防止航班延误，我是提前两天到的。当然，为了省点钱，我住的还是芝加哥青年旅舍——Hosteling International Chicago (HIC)。不得不说，HIC 是我住过的最好的青年旅舍，没有之一。

到达 HIC 后是当天的下午，我凌乱的睡了一晚后，开始准备我在 ClueCon 上要讲的 Demo。是的，今年是我准备最糟糕的一年，可以说，我什么都没有准备。我花了几乎一整天的时间重构了一些关于视频转码的代码，最后，它终于不再那么频繁的 Crash 了。稍微松了一口气后，给 Anthony 发邮件，他回复说他们已经在会场准备了，所以，我可以过去看看。

就在我准备出发的时候发现外面下起了雨，还不小，不过，等了一会，雨就停了。HIC 离 ClueCon 的会场只有三站的地铁路程，因此，很快就到了 ClueCon 的会场。这次，ClueCon 的会场选在了洲际酒店 (InterContinental)，会场 1080i 的双投影，比以前气派了很多。当然，他们为了准备今年的音效，特地准备了好多音频及视频控制设备。

当天下午只是随便聊聊，我也没帮上什么忙，最后，跟他们一起晚餐。就餐的过程中又下起了大雨，冲着我们的小蓬子，别有一番风味。好不容易等到雨停了，他们回去可能继续准备，我便径直回酒店了。时差还没有完全倒过来，我需要好好睡一觉。

接下来的一天，可以说是 ClueCon 的第一天，也可以说是一个非正式的开始。中午开始签到，下午是一个黑客马拉松 (Hackathon)。其它也没什么，就是大家自己在写程序，随便聊天什么的。Anthony 在忙着跟 OpenBTS 的人做一些对接，我则忙着我的视频 Demo。其实，这一天跟 OpenSIPS Sumit 有点小冲突，其实我也挺想参加 OpenSIPS 的活动的。

当天，在会场遇到了 James Body。他是从英国来的，他们公司叫 TruPhone，做一种电话服务，可以在一张 SIM 卡上绑定多个国家或地区的手机号（如英国、美国、香港、德国等），在这些国家之

间打电话只收本地通话费。好像他们从最早就赞助 Anthony 的团队和 ClueCon。今年，他们赞助的是一场 Dangerous Demo 活动。去年的时候，James 见我的 Demo 有趣，特别给我发了一个 Raspberry Pi，今年，则嘱咐我一定来个刺激的 Demo。

其实我真的没有理解 Dangerous Demo 的含义，现在来看，大致就是与 FreeSWITCH 有关，越帅越出人意料越好。我没有别的选择，准备把我的视频 Demo 拆成两半，一部做为 Dangerous Demo，另一半则作为正常的演讲。当然，第一天光顾跟他们聊天，其实进展不大，主要的工作还是在晚上，反正也睡不着。

接下来的一天才是 ClueCon 真正的开始。第一个演讲的当然是 Anthony。他主要讲了 FreeSWITCH 最近的一些更新。当然，最重要的就是 FreeSWITCH 1.4 版的发布，其中包含了对 WebRTC 的支持——SIP over Websocket 以及 Verto。其中，Verto 的演示相当帅，不管是重新刷新网页，还是 FreeSWITCH Crash 后重启，都能自动恢复通话！

后面其他人的演讲大都是应用 FreeSWITCH 的成功案例，我们很高兴看到越来越多的人利用 FreeSWITCH 成功部署了自己的通信平台，大部分都是所谓的『云』。

当然，在听他们讲的同时，我还是在偷偷地准备我的 Demo。晚上散场以后，有一个 Party，我没有参加，太困了。回去睡了一觉，仍然感到不解乏，但我还是得在凌晨两点起来准备我的 Demo。还好，到早上 6 点多的时候我基本都测试通过了，然后匆匆睡了一个小时去会场。

第二天，大家的演讲也都很不错，但我不记得许多了，当然有一部分原因是我在下面开小差准备下午演讲的 Presentation。Dangerous Demo 是在下午第一个开始的。Anthony 自然是第一个，他演示了 FreeSWITCH 对立体声的支持以及崩溃恢复。一切环节近乎完美，不过，也许是因为太完美了，也许是因为评奖的人没有理解，他没有获奖，对此，他还小有些不高兴呢。

轮到我上台的时候，著名的墨菲法则起作用了——我遇到了非常不幸的一瞬——Wifi 突然不好用了！本来，该 Wifi 在过去的两天都没有问题，我也安排好了让 William 帮助我拉几下皮条，配合我拨打几个号，这下都没法演示了。实在是太 Danger 了吧？还好有应急预案，插上有线，IP 地址变了，重新配置 FreeSWITCH 及各终端，我终于在 4 分钟内完成了一个步骤，本来，我是准备了 19 个的。或许因为如此，我最后也跟获奖无缘。当然，或许 Wifi 不出问题我也不可能获奖，但我其实能演示更帅一些的。

值得庆幸的是，我还有机会在下午的最后一场演讲的时候把 Demo 走完，本来嘛，我其实都没准备参加 Dangerous Demo 的。下午的最后一场演讲可以说还是比较成功，除了我在现场演示的时候敲错了一个字符导致 FreeSWITCH 崩溃，没法演示最炫的两个步骤。或许是我学习罗永浩提前打预防针起了作用（It will crash, It must crash, if it crashes, I'll start over），大家对此也并不是很在意。

我演示的内容是视频转码和视频会议，会后，看到还是有些人对此挺感兴趣的。

结束了 Dangerous Demo 和我的演讲，身上一下轻松了许多，晚上就跟大家去酒吧 Happy 了。当然，不得不说今年这个节目不如去年的，去年的时候，有数不清的电子游戏及啤酒，今年，只有很单调的电子游戏，啤酒也没渴多少。后来，有一个哥们非要请我去别的地方吃饭，我只好提前离场了。当然，后来的三文鱼还是挺好吃的。

我们没有吃到很晚，便各自回去睡了。第一次，我睡了一整晚。

ClueCon 最后一天的时候，我感到精力充沛。没了 Demo 和 Presentation 的压力，便有了更多的时间跟不同的人聊聊。虽然在演讲的时候还是觉得有些卡壳，但跟别人交流的时候明显感觉到我的英语比以前说的顺当了。

ClueCon 的最后一场是一场圆桌会议，Anthony、Mike、Travis 以及 William 讲了 FreeSWITCH 开发最近的一些动作及发展规划，大家也都积极参与讨论。ClueCon 胜利闭幕。

总之，这一次 ClueCon 感觉从硬件和软件上都比往届要好。内容除涉及各 FreeSWITCH 成功案例外，还有 OpenSIPS、Kamailio、WebRTC、SIP/VoIP 安全、OpenBTS 等，知识面还是比较广的。

由于接下来的一天是 FreeSWITCH 培训，因而 FreeSWITCH 团队的主要人员都没有走，当晚我们又在酒店举行了一个小型的 Party。有意思的是，我给他们演示了怎么大口喝 Liquor（好像酒精度才 34）后，我俨然成了他们心目中的英雄，Anthony 也多次说：『Seven drink that liquor like water!』。酒后吐真言，话匣子打开，感觉他们都是挺可爱的。当晚，我们聊了一些以前从来都聊不到的一些话题，也听到了不少有趣的故事。酒，还真是好东西啊。当然，我能参加当晚的 Party，也是因为我搬到了洲际酒店，因为，按照规定，我必须在那里住一晚。只是，当晚，我大约 3 点半才回房间睡觉，早上 7 点离开，真有点辜负那么好的房间……

2.4 为什么会是我？

2015-08-04/Seven Du/FreeSWITCH 中文社区

ClueCon 的第一天是个 Hackathon。Hackathon 是 Hack 和 Marathone 组合出来的一个词，就是黑客马拉松。

这一天，除了有一些临时安排的演讲以外，最主要的是写代码游戏。游戏的名字叫 DTMF-u，任何人都可以用任何 API 写关于 DTMF 的游戏或应用。当然，由于该活动早就公布了，还是有一些人有备而来的，现场一个小时的时间真不够做点什么。

Anthony 的游戏是 Tic-Tac-Toe 的一个变种，叫 Tic-Tac-Verto，主要用了 Verto 模块和会议模块。有 9 个人加入 FreeSWITCH 的会议，每个人占据视频会议九宫格上的一个位置。主持人邀请两名志愿者上台，志愿者则轮流从 9 名与会人员中选择会议中的一个成员，并向它提问，问题的答案则由志愿者确认是否同意。询问的题目我大多听不懂，总之，有一个规则可以确定，在某人九宫格的位置画 O 或画 X，如果 O 和 X 连成一条直线或斜线，则论输赢。

游戏不错，只是现场带宽不够好，始终不能完美地显示九宫格。

其他参与者大部分是用的 FreeSWITCH 或 Twilio 的 API 做的 DTMF 应用，做一些呼叫什么的，也有跟移动 App 结合的。他们做得还是很不错的，但是，我没看清。问题就在这里，现场是 1080i 分辨率的 HDMI 大屏幕，所以字体看上去很小，他们也不知道把字体调大一下。所以，不管他们的作品有多么好，对大部分人来说都是——看不见！

我当然也是有备而来，事先就研究了一下 VNC，用 libvncserver 做了一个简单的 VNC server，把 VNC 跟 FreeSWITCH 结合起来。但看不出有什么亮点。所以，我又趁夜里倒时差的时间，用

libvncclient 写了一个应用。模块的名字叫 mod_vnc。实现一个 App 可以连接到一个现有的 VNC Server 上。这样，用一个普通的视频客户端就可以经过 FreeSWITCH 呼叫到这个 VNC Server，并看到画面。

我用的 VNC Server 就是一个 X 服务器，里面运行了一个 xterm，这样，我就可以用 DTMF 控制 xterm 里的输入。我还演示了一个俄罗斯方块游戏、贪吃蛇等。这些游戏都是 Linux 上的，但我可以用 DTMF 转换成方向键进行控制。在大屏幕上演示出来，效果超赞。

最终，我超过了 Anthony 赢得了比赛第一名。奖品是一个崭新的 Chromebook。

所以，我觉得，游戏就是游戏，一定要让大家看见，让大家觉得与众不同，才有机会赢。

好了，ClueCon 的预热已经过去，期待后面更好的开始。

另外一件令人高兴的事就是，微信公众号 [Official Accounts] 终于有了赞赏功能了。你会赞赏吗？

[^Official Accounts]: 微信公众号：FreeSWITCH 中文社区

2.5 为什么又是我？

2015-08-06/Seven Du/FreeSWITCH 中文社区

今年的 Dangerous Demos³ 一共有 14 人参加。

说起来好险。上午的时候，不小心电脑触控板上洒了些水。没多少，擦了擦就没当回事。谁知道过了一会鼠标自己乱动，还乱点，就像是被人黑了一样。重启电脑好几次，好不容易禁了 WiFi，依然如故。看来应该不是被黑，而确实是触控板进水的原因。急忙回房间用电吹风猛吹一阵，才好一些。

一上午净折腾电脑了。中午草草吃了点饭，才把环境基本准备好。这才发现我被排在了第一个上场。第一个上场也有好处，可以事先把电脑插到投影上。投影是 1080i 的分辨率，比我电脑平时用的大些，因而字体就显得小了。我趁机把画面和字体调整了一下，以便大家都能看清。

我今年演示的是 mod_x11，通过 X11 协议把视频推到一个远程的 X11 服务器上，每一路视频都会在 X11 服务器上生成一个窗口。这样，当把多路视频发到 X11 服务器上时，X11 服务器就成了一个融屏的桌面。各窗口可以自由拖动。

这个 Demo 虽然与黑客马拉松那天的完全不一样，但看起来还是有些像，因而大家就不怎么觉得兴奋了。所以，我也没指望获奖。

看看还有时间，我又演示了一个解数独的游戏。事先设定一些数独的题目，当视频呼入 FreeSWITCH 以后直接把数独表格显示在视频上面。通过 DTMF，依次输入“行” + “列” + “数字”三个数字填数。刚填了几个时间就差不多到了，我用了一个快捷键，按“#”号让 FreeSWITCH 自己把数独解了。

³一种黑客游戏，开发者开发个应用或程序，越炫，越超人意料越好。



这两个项目完全没有关联，我强行关联起来了。因而，觉得并不怎么好。后面的选手其实做地也都挺精彩的，看来也都是有备而来。Anthony 换了一种形式照样演示了 Tic-Tac-Vert Truphone Dangerous Demos 游戏。有一个哥们还演示了用 WebRTC 控制机器小车。其他大部分人都是按照自己的工作内容写的一些应用。

投票是所有在场的人发短信到一个特定的号码，参与投票的人不算多，我自己投了自己一票，险胜。奖品是一个 Dangerous Demos 的水晶纪念品。上面写着：

ClueCon - Chicago August 2015
Audience Vote



后来，发奖后，看到有一个人得票好像超过我了，哈哈，晚了。

这一次，我真不知道为什么自己得票最多。

2.6 ClueCon 2015

我的记忆已经不行了，这一年没写。翻译了一个篇别人的文章，比我写得好。更重要的是，里面提到了我的名字。—Seven

上周，Flowroute 参加了 ClueCon 2015 年度盛会。ClueCon 的口号是『源于开发者、用于开发者』(For Developers, By Developers)，这些开发者们都专注于创造有创造性的音视频、即时消息 App 及服务。Flowroute 从很久以前就开始支持 FreeSWITCH 以及 FreeSWITCH 相关的活动，因为这些活动在开发者、技术与创新之间有很好的平衡。今年，也是如此。我们的团队得以就业界很典型的一些问题，如，如何拥抱 WebRTC、攻击防御以及 API 等与很多开发者进行了交流。

用 API 促进下一波电信创新

API 是电信领域创意的源泉，在整个活动中都洋溢 API 的信息。会上有很多生动的演示，其中一个亮点是如何在 10 分钟内通过短短的 6 行代码在通信通道中加入即时消息支持。考虑到时间、资源及成本的因素，这种神奇的演示即使在两年以前也是很难想象的。另一个独特的案例来自于一个富有盛名的 FreeSWITCH 平台专家 Seven Du。他演示了如何通过 API 创建、修改以及挂断 Channel (通话)，如何在会议中上传及演示 PDF、GIF、PNG 文件以及在 FreeSWITCH 会议电话中播放幻灯片等。

防盗打仍然是一个挑战

盗打电话越来越得到大家的关注，不管你是一个消费者，还是一个企业，预防盗打电话仍然是成本很高而且效果非常有限的事情。

会上，通过机器学习来检测盗打是一个共同的话题。盗打检测一直以来就植根于我们服务的核心，我们早就在 Flowroute 服务中建立了自动的控制机制。你能看到，自从我们的用户第一天用我们的平台打电话起，我们就为每个用户创建了专用的外呼配置逻辑，包括已经打了多少分钟，每分钟能打多少电话，以及都打了哪些号码等。我们把这些号码又按一周内的天数及一天内的小时数分段分类。通过分析呼叫数据，我们可以发现违规的呼叫行为。如，如果我们发现短时间内有 10,000 个到古巴的呼叫，但你以前却从来没有往古巴打过电话，那么，我们会禁止这样的呼叫，并给用户发送 Email 告诉他们这种异常的行为。

如果这些呼叫是正常的呼叫，客户只需简单的点击 Email 中一个『解锁』链接，所有的外呼流量都会恢复。不可避免的是，对有些新客户而言，由于我们掌握的数据量不多，确实我们有时会误判。减轻攻击的另一个方法是使用号码白名单和设置最大呼叫频率，这些都可以在 Flowroute 的账户管理界面上直接设置。

WebRTC 的革命

大势所趋。随着一些全球性的运营商开始在 WebRTC 技术上的建模及应用，通信系统都纷纷开始拥抱 WebRTC。通信本身正在聚焦于一个以 WebRTC 为标准的平台，互联网工程任务组 (IETF) 及全球信息网协会 (W3C) 正在合力合作以将其标准化。

在会上，我们也看到很多 WebRTC 方面的兴趣以及创意。Verto 是一个很好的例子，它开发了一个很丰富的信令协议与 FreeSWITCH 交互，并在视频组件方面做了很多努力。另一个 WebRTC 创新是我们看到 ClueCon 使用多方视频会议、发言者检测、以及 3D 立体声会议。我们也跟面向对象的 RTC (ORTC) 的开发者聊过，ORTC 可以使开发和维护连接变得更简单。ORTC 标准支持高级的视频会议技术如分层视频编码 (SVC) 以及联播 (Simulcast) 等技术可以很容易的开发类似『Google Hangouts』之类的应用。

结束的那一天

很显然，使用 API 来提供有创造性的语音及即时消息 App 和服务是通信市场新的方向和核心。现在成了一个很令人兴奋的兴趣点。ClueCon 为我们提供了一个很好的论坛，我们得以看到第一手资料，看到开发者们是如何通过 API 创造应用的，并有幸一瞥未来的市场方向。

时间仓促，翻译里未仔细推敲。大家将就着看，不过瘾的看原文：

<http://blog.flowroute.com/2015/08/12/what-happened-at-cluecon-2015>

2.7 ClueCon 2016

时间过得好快，暖暖的 8 月，又迎来了新一届 ClueCon。

以前一直坐美联航的飞机，这一次，特意试坐了美国航空。周日早上 10 点钟从北京出发。

新飞机，飞机上有电源插座，我可以打开电脑做一些工作。飞机上也有 Wifi，不过比较贵，\$12 可以用两小时，\$17 可以用 4 小时，\$19 可以 Cover 整个飞行时间。也没什么要紧的事，就没有开。

在飞机上，发现我电脑的电池完全罢工了。以前充满电差不多还能撑十几分钟，现在断电就崩溃，比较郁闷。

飞机大约晚了一个小时，到芝加哥周日上午 11 点钟的样子。通关，到酒店差不多是下午 1 点了。这次 ClueCon 在芝加哥瑞士酒店 (Swissotel) 举行。

去酒店下面吃了点东西。菜单不认识，随便点了一个，不好吃。吃完回酒店睡觉。

一觉醒来，半夜了，国内是上午。发现微信上小伙伴们正在讨论一个线上的问题。花了几个小时研究代码，没搞定，睡觉。

又一觉醒来是早晨。匆匆吃了点早餐，到会场还不晚。见过从意大利过来的老朋友 Giovanni，热情打招呼。ClueCon 第一天是个黑客马拉松 (Hack-A-Thon) 以及一些自由的活动，也有零星的演讲什么的。Hack-A-Thon 我没有参加，主要有一些家伙用 3D 打印机以及 RaspberryPi 做电话什么的，有些意思。我没什么事，签到，跟大家打完招呼后，去苹果店修电脑。

在密歇根大街上就有一家，走路也不远，十多分钟的样子。检测完毕说要换电池，由于电池跟壳 (Top Case) 是连在一起的，都要换。\$199，但没有件，所以，要等大约三天的样子。我算了时间还来得及，就让他们进配件，我回 ClueCon 写代码。



图 2.5: Anthony



图 2.6: Anthony 和 William



图 2.7: Mike



图 2.8: Giovanni、Travis 等

由于一直比较忙，我的 ClueCon 主题也一直没有定，PPT 也没写。感觉都有点对不起这千里迢迢。过去的一年，零零散散修了些 Bug，也没有什么重大的东西做出来，因此也没什么好讲的。算了，想了想，还是拿出我写的一部分 XUI 代码，HTTP 以及 JSON API 来讲吧。准备 PPT 过程发现都是坑，因为有些代码是一年甚至两年前写的，年代久远，重新温习了一下，Debug 两天才都跑通，后话暂且不提。

下午美女 Kathleen 送来了 FreeSWITCH Solutions 的 T-Shirt，感觉真正被接纳为 FreeSWITCH Solutions 一员了。

第一天很快就过去了，晚上有其它公司赞助的 Pizza 晚宴，报名比较早的人可以参加，一起吃饭，扯扯淡，就过去了。时差原因，困得要死，回酒店什么都不想干，睡觉。



图 2.9: Pizza Party

周二，也是第二天，Cluecon 正式开始，起床一看都快 9 点了。吃了两块昨晚打包回来的 Pizza，算是早餐了。到达会场，正好 Anthony 开始讲。

Anthony 主要讲得什么是 FreeSWITCH，什么不是 FreeSWITCH 之类的，讲的内容其实我早就知道，但还是不容错过。其中最亮的一点就是演示了用 Microsoft Edge、FireFox Nightly、Safari 和 Chrome 四种浏览器用 WebRTC 加入同一个 FreeSWITCH 视频会议。

当天上午有一个 WebRTC 的圆桌会议，有 WebRTC 委员会的人参加，以及几个微软的负责 WebRTC 方面的哥们远程参加，远程视频会议当然是使用 FreeSWITCH 的 Verto Communicator 了。效果不错。我也被叫在了台上，说了几句，不过感觉发挥不是很好。

其他的演讲也都很好，OpenSIPS、Matrix.org、2600Hz、CGRates ……都有很多新的进展。其它话题不是很吸引我的时候，我就在写我的 PPT。



图 2.10: WebRTC

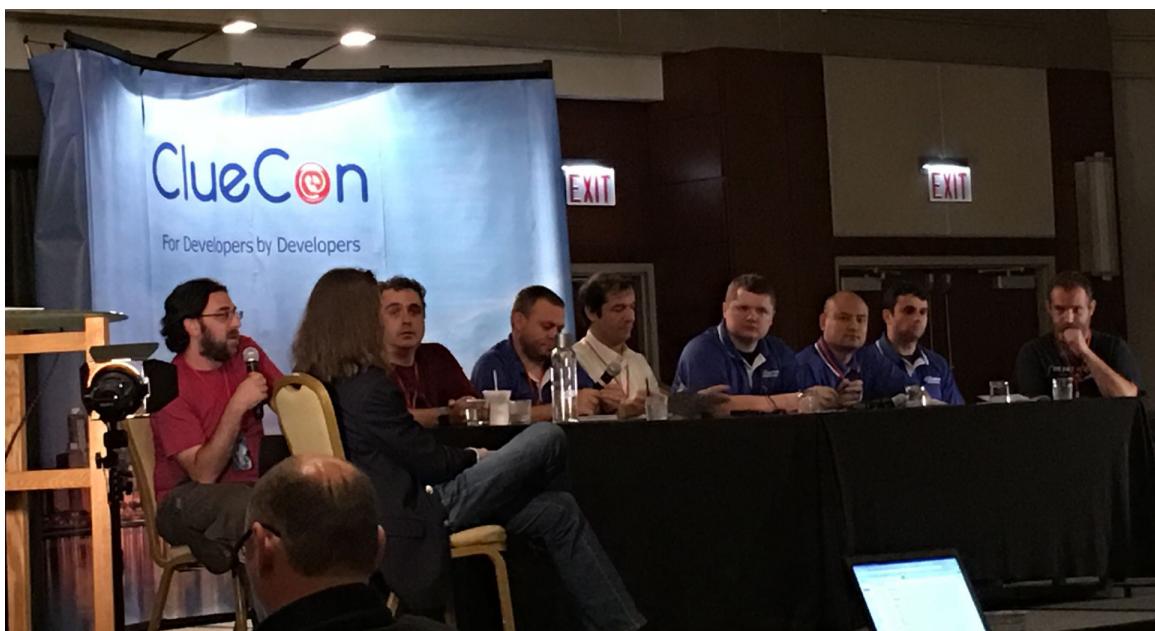


图 2.11: ClueCon Roundtable

晚上 ClueCon 请客，大吃一顿烤肉，回来，在会场有个小 Party，就是大家拿个酒瓶子随便找人聊天的那种。聊天的间隙我也不忘打开电脑干点活。10 点钟的样子吧，老戴来了，他飞机有些延误。老戴是我们 FreeSWITCH 中文社区的老朋友，很多年前就做 FreeSWITCH。现在在拉斯维加斯躲计划生育，又生了个美国娃。专程坐飞机过来看我。

晚上把 PPT 基本搞定，才陪老戴瞎聊会。他还要继续工作（他的同事都在国内），我就睡觉。

周三有点小迟到，上午听演讲，写 PPT，间或去大厅外面跟其他人瞎聊会。下午就轮到我演讲了。我就讲 HTTP 和 JSON API，以及我是怎么做的一些 Framework。还可以吧，演讲很顺利，大家还听得挺认真的。我告诉大家 $3+4=7$ ，而 7 (Seven) 是我的名字。并且，剧透了一下我明天要在 Dangerous Demo 上要演示的东西，跟这次演讲相呼应。其实，如果没有 Dangerous Demo 的话，我也就一块都讲完了。剩下那块留到 Dangerous Demo 上获奖。

晚上 ClueCon 还要请我吃饭，大致是看我远道而来没地方吃饭吧，不过我没去，因为约了老戴和另外一个朋友。

李小男是我以前在 Idapted 的同事，现在在加州上学。暑假出来玩正好来芝加哥，好多年不见，一块去 ChinaTown 吃小肥羊。话说，要想吃饱肚子还是得中餐啊。



图 2.12: 唐人街

吃完饭回来，听老戴讲他的创业故事。他真能讲，我都睡过去好几次。后来实在撑不住了，睡

下，他继续“上班”。

周四，又迟到，拜老戴所赐了。他上午睡觉，我还得去开会。上午最后一个环节是 Dangerous Demo。我都准备好了。

Dangerous Demo 其实就是个程序员的游戏，每个人 5 分钟演示个东西，越好玩、越刺激、越出人意料为好。现在 ClueCon 的老人几乎都知道我曾经有一年用一个电路板把主持人的手差点烫伤，从那以后我的每一次表演都是最 Dangerous，每年都得第一，Anthony 也就只得第二。他打趣说，他的 Demo 不 Dangerous，it always work :D。

今年我是这么做的。一上台，就告诉大家，我的电脑电池坏了，因此，上台只能先启动电脑，时间有限，导致演示更加 Dangerous。这是第一步，要让大家听到 Dangerous 这个字，很关键。

我演示的是用 Blocks 做 IVR，就是可视化编程的一种，用鼠标拖动一些组件，生成 FreeSWITCH 可以执行的 Lua 和 Javascript 程序。其实前一天我已经给大家演示过怎么拖动了，并用它做了个 $3+4=7$ ，为今天的演示更生动埋下伏笔。

接着，我演示了一个程序，就是写一个 IVR，有电话呼入后，用 TTS 播放 Hello ClueCon。

接着，做真正的 IVR，熟练的把不同的组件块拖出来一个 IVR。IVR 的内容是，欢迎致电 ClueCon，请按一个数字。如果按 1，就用 TTS 说“您按的是 1”，如果按 2 就说“您按的是 2”。实际上就是程序语言里的switch.. case语句。大家都知道switch.. case有default，就是，如果按的即不是1，又不是2，比如是x，就会说“x is too dangerous”。

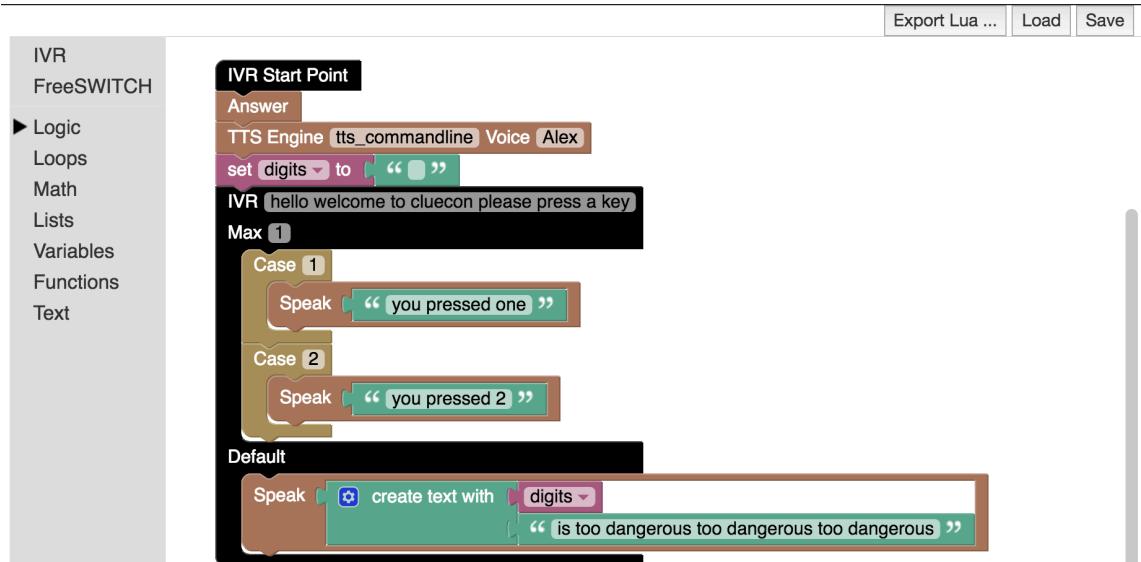


图 2.13: 用可视化编程拖出一个 IVR

最后演示的效果是：“7 is too dangerous！”为了增加效果，重要的事情说三遍：

Seven is too dangerous, too dangerous, too dangerous!

好了，最后，大家都记住了**SEVEN**，我的名字，是最危险的。

生成的 Lua 代码如下：

```

session:answer()
tts_engine = "tts_commandline"
tts_voice = "Alex"
session:set_tts_params("tts_commandline", "Alex")
session:setVariable("tts_engine", tts_engine)
session:setVariable("tts_voice", tts_voice)
digits = ''
digits = session:read(1, 1, "say:hello welcome to cluecon please press a key", 5000, "#")

if (digits == "1") then
    session:speak('you pressed one')
elseif (digits == "2") then
    session:speak('you pressed 2')
else
    session:speak(digits .. ' is too dangerous too dangerous too dangerous')
end

```

当然，它也可以生成 js 代码，只是略复杂些：

```

var digits;
var tts_engine = "tts_commandline"; var tts_voice = "Ting-Ting";
session.read = function(min, max, sound, timeout, terminator) {
    var dtmf = "";
    if (sound.indexOf(".") >= 0 || sound.indexOf("/") >= 0 || sound.indexOf("\\" >= 0) {
        session.streamFile(sound, function(s, type, obj, arg) {
            dtmf = obj.digit;
            return(false);
        });
    } else {
        session.speak(tts_engine, tts_voice, sound, function(s, type, obj, arg) {
            dtmf = obj.digit;
            return(false);
        });
    }
    if (max == 1) return dtmf;
    var digits = session.getDigits(max - 1, terminator, timeout);
    return dtmf + digits;
}
session.answer();
tts_engine = "tts_commandline"; tts_voice = "Alex";
session.setVariable("tts_engine", "tts_commandline");
session.setVariable("tts_voice", "Alex");
digits = '';

```

```

digits = session.read(1, 1, "hello welcome to cluecon please press a key", 5000, "#");
switch (digits) {
    case "1":   session.speak(tts_engine, tts_voice, 'you pressed one');
    break;
    case "2":   session.speak(tts_engine, tts_voice, 'you pressed 2');
    break;
default:
    session.speak(tts_engine, tts_voice, (String(digits) + String('is too dangerous too dangerous too dangerous')));
}

```

其他人的演示也都不错，但他们没有我这么能造。比如他们的演示太过复杂，屏幕上的字太小，让人看不清，最后的结果趋于平淡，在大家心中的印象就不是那么深。

这次的评分也很奇葩，主持人说一个名字，让大家喊，如果大家的声音越大，说明它的演示就越好。结果当然大家都知道了，说到我的名字时大家声音最大！

这次的奖品没看清具体是什么，在4种里随便选，我果断选了一本书—《Mastering FreeSWITCH》。这本书是两年前谋划的，本来是我主笔，但我瞎忙，这事就没做好，后来 Giovanni (也就是我第一天碰见的那个哥们，也是我们北京沙龙邀请的嘉宾) 主笔了，因此，我的名字就没出现在作者名单里，而是在贡献者列表里，小有遗憾。

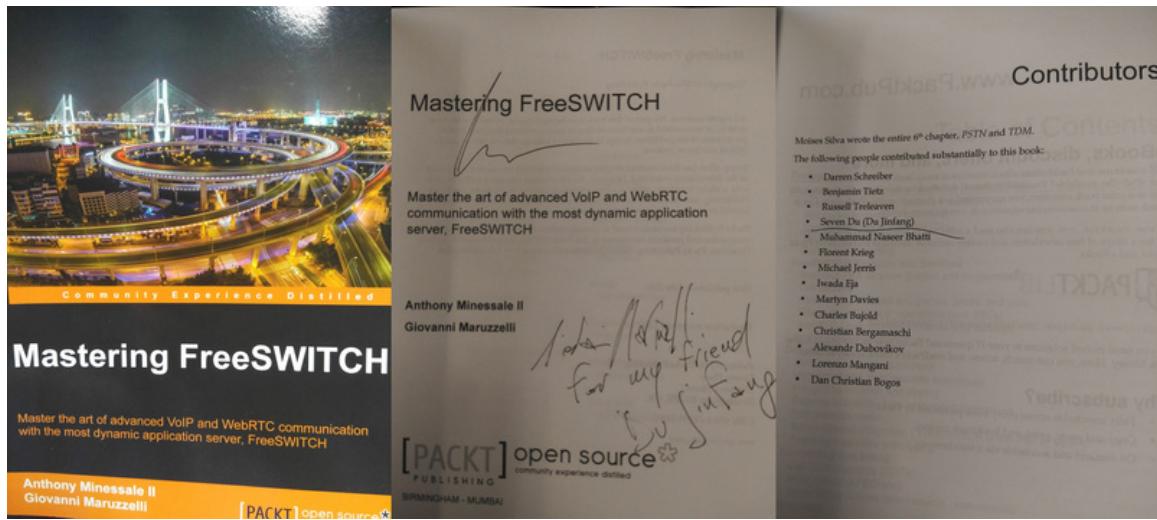


图 2.14: Mastering FreeSWITCH

下午，看到我的电池到了，抓紧跑去苹果店修，结果被告知需要 24 小时才能换好。我了个去，24 小时我得在飞机上了。让他们抓紧。他们说不是我想象的那么简单（其实我想不出来有多困难，就是换一下零件）。没办法，只好将电脑留在那里。他们说尽量在周五上午 10 点前修好。

下午收到邮件，说还有一个东西 (Display Assembly, 15-inch) 坏了，要换，600 多刀，我抓紧打电话过去说不换，600 多刀啊，不如买个新的了。他们说弄错了，他们已知那个硬件有问题，免费给换。晕死啊，有问题有问题有问题，我都用了好几年了…

该演示的都演示完了，又没了电脑，我感觉轻松很多（本来是想抓紧为公司干一些活的）。

下午又听了些演讲，在大厅聊聊天，ClueCon 就结束了。

晚上 ClueCon 又请客，跟他们一块去吃了号称芝加哥最好的鸡翅。吃完后还有一个 Party，由于某一年我喝酒表现突出，他们每年都会给我准备那种 Liquor，其实我也不知道牌子，说实话，我还是更喜欢喝点啤酒。

其中有好几个人都说喜欢我的 Demo，还有一个人说很喜欢我巧妙的植入了自己的名字，看来，我这个小伎俩有人识破了。

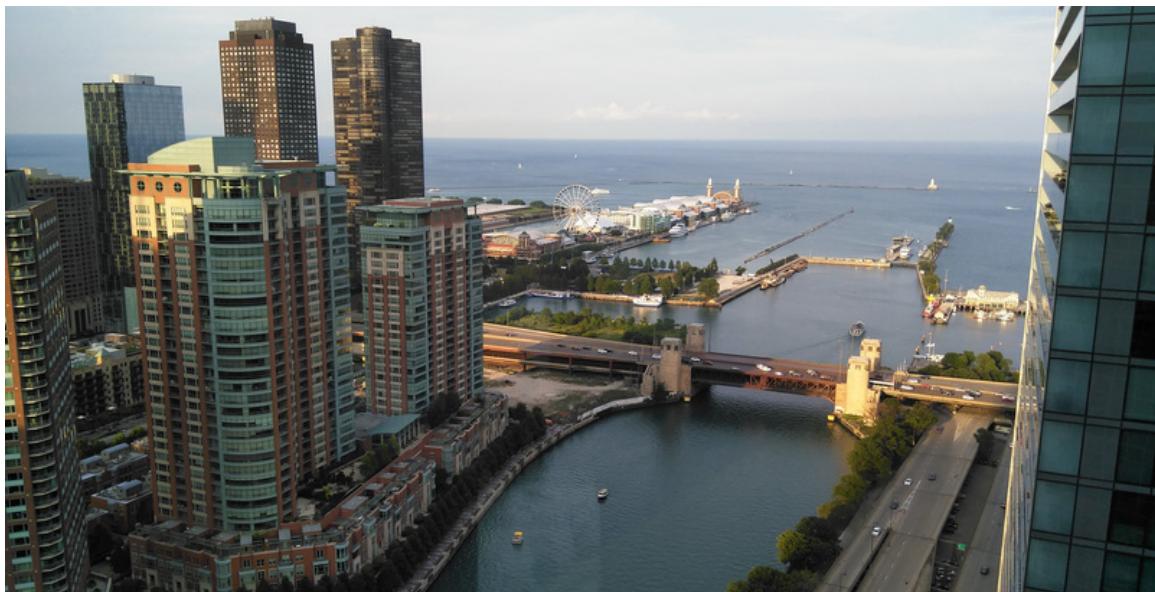


图 2.15: 酒店窗外

晚上回来见老戴不在，发信息说他在楼下抽烟，马上上来。上来后就又讲故事，讲到困死。我睡觉，他继续“工作”。

周五，有 FreeSWITCH 和 OpenSIPS 培训，两场同时在一个屋里，中间加个隔断。

我一觉醒来一看都 10 点了，直接就去苹果店拿电脑。却又发现多出来 400 多刀换了个内存相关的零件，我说，啊我都没收到邮件确认，怎么说换就换了呢，Blahblah blah，最后他们说，好吧，这钱我们出吧，我还是需要付\$199。

最后的价格是\$199 +\$39(人工) + 税，具体数字不记得了，反正是这么算的。

一次成型的电脑外壳都是全新的，显示器也是全新的（原来的脏了都擦不干净）…我想不出他们还有什么没换的，除了硬盘。

好吧，拿到电脑，我又可以工作了。当然，也有不好的地方，我本来希望今年秋季等新的 Mac Pro 出了再买新的，可能实现不了了。

回到酒店已经 11 点多了，收拾东西退房。老戴说他查到有个地方吃烤鸭，我说附近有个老四川，

他说咱俩这不说的一个地儿么，走起。烤鸭还是不错的，其它的酸菜鱼夫妻肺片就不正宗了，但中国味怎么也比老美的烂菜叶拌鸡胸好吃。

下午再回到酒店，找到 Mike，让他帮我调好一个模块的编译问题。时间不多了，一一告别，跟老戴一起坐地铁去机场。

地铁还是挺便宜的，从奥黑尔国际机场到市区是\$5，从市区到机场是一人\$3。老戴的飞机比我的晚，把我送到国际安检，依依惜别。

很遗憾，没有倒出时间给大家捎一星半点东西。

临登机的时候，惊奇的发现 Ethan 站在我面前。他也是我在 Idapted 时的同事，去美国转圈，跟我同一班飞机回国。简单叙叙旧，匆匆登机，相约飞机上继续聊。

上飞机先睡了一觉，跟 Ethan 聊半天，由于我们太兴奋，其他人都有意见了，我们才各自回到座位。

毫无睡意，拿出电脑写下这些文字。新电池！

第三章 FreeSWITCH 那些事

3.1 FreeSWITCH 是什么？

FreeSWITCH 是什么？其实很难给它下一个准确的定义。如果你是做 Web 开发的，那么，MySQL 和 Apache 大概是你最熟悉、最常用也是最基础、最不可或缺的软件，如果你是做通信的，那么，FreeSWITCH 就跟 MySQL 和 Apache 那么基础，那么鼎鼎大名。

是的，就跟上面提到的两款开源软件一样，FreeSWITCH 也是开源的。而且 FreeSWITCH 也是有定义的。FreeSWITCH 开源社区官方给它的定义是：The World's First Cross-Platform Scalable Free Multi-Protocol Soft Switch and Media Engine，翻译成人类能懂的语言就是：世界上第一个跨平台的、伸缩性极好的、开源免费的、多协议的软交换系统和媒体引擎。

首先它是**跨平台**的。它能原生地运行于 Windows、Max OS X、Linux、BSD 及 Solaris 等诸多 32/64 位平台，它具有很好的可移植性，也有人成功的将它移植到了 Linksys NLS2 平台及 Raspberry Pi¹上)。

它具有很强的**可伸缩性**—从一个简单的软电话客户端到运营商级的软交换设备几乎无所不能。

它是**免费的**，它采用 MPL 1.1²协议授权，意味着任何人都可以免费的使用并获取源代码，任何人都可以修改、发布、甚至出售自己的应用。

它支持 SIP、H323、Skype、Google Talk 等**多种通信协议**，并能很容易地与各种开源的 PBX 系统如 sipXecs、Call Weaver、Bayonne、YATE 及 Asterisk 等通信，它也可以与商用的交换系统如华为、中兴的交换机或思科、Avaya 的交换机等互通。

它可以用作一个简单的交换引擎、一个 PBX，一个媒体网关或媒体支持 IVR 的服务器，或在运营商的 IMS 网络中担当 CSCF 或 Application Server 等。

它遵循相关 RFC 并支持很多高级的 SIP 特性，如 presence、BLF、SLA 以及 TCP、TLS 和 sRTP 等。它也可以用作一个 SBC 进行透明的 SIP 代理（proxy）以支持其他媒体如 T.38 等。

它支持宽带及窄带语音编码，电话会议桥可同时支持 8、12、16、24、32 及 48kHz 的语音，可以支持立体声喇叭及麦克风，可以支持丰富的视频会议功能。

¹Raspberry Pi (<http://www.raspberrypi.org/>) 是一个装有 ARM CPU 的信用卡大小的计算机。事实上，笔者办公室的 IPPBX 就是一台运行着 FreeSWITCH 的 Raspberry Pi。

²Mozilla Public License。详见：<http://www.mozilla.org/MPL/1.1/>。

它一直紧跟时代步伐，永远支持最新的协议和最酷的技术，如最新的 WebRTC 技术、OPUS/VP8/VP9 语音和视频编码等。同时，它也提出 NDLB³，致力于支持最老旧的设备和通信技术，如传统的 TDM 话机和传真机等。

FreeSWITCH 从 1.6 版本开始支持视频转码和视频会议，FreeSWITCH 可以支持 4K 分辨率的视频会议，支持 64（甚至更大）方参会者融屏，支持录音录像流媒体推送（视频直播）以及向会议室中播放视频、PDF 文档等。

从技术上讲，它是一个**B2BUA**⁴。而它的黄金搭档，OpenSIPS 和 Kamailio，则通常作为 Proxy 跟它一起部署。

好吧，如果上面说的都太技术了，你可以把它理解成一个支持网络（IP）语音和视频通话、会议的服务器。

3.2 FreeSWITCH 能做什么？

这一段，我偷了个懒，抄了 FreeSWITCH Wiki 上的内容。

3.2.1 功能列表

这是一个不完整的列表：

- PBX
- 话务路由服务器
- 语音编码转换服务器(B2BUA)
- IVR 及语音服务器
- 语音和视频会议服务器
- 语音信箱
- 会话边界控制器(SBC)
- 传真服务器
- 可用作 T.38 网关或直接接收和发送传真
- 语音识别（ASR）及语音合成（TTS）系统对接
- Flash RTMP-SIP/PSTN 媒体网关
- WebRTC 服务器及媒体网关
- Call Center - 呼叫中心
- ACD - 高级呼叫分配
- onhook/offhook 坐席

³No Device Left Behind，不让任何设备掉队。

⁴Back-to-back User Agent，背靠背的用户代理。。

- 录音、录像服务器
- 指挥调度系统通信平台
- 视频监控
- 电信运营- 400/800 综合业务平台
- 企业总机
- 托管型呼叫中心及企业呼叫中心
- 彩铃、统一通讯
- IMS 中的 CSCF、Application Server

以下内容来自<https://freeswitch.org/confluence/display/FREESWITCH/Specifications>

◦

3.2.2 Features

- WebRTC support
- Centralized User/Domain Directory (directory.xml)
- Nano Second CDR granularity
- Call recording (In Stereo caller/callee left/right)
- High Performance Multi-Threaded Core engine
- Configuration via cURL to your HTTP server (mod_xml_curl).
- XML Config files for easy parsing.
- Protocol Agnostic
- ZRTP support for transparent RTP based key exchange and encryption
- Configurable RFC 2833 Payload type
- Inband DTMF generation and detection.
- Software based Conference (no hardware requirement)
- Wideband Conferencing
- Media / No Media modes
- Proper ENUM/ISN dialing built in
- Detailed CDR in XML
- Radius CDR
- Subscription server
- Shared Line Appearances
- Bridged Line Appearances
- Enterprise/Carrier grade Eventing Engine. (XML Events, Name Value Events, * Multicast Events)
- Loadable File formats and streaming

- Stream to and play from Shoutcast and Icecast
- Multi-lingual Speech Phrase Interface
- ASR/TTS support (native and via MRCP)
- Basic IP/PBX features
- Automated Attendant
- Custom Ring Back Tones (Early Media)
- XML-RPC support
- Multiple format CDRs supported
- SQL Engine provides session persistence
- Thread Isolation
- Parallel Hunting
- Serial Hunting
- Mozilla Public License
- Support
 - Paid support available
 - Free support via IRC & E-mail
- Many supported codecs
 - CELT (32 kHz ahd 48 kHz)
 - G.722.1 (wideband)
 - G.722.1C (wideband 32 kHz)
 - G.722 (wideband)
 - G.711
 - G.726 (16k, 24k, 32k, 48k) AAL2 and RFC 3551
 - G.723.1 (passthrough)
 - G.729AB (Requires a license unless using passthrough)
 - AMR (passthrough)
 - iLBC
 - Speex (narrow and wideband) with RFC 5574 fmp support
 - LPC-10
 - DV14 (ADPCM) 8 kHz and 16 kHz
 - SILK
 - OPUS - RFC 6716
 - Video Codecs:
 - Theora
 - H.261
 - H.263

- H.264
- MP4
- VP8
- VP9
- Live Migration of calls from one FreeSWITCH box to another.
- Voicemail
- Multitenancy - Enterprise/Carrier configuration
- Time of Day Greetings
- Urgent Message Tagging
- E-mail Delivery
- Playback and Rerecord messages before delivery.
- Keys are templates so you can rearrange to fit your needs.
- Callback support from inside voicemail.
- Podcast of Voicemail (RSS)
- Message Waiting Indicator (MWI)
- Support for Queues (via mod_fifo or mod_callcenter)
- Parking (via mod_fifo)
- Conference
- Software based Conferencing without any hardware requirements.
- Wideband conferences.
- Multiple on-demand or scheduled conferences with entry/exit announcements
- Play files into the conference or a single member.
- Relationships
- TTS integration
- Transfers
- Outbound Calling
- Configurable Key Lay
- Volume, Gain and Energy level per call.
- Bridge to Conference transition
- Multi Party outbound dialing.
- RFC 4579 SIP CC Conferencing for UAs
- Automatic or on-demand recording
- RSS Reader
- Fax endpoint, gateway and passthrough mode.
- T.30 (G.711) Audio Fax (via mod_spandsp) formerly known as mod_fax.
- T.38 faxing (gateway, endpoint and passthrough)

3.2.3 Protocols

- SIP with mod_sofia
- UDP, TCP, SCTP and TLS transports for full SIP compliance.
- SIP v.2.0 (RFC 3261)
- IPv6 Support
- SIP Session timers
- RTP Timers
- RFC 3263 (SRV and NAPTR)
- RFC 3325
- RFC 4694
- SRTP via SDES (Works with Polycom, Snom, Linksys and Grandstream)
- Blind SIP Registration
- STUN Support
- Jitter buffer
- NAT Support
- Distributed SIP registrations
- Late Codec Negotiation
- Multiple SIP registrations per user account.
- Multitenancy - Multiple SIP UAs
- SIP Reinvites.
- Can act as an SBC (Session Border Controller)
- Manage Presence
- SIP/SIMPLE (can gateway to other chat protocols)
- SIP Multicast Paging support for Linksys and Snom
- Intercom/AutoAnswer support.
- Call features like Call Hold (Re-INVITE), Blind Transfer (REFER), Call Forward (* 302), etc.
- Jingle with mod_dingaling
- Interop with Google Talk and Telepathy

3.3 FreeSWITCH 的版本

该选择哪一个版本呢？这个问题其实一言难尽。FreeSWITCH 已经发展 10 年了，在这 10 年里，有很多很多的故事，也有很多因素影响你的选择。下面，我们简单探讨一下。

3.3.1 FreeSWITCH 1.0

如果不是历史遗留项目，你肯定不想选择 FreeSWITCH 1.2 之前的版本。年代太久远了。

3.3.2 FreeSWITCH 1.2

FreeSWITCH 1.2 很稳定，用得也多。而且，FreeSWITCH 编译起来也容易（因为还没有发生 FS-353⁵）。1.2 是基于 CentOS 5 开发的，对旧的操作系统兼容性较好。缺点是，1.2 版再也不会更新了，如果用于生产系统，将无法得到官方的安全更新和技术支持。

3.3.3 FreeSWITCH 1.4

这是当前的发行版，你应该总是用最新的版本（如，本书截稿时最新的版本是 1.4.20）。不过，由于发生了 FS-353，系统编译起来比较复杂，依赖于系统提供的一些第三方库。但是，你可以用 Debian 或 YUM 仓库中安装 FreeSWITCH，那样会自动安装依赖。1.4 版还会有安全更新，如果现在开始用的话，最好用这个版本。

3.3.4 FreeSWITCH 1.6

FreeSWITCH 1.6 增加了视频转码和视频会议支持，其它的如视频 Jitter Buffer 以及 RTCP 控制功能是新加的。所以，如果你做视频应用的话，肯定应该选择该版本。

至本书截稿时，FreeSWITCH 最新的版本是 1.6.2 版，Debian 8 上有已经编译好的安装包，参照官方的 Wiki 页面⁶很容易安装。

编译运行 FreeSWITCH 最好的环境也是 Debian 8，其它 Linux 发行版会麻烦一些，需要很多 Linux 系统方面的知识，不建议初学者使用。

如果没有特别说明，本书中的内容都是大部分版本都通用的（很显然，一些新的视频应用只有在 FreeSWITCH 1.6 中才通用）。

3.3.5 FreeSWITCH 1.8

2016 年 8 月，又一届 ClueCon 胜利闭幕。FreeSWITCH 1.8 很快就要发布了。届时，将给我们带来基于 Session 的实时文本聊天功能。

基于 Session 的实时文本聊天，将可以通过 RTP 传输，也可以通过 MSRP 传输，以及在 Verto 模块中通过 Websocket 传输。

让我们拭目以待。

⁵早期，FreeSWITCH 把一些第三方依赖库都放到 FreeSWITCH 代码库里，编译简单。我们后面还会专门讲这段故事。

⁶<https://freeswitch.org/confluence/display/FREESWITCH/FreeSWITCH+1.6+Video>。

第四章 八卦江湖

有人的地方就有江湖。江湖上那些事…

4.1 Git 仓库迁移

杜金房/2016.05.30

时间过得很快，又是月底了。

很早之前，就收到了 GitCafe 关于 Git 仓库的通知，但一直没倒出时间来做。迁移截止日期是 5 月 31 日，再不迁移，项目就会被永远删除了。

其实，之前也试着迁移过，但是没有成功。今天倒还顺利。用 GitCafe 提供的工具，顺利就把 Git 仓库迁移到 Coding.net 了。

但，迁移完毕只是万里长征第一步，还有更复杂的事情在后面。

等等，你说什么叫 GitCafe？什么叫 Coding.net？甚至，什么是 Git？

科普下（其实今天要科普的事情很多）。

Git 是一个版本管理系统，是由著名的大神 Linus Torvalds 写的。是的，这个大神就是鼎鼎大名的 Linux 操作系统的作者。

Git 非常强大，好用，很快就代替了程序员们以前常用的 CVS，SVN 等。

Git 是全分布式的，但是，团队协作还是有个公共的仓库比较好，因此大家都会用各种各样的手段建 Git 仓库。直到出现了 Github。

Github 是什么？简单来说就是一个仓库平台，谁都可以申请一个账号，上去建仓库，所有账号和仓库都是免费的。厉害吧？

当然，上面的“所有”应该是加引号的。如果你创建的是一个公开的仓库，所有人都可以看仓库里的内容，那，就是免费的。但如果你想自己玩，不让别人或除你团队之外的人看，这叫私有仓库，是要收费的。毕竟，人家 Github 也要吃饭嘛。

但是，使 Github 风靡全世界的，正是这些公开的仓库。世界上各种各样的开源项目，都慢慢地迁移到了 Github 上。有一些开源项目，如 FreeSWITCH，不使用 Github，但是，也有在 Github 上的镜象（Mirror）。

在人人写博客时代，Github 又推出了一项服务，叫 Github Pages。即，你可以建一个公开的 Github 仓库，commit, push，关联你自己的域名，然后，你就有了一个静态的网站。静态网站除了直接用 HTML 和 JS 写以外，还可以直接用 Markdown 写。有一个叫 Jekyll 的软件可以将 Markdown 版的网站转换成 HTML。当然，你也可以不用 Jekyll，因为，你只需要直接 push Markdown，Github 会自动帮你转换。总之，很好，很强大。

但是，好东西总会离我们越来越远。不知道哪一天，不知道谁 push 了一些“敏感”的内容，从那时起，祖国大陆访问 Github 就不那么流畅了，隔三差五的抽风，有时，压根就不能访问。

是的，比起 Google、Facebook 和 Twitter 这几个根本不存在的网站来说，Github 对我们来说还是可见的。但是，对于天天趴在 Github 上看代码的程序员们来讲，卡顿一分钟就等于扼杀生命啊。

不过，世间总少不了雷锋。Gitcafe 横空出世，基本上在祖国大陆具备 Github 的所有功能。虽然，少了 Github 的生态环境，但是，把 Git 仓库尤其是 Github Pages 迁移到 Gitcafe 和 Gitcafe Pages 后，在速度上还是会带来很好的体验。慢慢的，很多项目就迁进来了。包括我的一些项目和个人主页。

其实，Github Pages 可能做的并不好。前一阵，它宣布跟 Coding.net 合并了。Coding.net 除了具备 Github 和 Gitcafe 的大部分功能外，还多了一些团队协作的工具。

既然兵合一处，将打一家了，服务名就只剩下 Coding.net 了。因此，它们要求在 5 月 31 日前将所有项目都迁移到 Coding.net 上。

今天，是 5 月 30 日，我动手还来得及。

如果有曾经关注我的书的，<http://book.dujinfang.com> 应该在早些时候就不能访问了。由于我用的是 Gitcafe Pages，而 Gitcafe Pages 已处于只读状态，我也不知道为什么就不能访问了。总之，一直没得闲修一下。今天，把网站迁移到 Coding.net 后，服务应该是恢复了。当然，由于 DNS 解析需要最长 72 小时才能同步全球，所以，有的人可能还要等待一段时间才可以看到。不管你能否看到，欢迎给我留言反馈下，如：北京联通，正常。

同时迁移的还有我们的 FreeSWITCH 镜像。是的，Github 上也有镜像，慢。我们从很早就维护了一个 Gitcafe 的镜像，在国内访问还是比较快的。现在，迁移到 Coding.net 了，网址和代码仓库地址如下：

<https://coding.net/u/dujinfang/p/FreeSWITCH/git>

<https://git.coding.net/dujinfang/FreeSWITCH.git>

资源所限，目前仅同步了 master、v1.4 和 v1.6 分支，每十分钟跟官网仓库同步一次。

再做一次雷锋，希望能帮上大家一点点。

晚安。

4.2 FreeSWITCH 问题解决的几个例子

大家经常在 QQ 群里问到关于 FreeSWITCH 的问题，却总是不得要领。我花了很多时间写《FreeSWITCH 新手指南》，可是大家都不怎么爱看。当然，可能《指南》比较泛泛，没有针对性。今天，我们就拿几个实例来说吧。

0x01:

问题是这样的，在 QQ 群里，有个同学问：FreeSWITCH 启动一闪就退出了，怎么办？

我的判断：FreeSWITCH 一闪的情况，可能只会在 Windows 上出现。但是，这位同学确实应该这么问：

我使用 Win7，安装了 FreeSWITCH（或从源代码安装了 FreeSWITCH），版本是 XXXX。我尝试启动时双击开始菜单里的 FreeSWITCH 启动项，有个窗口一闪就退了，看不清，请问这是什么问题呢？怎么解决？

这样提问就比较明确了。而前面的提问其实有效信息真的很少。其实 FreeSWITCH 典型的应用是在 Linux 上。如果是 Linux，还要说明，是 CentOS，还是 Debian，还是 Ubuntu 等，还要说明版本，如 CentOS 5.5 或 Debian 8.2。但是遗憾的是，大多数同学的提问基本是就假设 QQ 群里的人都站在他身后，“你看，这个一闪就退了，怎么办”，他不知道其实别人不知道他用的是什么操作系统，用的什么 FreeSWITCH 版本。

好了，再回到我们的问题。根据经验判断，可能是因为 FreeSWITCH 在启动时没有权限无法锁定 FreeSWITCH.pid 文件，致使进程退出。跟他说了用管理员身份运行一下（这个不用教了吧？）。

后来它以管理员身份运行了还不行。那就得找其它原因了。我让他进到命令行模式下，找到 FreeSWITCH 的可执行文件（如 c:\Program Files\FreeSWITCH\FreeswitchConsole.exe），手工执行一下，就应该能看到出错信息。

他照做了，出错信息是类似 XML Parsing Error, Unclosed Tag <endpoint 之类的。

根据经验判断，他改过 XML 文件，改错了，所以才出现这样的错误。

他说他改回去了，还是这样的错误。我建议他全部重装。怎么解释呢？他说改回去了，但肯定没有完全恢复到原来默认的状态！

最后，其实这个问题应该这么问：

大家好，我使用 Win7，安装了 FreeSWITCH，版本是 XXXX。我尝试启动时双击开始菜单里的 FreeSWITCH 启动项，有个窗口一闪就退了，看不清。后来，我从命令行模式下输入 freeswitchconsole.exe，显示的出错信息如下：xxxx，我记得改过 XML 的某某部分，现在又改回去了，还是出现这个错误，请问我该怎么办？

这样就是一个比较好的问题了。

我们没收到这个同学的反馈。这也是在 QQ 群里提问的问题。因为我回答完就睡觉去了，后来也没看到他回复。不管问题解决没解决，最好都有个结果反馈。当然，更好的方式是把问题发到 BBS 上，这样便会有完整的记录。

0x02:

A: 为什么按照官网上的语法 `api.executeString("version");` 会报错，Error in `API::executeString` expected 2..2 args, 通过 `api = freeswitch.API();` 获取的 api

B: 这样

```
api = freeswitch.API(); api:executeString( "version"):
```

A: 我就是这么写啊。

点评：A 和 B 写得其实不一样。找不同，你能找到几处不同？

0x03:

A: 我按照书上的方式配置了 Dialplan，可是无论打什么号码都会进入这一个，进不了其它的，请问是什么原因？

B: 把日志贴出来

A: [日志]

B: 把 Dialplan 贴出来

A: [Dialplan]

B: 你是照着书上写的，但是把字母抄错了。

点评：为什么我们要求提供各种背景信息和日志？因为好多人非常坚定地声称他是怎么怎么做的，但是，你还是不能相信它。一切，以日志说话。

说到这里，其实我也犯过错误。我去给人家修网络，最后定位在线路的问题，打电话到联通客服，经过一番排查，最后发现是 ADSL 猫没加点，囧。

4.3 FreeSWITCH 代码最新变化

是的，FreeSWITCH 快要支持 VP9 了（实际上以前支持过，但跟不上 Chrome 更新的速度）…

记得以前说过关于 FS-353 的问题。FreeSWITCH 依赖很多现成的第三方库，最初，FreeSWITCH 把大部分第三方库直接放到 FreeSWITCH 的代码树里，编译安装比较方便，但是，编译起来也慢一点。

后来，FreeSWITCH 为了做安装包，就把大部分第三方依赖库从代码树里删除了，而是转而依赖系统提供的库。编译起来快多了，尤其是对我们这些一天编译好多遍的开发人员来说。但是，各种

操作系统提供的库都有很多差异，即使同是 Linux，也差别很大。因此，FreeSWITCH 官方仅支持 Debian 8，对于其它的系统，都由社区的其他参与者支持。

一个不可忽略的事实是，很多人在用 CentOS、Ubuntu 等。编译起来不顺畅，这是大家比较抱怨的地方。

最近，也有人反映，即使在 Debian 8 编译也出错。为什么呢？命不好，正好遇到 FreeSWITCH 代码调整。

FreeSWITCH 内部使用了 libvpx 库，直接挪用了 vpx 提供的 I420 图片格式支持，以及 VP8/VP9 的支持。另外，还使用 libyuv 做图片的转换和缩放等。

这两个库是随着 Chrome 一起升级的。Chrome 更新了，库更新了，所以，我们也要更新 FreeSWITCH，这就是现实。

本来，一切都做好了，更新也就是分分钟的事，但是，总会有不顺利的事情。

为什么这么麻烦呢？还得从以前说起。

Anthony 发现 libvpx 有 Bug，无法支持高并发（也就是说视频会议开到几路就不行了），但人家 Chrome 是个客户端软件，开发人员也不怎么积极去修，所以，FreeSWITCH 官方不得不维护了自己的版本。这就是为什么说如果编译 FreeSWITCH，要从 FreeSWITCH 官方下载这些第三方库的原因。

为了防止与实际的 libvpx 冲突，FreeSWITCH 官方将这个版本命名为 libvpx2，对应原 libvpx1.4 的版本。虽然经过大量测试，但难免挂一漏万，所以，编译起来有时多多少少总会出点小问题。不过，最后的安装包应该都是比较稳定的，也就是说，按官方的指南，直接 apt-get 安装问题应该不大。

问题来了。libvpx 更新到了 1.5.0，这个版本改变了内部的 ABI（没写错就是 ABI），但是，没有按规矩更新 ABI 版本号。这，是一个纯粹的 Bug，让我们无法适从。但问题是，FreeSWITCH 维护的 vpx 代码 master 版本已经更新到这个版本了。所以，有些同学直接 clone/pull 代码，一编译就出了问题。因为 FreeSWITCH 还没有同步更新。

FreeSWITCH 团队其实也没闲着，早就给 Google 报了 Bug，但是，等了一周基本没什么进展，我们才又抓紧把代码适配到 1.5.0。说起来简单做起来难，且不说 vpx 有 Bug，就是没有 Bug，也得适配其 API/ABI 的变化，我们又看代码又查历史记录，真花了不少时间。

好吧，最后问题基本解决了。又发现，libvpx 又更新了，而且，这次，ABI 也更新了，而且，Chrome 里用了更新的 libvpx …

最终，FreeSWITCH 团队不得不痛下决定，算了，我们再也不能依赖操作系统提供的库了，因为操作系统提供的库总会滞后 N 个版本，而 Chrome 之流的软件更新又很激进。所以，就又回到了 FS-353 所描述的争论起点。把 libvpx 和 libyuv 的代码直接放到了 FreeSWITCH 源代码树里，跟 FreeSWITCH 一起编译。

嗯，放进去归放进去，还是有不少曲折的。因为，原来的 libvpx 是编译成动态库的，内置到 FreeSWITCH 里，标准的方法是编译成静态库，而在编译成静态库这一步，就偏偏出了问题。Mike

就是专门管这一块的，他问我对汇编熟悉不，问题出在一块汇编的代码文件里。我汇编也就会那么一丁点了，最后还是 Mike 找出了问题，编译通过了。

所以，如果你是使用最新的master代码，`libvpx`和`libyuv`已经内置到 FreeSWITCH 里了，再自己编译的话，就省了不少麻烦。

另外，`mod_vpx`已经不存在了，因为它已被移动到核心里去了。

有 Redhat 用户说编译可能还有问题，如果你使用 CentOS 或 Redhat，不妨测试一下，如果有问题，向 FreeSWITCH 官方报 Jira，这也是为 FreeSWITCH 做贡献的好机会啊。

基本就是这样，FreeSWITCH 肯定会越变越好。当然，也离不开大家的支持。

4.4 FreeSWITCH 与 FFmpeg

杜金房 / 2016.03.11

关于 FreeSWITCH 与 FFmpeg 的恩怨可以讲很多，不过，让我们长话短说。

FFmpeg 是比较流行的多媒体库，可以处理语音视频之类的，在开源领域内得到了大量应用，包括 Android 和 Chrome。如果我没记错的话当年 QQ 也用过它，但因没有遵循开源协议开放源码而被钉在了耻辱柱上。

几年前（具体时间懒得查了），由于开发团队的分歧，FFmpeg 分裂了。部分开发者另起一摊，Fork 了一下，起名叫 Libav。但问题是，虽然项目名称改了，但为了跟大多数现在应用兼容，库名称依然叫`libavcodec`、`libavformat`之类的。这是所有恶梦的开始。

其实我在更早的时间就开始在 FreeSWITCH 里基于 FFmpeg 写一个模块，最初叫`mod_ffmpeg`。第一个可以运行的版本是在从 Cluecon 回来的飞机上调试成功的。

后来，由于 CentOS 的诡异问题，FreeSWITCH 开发团队将开发平台迁移到了 Debian，而 Debian 使用 Libav，所以，我们趁机将`mod_ffmpeg`改为两个模块，叫`mod_avcodec`和`mod_avformat`。

`libavcodec`是做编解码处理的，`libavformat`是处理多媒体文件的。相对的，`mod_avcodec`就实现了 H264、H263 之类的编码，而`libavformat`就支持mp4文件等。

后来，这两个模块合并成了一个模块，叫`mod_av`。就是大家在 FreeSWITCH 1.6 里看到的。不过，这个模块默认是不编译的，所以，如果需要的话要手工编译。原因很简单，`libav/ffmpeg`里有一些依赖库使用的是 GPL 的（如`libx264`）。

在 Debian 上编译很简单，要知道，为了能在 Debian 上顺利编译，开发团队也是费了很大劲的。大家都希望能支持 CentOS，但 CentOS 上是 FFmpeg，需要做兼容，FreeSWITCH 团队的开发力量不够，因此，这个依赖于社区支持。但遗憾的是，没有任何一个人贡献一行代码。我们中文社区的群里也有很多人遇到这个问题，不厌其烦地问，也没有任何一个人说出点钱资助一下解决这个问题。

所有人都告诉我 CentOS 是刚需，但对于我来讲，没有人愿意贡献代码或出资来做这件事就不是刚需。关于刚需这个话题，我改天再专门写文章来讲。今天就不多说了。我们小樱桃团队刚刚成立，

作为一个有情怀的团队，我们投入了一些力量，把这个任务完成了。虽然过程很曲折，但其实最后改的代码也没有几行。不敢独享，跟大家分享一下相关的技术要点。

首先，FFmpeg 本身有很多版本，分裂后版本就更多了。我最初开发是基于 0.8.x 的，后来就直接基于了 FFmpeg 的master 版，后来，就试了 libav 11.3、11.4、11.6（上个月刚刚发布）。最近试了 FFmpeg 2.6.x、2.8.x、3.0，基本都能顺利编译过。

为了同时测多个版本，我们需要一些技巧。

首先，卸载所有随系统安装的版本。重新执行 FreeSWITCH 的configure，让 FreeSWITCH 找不到 libav 和 FFmpeg。

然后，编译安装各个版本的 Libav 和 FFmpeg。

编译步骤满大街都是，主要的几个参数是：

```
--prefix=/opt/av
```

我安装到了 /opt/av，当然你也可以装到 /opt/av-11.3、/opt/av-11.6 之类的，FFmpeg 也是一样。

在开发过程中我们还遇到 libx264 新版本导致的问题，所以还测试了很多版本的 libx264：

```
/configure --prefix=/opt/x264
```

为了让 Libav 能找到我们自己编译的 x264，指定 PKG_CONFIG_PATH：

```
PKG_CONFIG_PATH=/opt/x264/lib/pkgconfig ./configure --prefix=/opt/av
```

其它的参数就看你的需求选了，下面的例子是我们 Mac 上编译 Libav 的例子

```
/configure --prefix=/opt/av --enable-shared --enable-pthreads --enable-gpl --enable-version3 --enable-hardcoded-tables
```

FFmpeg 的命令行也类似。实际上，最重要的是 --enable-libx264。

当然，configure 完了后需要 make && make install，这是 UNIX 在软件编译安装的标准步骤，不多说。

好了，有了多个 Libav 和 FFmpeg，怎么让 FreeSWITCH 找到它呢？

到 FreeSWITCH 源代码目录下

```
cd src/mod/applications/mod_av
```

创建如下 Makefile (如果已经有了就替换掉)

```
#AV=/opt/av/  
AV=/usr/local/Cellar/ffmpeg/2.8.6  
LOCAL_CFLAGS=-I$(AV)/include  
LOCAL_LDFLAGS=-L$(AV)/lib -lavcodec -lavformat -lavutil -lavresample -lswscale  
LOCAL_OBJS=avcodec.o avformat.o  
  
include ../../build/modmake.rules
```

然后在mod_av下面执行`make install`(你的 FreeSWITCH 必须正常编译, 过一遍啊, 别说我没告诉你)

一切顺利的话, 你就可以在 FreeSWITCH 里面`load mod_av`了。

什么? 还是不行?

在 Linux 上, 还需要设置动态库的加载路径。最简单的办法是启动 FreeSWITCH 的时候加到环境变量里, 如, 可以用以下命令启动 FreeSWITCH:

```
LD_LIBRARY_PATH=/opt/av/lib /usr/local/freeswitch
```

另一种就是放到`/etc/ld.so.conf`或`/etc/ld.so.conf.d`里, 并执行`ldconfig`。如果你需要经常切换多个版本, 还是用环境变量来得快些。关于`ldconfig`, 也是 UNIX 开发者的必备修养, 不知道的自己 Google 吧。

好了, 正常`load mod_av`后, 你就可以尝试使用它提供的 H264 编码, 录音、录像、播放视频等功能了。

当然, 如果你使用 Git master 代码, 可能直接就什么也不用动, `yum install ffmpeg-devel`, 然后`make mod_v-install`就行了, 我没有试过。大家可以测一下如果使用 CentOS 自己带的库有没有问题, 也可以告诉我各种版本的 CentOS 都带了 FFmpeg 的哪个版本。

4.5 如何录制喜欢的电视节目

杜金房/2015.12.15

今天，说个技术话题。

我上电视了。但是，我不能看。虽然家里有电视机，但房子住了好几年了，从来没有开通有线。前一阵，联通搞活动，装了个联通牌的机顶盒，也算是过上了小康生活。

小康归小康，其实我们家也不大看电视。

日子一天天过去，忙碌而又充实。

偶有小波澜——我上电视了。将在烟台台黄金时段播出。

拿起遥控器找台，几百个频道的样子，找了八遍，没找到烟台台。

给同学打电话，我上电视了，快快看吧。同学说，不看，还在加班呢。再说了，上电视有什么大惊小怪的，俺老婆经常上电视……

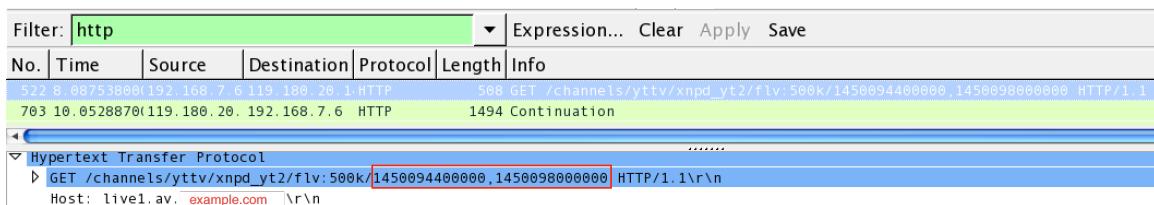
但俺是第一次啊！

上网看。别说，在胶东在线上还真搜到一个页面，烟台 1、2、3、4 套节目全有。

页面用 Flash 做的，有直播,有点播。很卡，但是，先看再说，至少截图发朋友圈是足够了。

终于在漫长的卡顿中看完了。

怎么存下来呢？做技术的，总喜欢钻研一下。网站上有三天内的视频回放，因此，先到视频回放页面找到相应的视频，确定可以播放。然后打开 Wireshark，启动抓包，刷新页面。看到好多 119.180.20.x 的包，可能就是它了。用该 IP 地址作为过滤条件 (`ip.addr == 119.180.20.x`) ,包太多。一般 Flash 的直播都是用 rtmp 协议传送的，用 rtmp 过滤，没看到包，改用 http 过滤，这次看到包了，如下图（注意，真实网址用 example.com 代替了）



到这儿，似乎一切看起来都挺顺利。后面的『1450094400000,1450098000000』部分看起来就像一对时间戳，像起止时间之类的。

打开终端，输入以下命令（Mac 和 Linux 上）

```
$ date +%"s
1450090255
```

可以看到时间戳是 10 位，单位是秒。那么，上述网址中的时间戳应该就是毫秒了。试一把，打开 Ruby 交互终端，使用 Time.at 函数可以看到如下输出：

```
$ irb
> Time.at(1450094400)
=> 2015-12-14 20:00:00 +0800

> Time.at(1450098000)
=> 2015-12-14 21:00:00 +0800
```

Yeah，确实是我们的黄金时间。

先用 VLC 试一下能不能播放。把图中的地址拼成：

```
http://live1.av.example.com/channels/yttv/xnpd_yt2/flv:500k/1450094400000,1450098000000
```

在 VLC 中打开以上网址，发现可以正常播放的。

成功一半了。

VLC 自有录像功能，但我从来没用过。尝试一下 rtmpdump，由于直播地址不是 rtmp 的，rtmpdump 不好用。换 1avconv1：

```
/opt/av/bin/avconv -i "http://live1.av.example.com/channels/yttv/xnpd_yt2/flv:500k/1450094400000,1450098000000"
-acodec copy -vcodec copy x.flv
```

开始下载了，几秒钟后在 flv 中打开 x.flv，发现确实是我们要的时间段的视频。

有我的那块当然没有一小时那么长，因此，缩小了一下范围：

```
/opt/av/bin/avconv -i "http://live1.av.example.com/channels/yttv/xnpd_yt2/flv:500k/1450095270000,1450095540000"
-acodec copy -vcodec copy x.flv
```

当然，x.flv 也可以换成 x.mp4，区别只是 flv 的文件信息是写在头部的，可以直接在下载的过程中播放，mp4 是写在文件尾部的，只能等下载完毕后再播放。

注意，我用了 avconv，你应该也可以用 ffmpeg。avconv 本来也是 ffmpeg，后来，Fork 了。技术圈那些事，不多说了。感兴趣的可以自己搜一搜 ffmpeg vs avconv。

好玩不？

4.6 我看首届胶东开发者大会

杜金房 / 2015.12.14

双 12 那天，我去参加了首届胶东开发者技术大会。先来看一看阵容吧：

12 月 12 日，由烟台易云网络科技有限公司、蓝色智谷·互联网+综合体、胶东在线联合主办，杭州聚势企业管理有限公司、胶东开发者部落协办，由烟台广播电视台、胶东在线网站、烟台论坛全程媒体支持的“2015 年胶东开发者技术大会暨胶东开发者部落启动仪式”在港城成功举办。烟台高新区投资开发有限公司副总经理于柯先生、烟台高新区投资开发有限公司项目开发部经理孔向阳先生、鲁东大学信息与电气工程学院院长刘启明先生、胶东在线网站副总编辑殷守龙先生、胶东在线网站技术中心主任王中群先生、烟台论坛技术总监张欣先生莅临现场。

当然，我是去打酱油的。

故事还得从早些时候说起。其实，要这么说，这个想法也有一两年了。我在北京创业，这些年没好好照顾家人和孩子，深感不安。北京不管是从技术环境和商业环境，都比烟台好得多，但说到自然环境，烟台就有优势了。看着孩子一天天长大，后来还是决定回到烟台。

回烟台首先要考虑的是做什么。这些年，我一直做 FreeSWITCH 开源软件，其实本身是不受地域限制的，我也经常在家里远程办公。但，如果真来烟台办公司，那就是另一回事了。市场我们倒不是太担心，毕竟这些年我们做的一个小众市场，客户大部分都是线上来的，在哪里都一样。但在烟台能不能找到合适的开发人员，就是另一回事了。

因此，两年前我就在考虑在烟台这边搞搞技术圈的交流活动之类的。搞活动，我其实比较热心的。很久前，在北京，经常参加 CSDN 和 infoQ 的活动，也去西安参加 Google Developers 活动，当然，我们也自己搞 FreeSWITCH 沙龙活动。FreeSWITCH 沙龙每年一届，今年是第四年了。前两届都是在车库咖啡，去年换到了北邮科技酒店，今天到了鹏润国际酒店，感觉技术社区一年比一年壮大。

来烟台当然不能搞那么大的活动，但可以从十个八个的，甚至三五个人开始搞起，慢慢地，圈子就大了。

只是，可惜，这些年一直在忙公司的事情，竟一直没有开始。

一次偶然的机会，认识了衣明志先生。衣是微软 MVP，也是从烟台去了北京，后来又来烟台创业。有这么一点相似性，技术人员就很容易聊到一起。当聊起烟台的技术氛围时，我们的看法想法也非常一致。只是，没有想到，胶东技术开发者大会这么快就来了，而且，如此大规模，如此高规格。

我当然不能错过机会啊，拿了几本《FreeSWITCH 权威指南》去大会上赞助一下奖品，顺便做做广告:)。其实做广告倒不是重点，烟台搞 FreeSWITCH 的也比较少，只是，作为一个程序员，天生就喜欢这种圈子。

由于前几天在 FreeSWITCH-CN 的微信公众号上发了个消息，还有人以为我要上台讲 FreeSWITCH，从很远的地方赶到烟台。看到这些，心里暖暖的。

大会上的演讲内容也受益匪浅。高春辉讲了他做 IP 库的那些事；桂老师给大家演示了微软的 Kinect 技术，让我们开了眼界；卢老师从 ReactJS 的基础知识，扩展到 Facebook 最新的开源框架 ReactNative 中去，为我们讲解了 React 的“前世今生”，我们团队也正在研究并在一些小项目中使用了 React 技术，受益良多；胡老师的 DevOps 也让我们开了眼界，本身我们这些草根程序员根本就不遵守什么章法，还总以敏捷自居；衣总的微服务模式也一直是我们在研究和实践的模式，心有戚戚。

关于大会有更详细的报道，由于微信不允许直接链接其它的公众号，所以，感兴趣的可以关注我们的公众号(FreeSWITCH 中文社区)，回复『胶东』查看。

会上，有幸接受了烟台电视台的采访，不让提前打草稿就直接对着镜头喷，这还是第一次。

这是第一次开发者大会，是一次团结的大会，胜利的大会。相信，烟台的技术氛围会越来越好。

4.7 与跟 FreeSWITCH 不大相关的故事

Seven/2015.03.15

想起来几个段子，都是我上一家公司时发生的真事，虽然跟 FreeSWITCH 无关，但我是在上一家单位跟 FreeSWITCH 接上头的，大家欢乐一下。

有次我写邮件：Hi Brain……，正好被老板（外国人）看见，他问我说：这个人叫 Brain（脑子）？我说是啊，后来查了查人家叫 Brian（布莱恩），以前一直叫人家脑子……汗

我上一家公司是外企。去面试时，CTO 是大胡子老外。现场在 Amazon 上给我开了台云服务器（Linux），分给我台 Windows 让我登录上去做题。结果搞了半天上不了网。我用极不熟练的英语问他—Is this part of the interview（这是面试的一部分吗）？他有点不好意思地说，No。

第二季：终于连上网可以做题了，但服务器在国外，在 Shell 上打字非常慢。为了怕突然中断，我把所有我打的命令都存了下来。最后，题都做完了。面试官问我，你写得脚本呢？我说，网速太慢了。我把命令都粘贴到本地的记事本了。后来看了看题，天，原来最后一句是说要把所有安装过程写成脚本，一身汗。还好当时网络慢，要不我连个 Windows 记事本文档也没有。（注：最后面试通过了，要不也没有这个段子了，呵呵）

上面是我的故事，下面是以前的小伙伴们发来的：

接到 CTO 发给我的 Offer 之后，我在 MSN 问他入职前需要准备什么。他问我知不知道 Rails 和 Restful，我说我刚接触 Rails，可以继续学，至于 Restful（以为指的是会不会休息…汗，实际指一种接口技术），我回答我一直都懂 Restful。

我面试时老板（外国人，懂中文）说用英语聊吧，我说好。结果全程用汉语下来了。

有一次，一个程序员去面试，在桌子上做题，做着做着，他就困了，睡着了。那个香，愣是一个小时没醒。我们都不好意思打扰人家，恨不得给疲劳的小伙子安个床。从那时到现在，对程序工程师佩服得不得了。真拽。

想起来，以前有个同事叫 Ryan，我总叫人家 Rain，Ryan 说，我啥时间改了名了？

我想起技术组加班，然后，谁鼓励了一下— You are all hard (硬！少儿不宜，猜本想说 hardwork 的)。我从中午笑到睡觉，大肠都憋涨气了。

有一个男同事，天天吃完午餐在会议室喝茶，用那种小巧的有诗意的杯子，特别小，装满了喝下去还占不到口里的三分之一。还有一个女同事爱抚琴。我也偶尔喝点，但很快就走了。有一个萌小伙，天天也跟着蹭杯喝。然后在旁边一脸茫然又无比服气地听他们论道。一年后，帅哥美女结婚了。萌小伙才知道自己原来是白炽灯。

还是那位同事，传说是他的座位比较神。后来换给了另一个技术组的同事。半年后发喜糖了…，女神也是我们公司的。

想起一个笑话，刚开始办公室人不多，每天早上都爱学老友记里面的打招呼方式，所以很长一段时间，经常听到：Hi, gays (同志(同性)们，本来是想说伙计们-guys)。

最后一个是我的故事，倒是跟 FreeSWITCH 有点关系。

一次到一家大公司去测 FreeSWITCH。人家说得杀毒，我说 Mac 电脑上没杀毒软件，人家说那不让进。层层审批之后结果进了。出门时，说得格式化硬盘。我说那不可能。结果说那也得每个文件都检查。我只好把每个文件夹用 ls 命令给他们列，告诉他们每个文件是干什么的。要知道，有个文件夹下有几十个子目录，还有的有成百上千个文件。结果他们快吐了，问另一个人 (双人检查)，我看这没问题了，你觉得有问题吗？那人说，我觉得也没问题，快走，快走……

4.8 盘点《FreeSWITCH 权威指南》

2016-01-25/Seven Du/FreeSWITCH 中文社区

快过年了，盘点一下。本来去年年底就盘点了，但是，嗯，没保存好，丢了，我只好再重写一遍，所以，一直就拖到现在。

大约有一两个月了吧，有好多网友都反映各大电商网站《FreeSWITCH 权威指南》缺货。还好，我之前买了很多还没有卖完，因此，好像只有我这里有纸版的书了。

在这本书的网站上 (<http://book.dujinfang.com>) 收录了大部分的购买链接。其中，有赞商城是我自己经营的，只能用微信购买。随着各大网站的缺货，最近有赞商城出货量也有所增加 (题图)。其实呢，我这里也没有多少书了，因此我不怀好意地提了提价，现在，终于可以以原价出售了。当然，还是包邮。

但即使纸书没有了，大家也都可以买到本书的电子版，看起来方便，也便宜。

其实卖书根本就是不挣钱的，尤其是这种小众的技术书。书是我自己写的，但是我想有书，也得自己买，当然折扣比网上要大一些。虽然有一些版税，但是我也送出去不少书，所以，算起来，基本都是持平的。对于那些明知道可以从我这里要一本却还是坚持自己下单买了书的朋友，我心里是由衷的感激。

当然，今天主要不是为了说钱的事。写本文的初衷主要是缘于前一阵看了一下各大电商网站上大家的留言，感觉总体还是挺不错的。其中有一些人提到书的印刷质量不好，我觉得可能是个别现象，至少我“开光”过很多本书，没发现质量不好的问题。另外，也有几个读者说到书的内容不够深入，这个我在之前也解释过。主要是，作为国内第一本 FreeSWITCH 书，在写作的时候主要还是考虑多涵盖些内容。而且，书已经很厚了，因此，也没法再特别深入。其实，如果深入的话，书中的很多章节，每个章节都几乎可以写成一本书。但那样写几乎是不现实的。

以前也有读者说过，其实前几章都没必要写，把那些基础性的知识篇幅省下来，可以直奔主题。当然，我是有不同看法的。我写作时主要的想法还是，让没有专业基础的人也能看得懂。也就是说，他可以作为刚毕业甚至未毕业的大学生的一本学习教材，而不是面对通信搞了十来年的老鸟。

其实呢，这本书并不只是入门级的，其中有很多深入的内容，都是我从实践中来的，在别的地方都是看不到的。我花了很多心思把这里的东西讲明白，我相信即使通信干了很多年的老鸟也是感觉很有帮助的。事实上，很多我的老朋友和新朋友也都是这么跟我说的。

总体来说，我觉得在内容方面我把握的还是不错的。如果你多读几遍，定能理解我的苦心。看到有的读者接连买了好几本，我心里也有些欣慰。

另外，我在书中添加了很多的脚注，对一些问题进行了说明，并有很多指向深入学习的资料的链接，就是为了给有心人更深入学习的参考。略遗憾的是，大家在评论中都没有提到这一点（就好像没注意到一样）。

再回到评论上。总体上，大部分都只是应付一下购买的流程，但也确实有一些读者是真心写的。有几个读者明显跟我认识，但大部分的读者基本在买书的时候都是不怎么了解我的。不多说了，仁者见仁。在此，特摘录一些评论，与大家共勉。

一本作者心血之作 非常有幸可以获取改书试读名额，让我可以学习这门一本好书。这绝对不仅仅是一本介绍如何熟练掌握 FreeSWITCH 的书籍，作者围绕 FreeSWITCH 讲解了 VOIP 方方面面，也穿插了相关协议分析和媒体处理，常用 Debug 方法，还有 VOIP 相关业务方向。丰富的讲解了 FreeSWITCH 几乎所有知识面。作者还分享了自己的学习方法，后面还专门使用几个章节深入讲解 FreeSwitch 架构以及源码分析。我把这本书推荐给每一位 VOIP 从业人员，就算你不从事 FreeSwitch 相关工作，学习这个成熟的软件架构对你来说也是一个非常有价值的。

不错的好书 看了下样张，觉得书的技术性很强，很适合我这样的技术极客（自我感觉我是这样的人）拿来好好的研究研究。尤其是 FreeSWITCH 可以方便的在 Linux 下进行部署，这样我就可以继续扩充我的计算机的功能了。感觉这样可以更好的理解计算机及整个通讯网络的一些理论。学习起来也更具体更深入！！！

工作中大量使用 FreeSwitch，FreeSwitch 构建了非常具备伸缩性，功能非常强大，可以感觉是一个具备强大社区驱动成长的产品。我们知道国内 voip 相关政策问题，国内 voip 行业没有国外普及，但通信国内开放是迟早的事情。在学习 FreeSwitch 中，中文资料相对较少，本书作者杜金房的个人博客给我很大的便利，知道杜金房先生是 FreeSwitch 专家，对 FreeSwitch 国内普及和开发都做了较大贡献，老早就知道杜金房先生要出 FreeSwitch 的书，当时非常期待，没想到过这么久终于盼来了，非常希望能有机会参读这本书。

书到的相对比较早，正在学习

0 基础学会网络电话开发 项目需要开发一个网络电话，所有需要的功能，这本书都可以找的到，真赞

做呼叫中心的必备书籍 买了 kindle 版本，感觉 kindle 版本对代码的排版不是太好，又买了多看版本。这本书讲了很多电话、电信相关的入门信息，对于初学者使用 FS 是遇到的一些不明白的概念，在这里可以看得很明了；并且，作者说了很多实际工作场景遇到的问题，这是很难得的！谢谢杜先生！

电子版本 还不错，介绍的很全面。。。但怎么不能用电脑版本的阅读器看呢？手机看太累了

哈哈终于有电子版的，实体版和电子版的都有了 东西确实是好东西，特别是有心做语音平台的开发者。

非常实用 非常实用非常好的一本通信方面的书，用通俗易懂的语言讲解了通信的基础知识，非常适合入门学习。学习通信强烈推荐。

FreeSWITCH 方面手册书 介绍的很全面，非常适合入门看，尤其是开头讲解了电话交换的发展历史。关于协议方面介绍的也很详细。

挺好挺好挺好真的挺好……

京东就是给力，东西不错，以后还会继续支持

总体还是不错的一本书，也体现了作者的职业素养。

FreeSWITCH 权威指南，不错，看看再说吧

这本书真的很好，，我很喜欢

好评好评好评好评好评

称得上是 FreeSWITCH 国内最好的一本书了，专业必备

专业扩展学习，很有用

写的好，专业书籍，对使用很有帮组

好好好好好好好好好好好好

好书，可全面了解程控交换机的机制。

书不错，通讯网络工程必备

好好。。。

还不错还不错还不错还不错

内容丰富，非常好的书

不错。

挺好的，书的内容也很好。很有学习的价值

书不错!!!!!!

实用。。。。。。。。。。

对于初学者来演讲的相当的不错，是一个入门的不错的选择。

东西不错，有需要会再光顾

fs 经典，正品书籍，值得拥有，呵呵

还行老杜写的不错还行老杜写的不错

买来虽然没看，放在那当摆设

还可以。京东价格实惠

首先送货速度还可以，书的内容也值得推荐

书很好，买了电子版，又来买实体书

书还是不错的，希望以后能赚大钱

东西不错我已经买了很多次了

印刷质量不好，书上的油墨容易花纸

专业必备帮助很大

freeswitch 方面的书不多，最近在搞语音通话方面，看一看如何

工作用图书，希望能有所帮助

推荐购买，不错不错不错

不错，活动买的，物美价廉，大爱

给了很多帮助，是不错的书

好好好好好好好很好

看这书可以零基础，不错的一本书

对 Freeswitch 有了初步了解，很油帮助，多谢!!!!!!

hao !!!!!!!!!!!!!!!

挺不错的，这个领域就只有这本书可以看了。

看是没问题的，但是这么贵的书切成这样，还是有点不爽

印刷质量太渣了，在看书的过程中，手一抹字，就花了，就跟刚印的东西，油墨还没干似的。买了这么多书，第一次遇到这种搓事情。。

开源软交换方面的书，尤其是中文的，实在是太难找了，像这本介绍专门全面的实在是很不错，本身这个开源软件换框架也是比较优秀的。

还不错，有空时候翻翻看看

写的很系统，可读性好

刚收到，还没来得及看呢

这本书很适合入门者哦哦

这书不适合阅读，适合有问题翻阅，学习的话还是动手的好

我表示写的很好!!!

非常好用，又便宜，搞活动买的很抵，太喜欢了

FREESWITCH 国内第一本书，有一些讲得不够深入

freeswitch 的参考书太少

不错，看了很久了，书入门超级合适

写得还是不错的了，开源的少有中文版的资料。

这已经是第 5 本了，还不错

书买了，还没来得及

其实 IT 技术没必要发行实体书，会员制做一个在线电子版，还可以随时修正。推进前沿技术时，也不要钻得钱眼里太深了。

还不错!!!!

第一次选错了，害的送货的师傅走了两趟，真是不好意思了。

还可以啊啊啊啊啊啊啊啊啊

赞一个!!!!!!!!!!

书很不错，送货也快

书上有些东西写的不是很清楚，有些实例需要自己去多推敲才能理解，没有解释清楚

正在看着，看完再来说说。

书籍纸张很好，确实是正版，内容不错，总的来说对学习挺有帮助的。物流很快，有现货的下单第二天就到了。客服什么的也很不错，态度好，很专业，有问必答。但是现在怎么不能先发有货商品了呢，订单里有的缺货都得等到齐了才发。

京东现在的送货速度太赞了，书比较基础，适应面广，不过不够总觉得还不够深入。

入门级的好书，值得拥有学习。

一直没有中文版的，买来学习。

写的很基础，入门不错……

不错不错，不错不错，

书不错，正是需要的，由浅入深，适合新手、专业用户

中午下单,下午送到,惊人

不错的一本书，能从里面得到最专业的只是，可能是翻译版本，有些地方还是不太通，不过总体不错。

作为入门书籍手边翻翻还可以，就这么多吧

写得很好！

送货快很好看上去正版

仰慕好久了，终于买下来了，努力学习中

从基本的讲起，该说的基本都说清楚了。

书不错，但有损害，

这本书的包装质量都没问题，要说的就是代购速度啊，两星期了才收到书，我都快给忘了

挺不错的一本书，值得好好学习学习。

老公买的，没听说好不好。但是好久才送来。

适合动手能力强且有一定相关经验的人入门。NAT 部分内容偏少没有深入，如果多加入一些具体的 NAT 解决实例最好。

专业细致，值得看！！

《freeswitch 权威指南》对 freeswitch 的原理、应用及开发讲述的非常详细，还介绍了相关的交换知识、呼叫中心开发知识，是指导 freeswitch 初学者的非常有用的参考书。

不错的书籍，是学习专业软交换开发的优秀书籍，强烈支持 MR 杜！!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

内容很丰富，虽然我当手册来用了很多没有看，但这本书称得上是“权威指南”

很好很很全面…

比较适合入门学习，内容安排节奏好

比较适合待入门者，深一点的内容就比较少了。

不错，非常好的一本书

刚发售就买了，价格比较优惠。

很不错，受益匪浅

很好的一本书，很实用的，就是价格比较贵

书籍很好包装不错书籍给人的感觉不错性价比很高书籍是正版纸张好排版不错我很喜欢

很好很好很好很好很好很好很好很好

好书，值得一读

很不错的一本书，值得拥有

通过这本书，揭开了软交换的面纱，有种豁然开朗的感觉。

很不错！专业

开源通信难得一见的好书，需者赶紧买！

昨天晚上下的订单，今天早上 9: 09 收到书，速度真快，赞一个！不过书还没有时间看，但绝对是正版。对当当上的这次购书非常满意

内容详实，也算是 Freeswitch 领域专业的教材了

内容不错，对行业了解有帮助

还不错，又给同事买几本

书本内容很全很宽泛，不够细化，很多细节内容没有详细描述，网上也找不到相关资料，入门书籍。

在本书出版后不久，我们搞了一个写书评的活动，大家写得都不错，回头我们会把那些书评都在这里发出来。

与大家共勉。

第五章 科普解惑

科学技术是第一生产力。

5.1 PSTN 起源与发展

中国移动开通 VoLTE 了，让我们盘点一下 VoLTE 是怎么来的…

PSTN (Public Switched Telephone Network) 的全称是公共交换电话网，就是我们平常打电话所使用的电话网络。

第一次语音传输是苏格兰人亚历山大·贝尔(Alexander Graham Bell)在 1876 年用振铃电路实现的。在那之前，普遍认为烽火台是最早的远程通信方式(当然，还有更早的，那就是通信基本靠吼的年代了)。

当时没有电话号码，相互通话的用户之间必须有物理线路连接；并且，在同一时间只能有一个用户可以讲话(半双工)。发话方通过话音的振动激励电炭精麦克风而转换成电信号，电信号传到远端后通过振动对方的扬声器发声，从而传到对方的耳朵里。

由于每对通话的个体之间都需要单独的物理线路，如果整个电话网上有 10 个人，而某人想要与另外 9 个人通话，他就需要铺设 9 对电话线。同时整个电话网上就需要 $10 \times (10-1)/2 = 45$ 对电话线。如图 5.1 所示，图中只显示了 6 个人的情况：

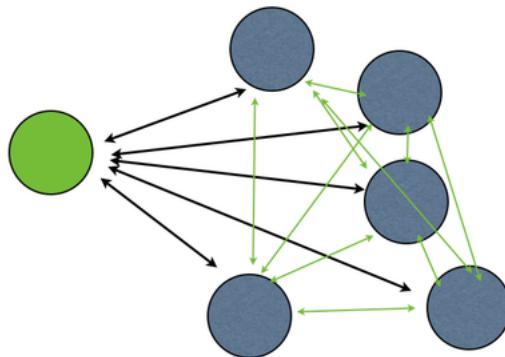


图 5.1: 6 个人的通信网络

当电话用户数量增加的时候，为每对通话的家庭之间铺设电话线是不可能的。因此一种称为交换机（Switch，又称 Exchange）的设备诞生了。它位于整个电话网的中心，用于连接每个用户。用户想打电话时，先拿起电话连接到管理交换机的接线员，由接线员负责接通到对方的线路（如图 1-1）。这便是最早的电话交换网，交换接续工作全部由工人完成。

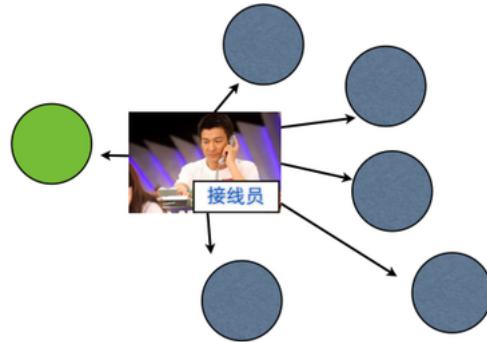


图 5.2: 电话交换网、接线员节省了线路

1889-1891 年，美国阿尔蒙 · B · 史端乔 (Almon B Strowger) 发明了步进式自动电话交换机。有趣的是，它发明自动交换机的目的并不是为了把接线员从繁忙的人工交换机上解放下来，而是来源于另外一则故事：他本是一个殡仪馆老板。他发觉每当城里发生死亡事件，用户给“殡仪馆”打电话时，接线员不知有意还是无意地总会把电话接通到另一家殡仪馆。这使史端乔非常郁闷，发誓要将电话交换自动化。功夫不负有心人，史端乔凭他那过人的聪明和毅力，终于发明了一种步进式的自动电话交换机，并申请了专利。人们为了纪念他，也称这种电话交换机为“史端乔交换机”（图 1-3）。这种交换机的特点是由用户话机的拨号脉冲直接去控制交换机的动作，属于“直接控制”方式。

以后，又出现了旋转式和升降式的交换机。这类交换机采用了一个叫做“记发器”的部件来接收用户的拨号脉冲，然后通过译码器译成电码来控制接线器的动作，属于“间接控制”方式。采用记发器后，增加了选择的灵活性，从而可以使用交换机的容量得到提高。

1919 年，瑞典的电话工程师帕尔姆格伦和贝塔兰德发明了“纵横制接线器”，并申请专利。这种交换机将过去使用滑动摩擦方式的触点改成了压接触，减少了磨损，从而提高了交换机的寿命；而且，由于采用了贵金属（如银）做金属触点，也大大提高了接触的可靠性。这种交换机的另一个特点是把控制部分和话路部分分开，控制部分由“标志器”和“记发器”来完成，称为“公共控制”。公共控制对对户拨号盘要求低，而中继部署的灵活性大大提高。

瑞典和美国分别在 1926 年和 1938 年开通了纵横制交换机，接着，法国、日本和英国也相继生产出纵横制交换机。

随着电子技术，尤其是半导体技术的迅速发展，人们开始在交换机内引入电子技术，称为电子交换机。由于当时技术条件的限制，仅在控制部分引入了电子技术，而话路部分在较长的一段时间内仍然采用机械触点。这种交换机一般称为“半电子交换机”或“准电子交换机”，区别是后者采用了速度较快的“箔簧接线器”。

1946 年，第一台存储程序控制的电子计算机诞生，对现代科学技术起到了划时代的作用。通过

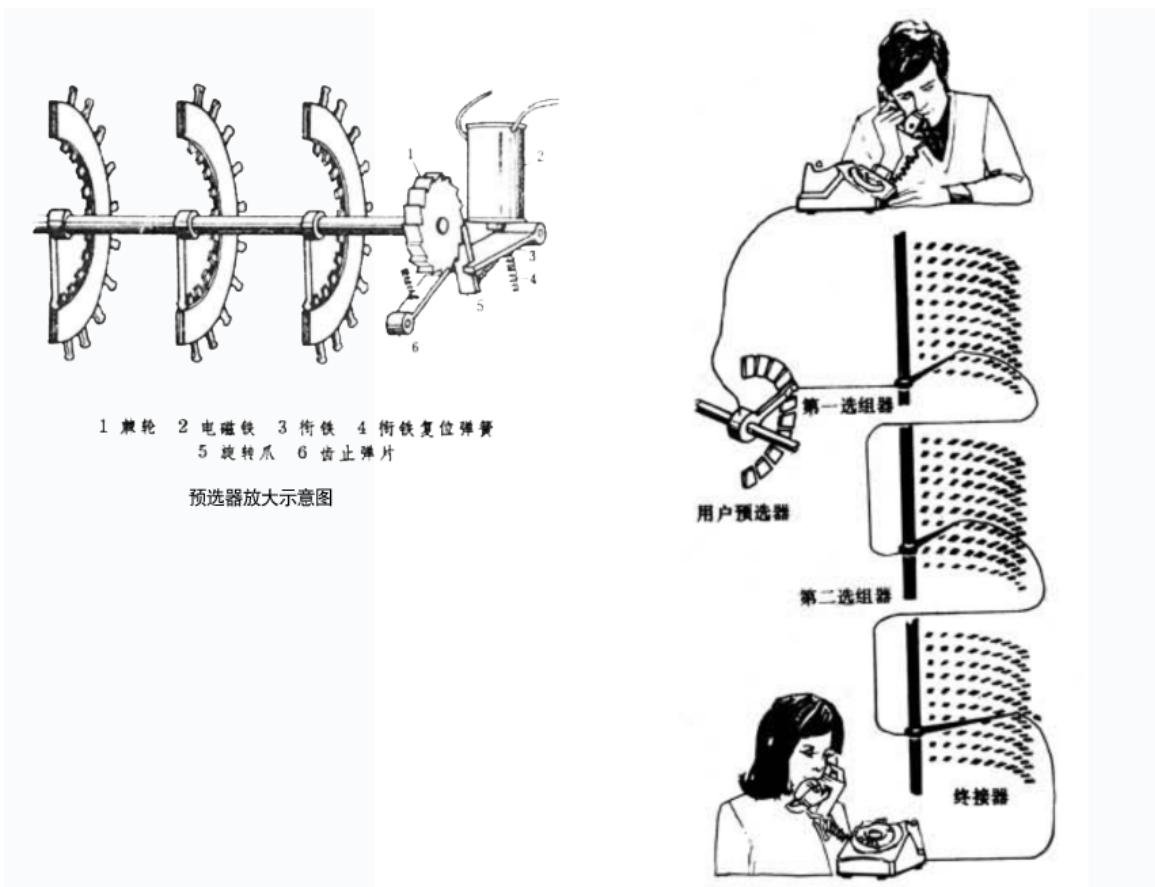


图 5.3: 步进式交换机，4 位拨号电话中继示意图

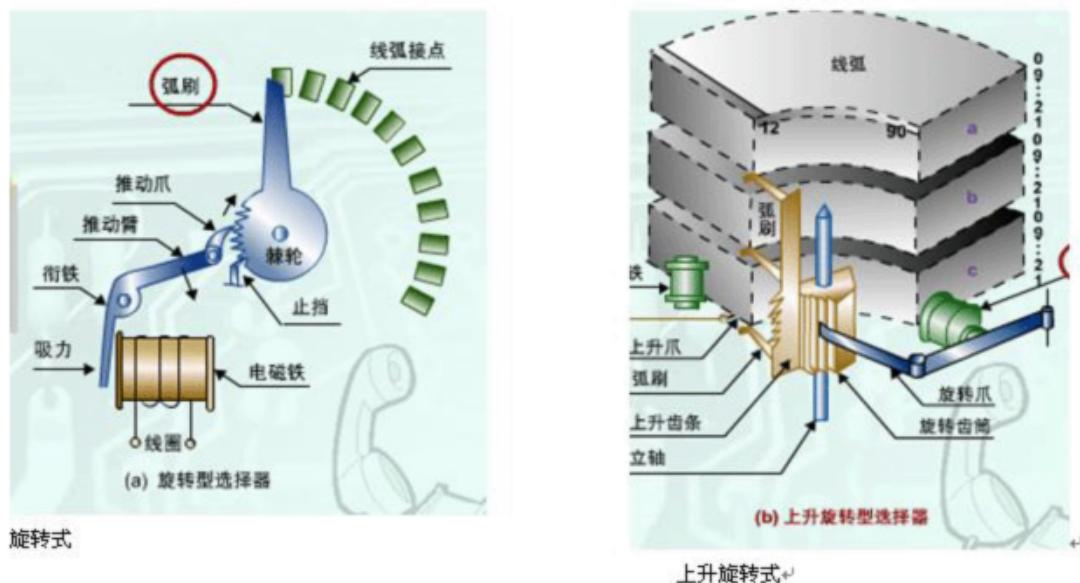


图 5.4: 旋转式交换机

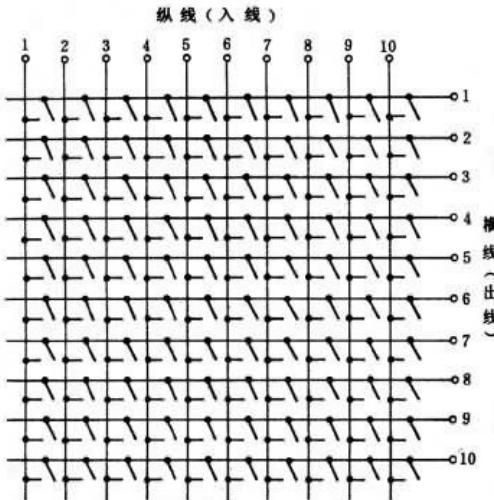


图 5.5: 纵横式交换机, 纵横接线器交叉点示意图

在交换机中引入“存储程序控制”的概念，1965 年 5 月，美国贝尔系统的 1 号电子交换机 (ESS No.1) 问世了，它是世界上第一部开通使用的程控电话交换机。当时的交换机话路部分还保留了机械触点，以“空分”方式工作，因此称为空分交换机。并且，它交换的还是“模拟”信号。

20 世纪 60 年代初以来，脉冲编码调制 (PCM) 技术成功地应用于传输系统中，它通过将“模拟”的信号数字化，提高了通话质量、增加了传输距离，同时，节约了许多线路成本。1970 年，法国开通了世界上第一部程控数字交换机 E10，开始了“数字”交换的新时代。

随着技术的进步和人们对通信要求的增加，世界上许许多多的交换机间也需要相互通信，这些交换机之间通过中继线 (Trunk) 相连，随着电子交换机和程控交换机的发展，便出现了现代意义的 PSTN 网络。PSTN 网络把世界上各各角落的人们都联系在了一起，很显然，有时一个通话需要穿越好多台交换机。

70 年代后期出现了蜂窝式移动电话（当移动电话小到可以拿在手里的时候就开始叫“手机”）系统，是无线电话发展的重大里程碑。而在此之前，虽然无线电通信是在 1895 年发明的，但无线电话却是在 20 世纪初发明了真空三极管之后才出现的。1915 年首次成功地实现了跨过大西洋的无线电话通信；1927 年在美国和英国之间开通了商用无线电话。

专门用于对移动电话进行交换的通信网络称“移动网”，而原来的程控交换网则称为“固定电话网”，简称“固网”。简单来说，移动网就是在普通固网的基础上增加了许多基站 (Base Station，可以简单理解为天线)，并增加了归属位置寄存器 (HLR, Home Location Register) 和拜访位置寄存器 (VLR, Visitor Location Register)，以记录用户的位置 (在哪个天线的覆盖范围内)、支持异地漫游等。移动交换中心称为 MSC (Mobile Switch Center)。

随着分组交换技术的成熟及因特网的发展，人们认识到了将原始的基于电路交换的语音网络与基于分组交换的因特网络进行融合（即语音通信和数据通信相结合）的必要性，因此一个称为 NGN (Next Generation Network) 的概念被提了出来。NGN 是指下一代网络，ETSI 对它的定义是：“NGN 是一种规范和部署网络的概念，即通过采用分层、分布和开放业务接口的方式，为业务提

供者和运营者提供一种能够通过逐步演进的策略，实现一个具有快速生成、提供、部署和管理新业务的平台。”

在 NGN 提出之后，经过数年的研究和探索，人们提出了各种 NGN 的解决方案，但最终基本上都统一到 IMS 技术。IMS (IP Multimedia Subsystem) 运行于标准的 IP 网络上，使用一种基于第三方伙伴计划 (3GPP) 的 SIP 标准的 VoIP 实现方式。IMS 的目标不仅是在现网基础上提供新的业务，而且它还要能提供现在以及未来因特网上能够承载的所有的业务。

当然，IMS 是属于核心交换层的技术，它是全部基于 IP 网络的。但在接入层，目前的语音大部分还是基于电路接入的方式接入的，因此，这就要求 IMS 在接入层要在一定时间内继续兼容电路接入。同时，在移动通信领域，随着移动通信技术的成熟以及众多智能终端的出现，对高速 IP 网格的要求也就越来越迫切。最新的 3G、4G 技术就是应此要求而产生的。未来的通信中要完全取消低效的电路传输及电路交换，而全部集中到 IP 通信上来，也就催生了一个新的无线通信标准 LTE。LTE 的定义是长期演进，在为现代的手机及其他移动设备提供高速的数据通信手段，逐步实现全 IP 交换。

关于通信网络的演进，简单来说，在无线侧体现为从 GSM/CDMA/UMTS 等向 LTE 发展，核心网侧则体现为从电路交换向 IMS 发展。过去几年，围绕 LTE 语音曾经出现过多种观点、技术和演进路线。由于 LTE 标准不再支持用于支撑 GSM、UMTS 和 CDMA2000 网络下语音传输的电路交换技术，它只能进行全 IP 网络下的封包交换。因此，随着 LTE 网络的部署，运营商需选择 VoLTE、CSFB、SVLTE、OTT 等方法之一解决 LTE 网络中的语音传输问题。据报道，中国移动已于 2013 年 6 月发布了 VoLTE 技术白皮书，并计划于 2014 年下半年大规模商用。未来通信网络将走向何方，让我们拭目以待。

本文写于 2012-2014 年，首发于《FreeSWITCH 实战》，比《FreeSWITCH 权威指南》上的版本要早。

5.2 VoLTE 给我们带来了什么？

杜金房 / 2016.02.23

中国移动终于为用户开通了 VoLTE，现在，打电话和上网终于可以在一个通道上跑了。

VoLTE 是什么鬼？

这么说吧，在最早的时候，手机用的是模拟信号，只能打电话。到了 2G 时代，电话技术进入了数字时代，有了 GSM，可以打电话，也可以上网，但两者不能同时用（当然，通过 DTMF 等技术也可以做到同时用，但需要网络侧和手机侧都支持，而那时候根本没有普及），也就是说，你打电话的时候，网络就断了（收不到微信消息了，当然，那时候还没有微信）。

基本的电话通道叫做电路域 (CS, Circuit Switch, 也就是电路交换)，而网络业务属于 PS 域 (PS, Packet Switch, 也就是包交换)。到了 3G 时代，基本可以 CS 域与 PS 域共存了，打电话时不用断网。通常，手机就在 PS 域工作，但是，当打电话时仍然需要切换到 CS 域才能将电话接通，这

个切换过程比较漫长，通常，需要 6-7 秒甚至更长的时间，所以，电话的接续时间比较长（而且切换过程中还可能会掉线）。

时间过得很快，我们终于迎来了 4G 时代。当然，系统的升级换代需要一个过程，所以，最初的 4G 手机还是跟 3G 差不多，打电话时会产生从 4G 到 2G/3G 的切换。后来，也就是前几天，中国移动终于为用户开通了 VoLTE。通过使用 VoLTE，电话也在 PS 域上走（也就是真正的 VoIP），因而，再也不用切换了，电话接续的过程可以缩短到 2-3 秒。

如果你是移动的 iPhone 用户，恭喜你已经可以使用这项业务了。关于如何开通，可以咨询 10086 或搜索网上的其它教程，笔者没有 iPhone，这里就不给大家举例子了。

除了接续速度快，VoLTE 当然还有其它的好处。

一是高清通话。普通的通话使用的是 8000Hz 的抽样频率，而高清通话使用 16000 甚至 48000 的抽样频率，听起来更逼真，有人形容是打电话就如同面对面说话一样。

另一个好处就是视频通话，哈哈，查岗时就能更逼真了。

当然，好归好，如果只有你自己是高清语音，而对方没有，那也没什么用。这就如同你能视频，对方电话不能视频是一样的。

好了，关于 VoLTE 就说这么多，如果想看看 VoLTE 的来龙去脉，可以关注我们的公众号（FreeSWITCH 中文社区）并输入『PSTN』从它的前身开始看。

5.3 模拟信号与数字信号

现实生活中的一切都是模拟的。模拟（Analog）量是连续的变化的量，如温度、声音等。早期的电话网是基于模拟交换的。模拟信号对于人类交流来讲非常理想，但它很容易引入噪声。如果通话双方距离很远的话，信号会衰减，因而需要对信号进行放大。问题是，信号中经常混入线路的噪音，放大信号的同时也放大了噪音，导致信噪比（信号量与噪声的比例）下降，严重时甚至会难以分辨。

数字（Digital）信号是不连续的（离散的）。它是按一定的时间间隔（单位时间内抽样的次数称为频率）对模拟信号进行抽样（图 1-6）得出的一些离散值。然后通过量化和编码过程将这些离散值变成数字信号。根据[抽样定理¹](#)，当抽样频率是模拟信号最高频率的两倍时，就能够完全还原原来的模拟信号。

5.4 PCM

PCM（Pulse Code Modulation）的全称是脉冲编码调制。它是一种通用的将模拟信号转换成以 0 和 1 表示的数字信号的方法。

¹又称采样定理或奈奎斯特·香农定理，见 <http://zh.wikipedia.org/wiki/采样定理>。

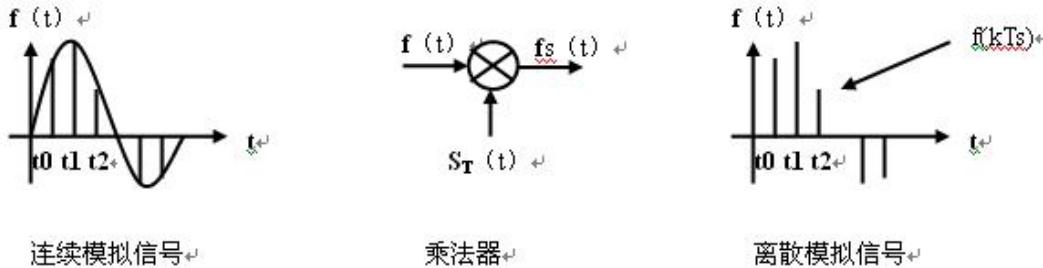


图 5.6: 抽样

一般来说，人的声音频率范围在 $300Hz \sim 3400Hz$ 之间，通过滤波器对超过 $4000Hz$ 的频率过滤出去，便得到 $4000Hz$ 内的模拟信号。然后根据抽样定理，使用 $8000Hz$ 进行抽样，便得到离散的数字信号。使用 PCM 方法得到的数字信号就称为 PCM 信号，一般一次抽样会得到 $16bit$ 的信息。

为了更有效地在线路上进行传输，通常对 PCM 信号进行一定的压缩。通过使用压缩算法（实际为压扩法，因为有的部分压缩有的是扩张的。目的是给小信号更多的比特位数以提高语音质量），可以将每一个抽样值压缩到 8 个比特。这样就得到 $8 \times 8000 = 64000bit$ 的信号。通常我们就简称为 $64kbit/s$ （注意，通常来说，对于二进制数， $1kbit=1024bit$ ，但此处的 $k=1000$ ）。

PCM 通常有两种压缩方式：A 律和 μ 律。其中北美使用 μ 律，我国和欧洲使用 A 律。这两种压缩方法很相似，都采用 8bit 的编码获得 12bit 到 13bit 的语音质量²。但在低信噪比的情况下，μ 律比 A 律略好。A 律也用于国际通信，因此，凡是涉及到 A 律和 μ 律转换的情况，都由使用 μ 律的国家负责。

5.5 我国电话网结构

如图 1-7 所示：图中主体部分为一地市级固定电话网的结构。通常，话机（如 c）通过一对电话线连接到距离最近的交换机上，该交换机称为端局交换机（一般以区或县为单位）。端局交换机通过局间中继线连接到汇接局。为了保证安全，汇接局通常会成对出现，平常实行负荷分担，一台汇接局出现故障时与之配对的汇接局承担所有话务。长途电话需要通过长途局与其他长途局相连。但根据话务量要求，汇接局也可以直接与其他长途局开通高速直达中继。为节省用户线，在一些人口比较集中的地方（如学校、小区），端局下会再设模块局或接入网，用户则就近接入到模块局上。

智能网一般用于实现电话卡、预付费或 400/800 类业务，前几年新布署的 NGN (Next Generation Network，下一代网络，一般指软交换) 则支持更灵活、更复杂的业务。

但是技术发展很快，响应国家“光进铜退”³的号召，直接光纤到户，就全网都能 IP 化了。新技术代替旧技术，给用户带来了五彩斑斓的业务，但老旧的技术永远有让人值得怀念的地方，那就是原来

² 目前我国使用的是 A 律 13 折线特性。基本原理是使用非均匀量化方法和压缩扩张技术得到 13 段折线。感兴趣的读者可在 Google 上搜索“13 段折线”进行更深入的了解。

³ 用户家里的电话线使用铜双绞线。

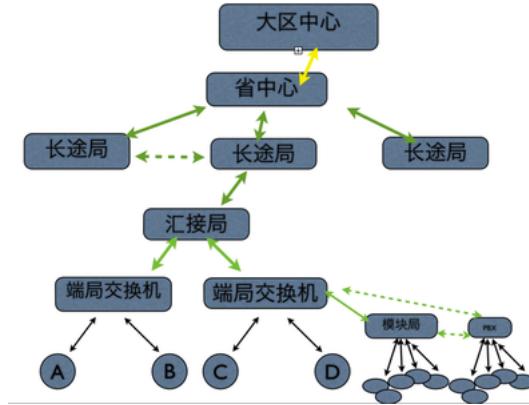


图 5.7: 我国电话网结构

的电话线是带电的，家里停电不影响打电话。但光纤是不能馈电的，因而没有了铜线，停电以后用户就不能打电话了（除非加 UPS）。

在移动网络中，大量部署了 IMS 系统。IMS 系统组网灵活，但涉及的网元和基本概念都很多，我们将在本章的最后一节中讨论。

5.6 电话号码的知识

我们的生活已经离不开电话，打电话就离不开电话号码。但好多人对我们天天使用的电话号码却即熟悉又陌生，因此，在这里，我们也补充一下现行电话网中电话号码的知识。值得注意的是，这些号码的出现大部分与行政区划，也有一些历史背景，但限于篇幅，我们就不深入讨论了。

5.6.1 固定电话号码

现行的电话网中采用 E.164 号码格式。先以我国固话电话的编号规则为例，我们先来看本地号码。

一般来说，在比较大的省市（如北京、上海等）使用 8 位号码，比较小的省市使用 7 位号码，北京的号码一般是 ABCD EFGH，其中，ABCDE 称为一个千群，即该群可以包含 1000 个电话号码，同理 ABCD 称为一个万群。小规模的交换机可能只包含几个千群，而大规模的可能包含几个万群。

如果在不同的省市之间打电话，则称为长途电话，一般需要通过长途局进行路由。打长途电话需要在本地号码前加上长途区号。根据城市的大小不同，我国电话区号的长度一般是 2 位到 3 位，如：

-
- | | |
|-----|----|
| 10 | 北京 |
| 20 | 广州 |
| 21 | 上海 |
| 755 | 深圳 |
| 535 | 烟台 |
-

读者可能会问，北京的长途区号不是 010 吗？答案是——“**不是**”。许多人会错误的认为北京的区号是 010，这也正是作者要写这一节的原因。

由于各地区号的长度不同，而且数字也不规则，因此，为了区分长途电话和本地电话，我国规定，在拨打长途电话前除了要加上区号外，还要在最前面加“0”。因此，“0”称为国内长途字冠，它本身不是区号的一部分。为了帮助读者理解，我们先考虑国际长途的情况。为了区分国内长途与国际长途，我国规定，国际长途要在国家代码前加两个 0。因此，如果拨打美国的号码，如：1-234-567-890，则需要拨 00-1-234-567-890。其中，1 为美国的国家代码，我们稍后再讲到。

交换机的拨号规则遵循短号优先的原则，拨号时不加 0 就认为是本地号码，加一个 0 就认为是国内长途，两个 0 是国际长途。我国的国家代码是 86，因此，一个北京号码的表示是：

86 10 ABCD EFGH

与我国不同，美国的国际长途字段是 011，因此如果从美国拨打中国的号码，需要拨：

011 86 10 ABCD EFGH

类似的，其他国家和地区也有不同的拨号方式，这给电话号码的书写带来了复杂性。因此，在实际写电话号码时，会省略国际长途字冠，而统一用“+”号代替，如：

+86 10 ABCD EFGH

读到这里，读者就可以理解为什么北京的区号不是 010 了吧？另外，我们还知道，使用固定电话拨打异地手机前要加拨 0，同样说明 0 并不是区号或手机号码的一部分。

5.6.2 移动电话号码

移动电话号码，俗称手机号，由于其可移动与漫游的特性，因此与普通固定电话号码略有不同。众所周知，这些号码都是以 1 开头的，按不同的运营商来分配的。如中国移动的号码由 135、136、137、138、139 等开头；中国联通的由 130、186 等开头，中国电信的由 133、189 等开头。它们后面都是 8 位的号码，因此整个手机号共 11 位，这里就不多介绍了。

5.6.3 短号码

有一类特殊的号码称为短号码，这些，如大家熟知的 110、119、120 等，这些属于公益性的号码（又称紧急号码，供紧急情况下使用），拨打都是免费的。

还有一部分是预付费的电话卡类业务，如 200、201、300 等。

另一些短号码如 114 等由于运营商和分拆合并，有的演变成了 116114；还有一些 5 位的短号码专门供一些大的集团如银行等使用，如 95555、95588 等，这些号码不论何时何地，拨打一般只收到市话费。

这些短号码的资源比较紧张，一般不会给个人使用。另外还有其他一些短号码，我们就不多讲了。

5.6.4 800 和 400 号码

800 开头的号码是被叫付费的业务，主叫呼叫这些号码是免费的。这些号码主要由一些大的企业集团使用。这类号码都是虚拟的电话号码，在实际呼叫过程中通过查询数据库转换成真正的目的号码。

但是 800 号码有一个致命的缺点，就是用手机打不通，这主要是电信业务的历史原因（主要原因是不同运营商的网间结算）。随着移动业务的发展，手机用户越来越多，因此，出现了 400 业务。这类业务的特点是主被叫分摊付费，主叫付本地通话费，被叫付长途电话费（如果主被叫不在一个城市的情况下）。400 业务使得手机用户可以呼叫它。

5.6.5 北美电话号码分类计划

作为与国内号码的对比，我们来简单说一下北美电话号码分类计划（North American Numbering Plan⁴）。

加拿大和美国使用北美电话号码分类计划，其区号由 3 位数字组成，本地号码为 7 位数字，1 为长途接入码，即长途字冠（在有的情况下可以省略长途字冠），如，一个完整的号码为：

1 (ABC) DEF-GHIJ

如果是在本地拨打，则可以直接拨“DEF GHIJ”，如果是拨打长途，则需要先拨长途字冠 1 及区号，即：

1 (ABC) DEF-GHIJ

值得一得的是，其中的区号 ABC 如果是 555 的话，除 555-1212 是查号台外，其他的号码都是不存在的。这类号码一般用于电影或戏剧里，防止与真实环境中的电话号码相冲突。

⁴<http://zh.wikipedia.org/wiki/北美电话号码分类计划>。

5.6.6 电话号码的书写格式

电话号码就是一长串号码，但有时候，为了便于阅读，在写的时候常用连字符“-”、括号、空格等将数字分开，如上一节我们看到美国电话号码的格式。

对于国内号码的书写，一般采用如下方式：

(010) ABCD EFGH (没有国家代码，虽然 0 不是区号的一部分，但是，习惯了)

+86 (10) ABCD EFGH (固话，8 位，国际号码格式)

+86 (535) ABC DEFG (固话，7 位，国际号码格式)

+86 139 ABCD EFGH (手机，国际号码格式)

当然，具体的写法没有统一的规定，只要让看到号码的人知道怎么拨打就行了。

5.7 时分复用与局间中继

5.7.1 时分复用

通过将多个信道以时分复用的方式合并到一条电路上，可以减少局间中继线的数量。通过将 32 个 64k 的信道利用时分复用合并到一条 2M ($64k \times 32 = 2.048M$ ，通俗来说就直接叫一个 2M) 电路上，称为一个 E1 (在北美和日本，是 24 个 64k 复用，称为 T1，速率是 1.544M)。在 E1 中，每一个信道称作一个时隙。其中，除 0 时隙固定传同步时钟，其他 31 个时隙最多可以同时支持 31 路电话 (有时候会使用第 16 时隙传送信令，这时最多支持 30 路电话)。

5.7.2 局间中继

这些连接交换机(局)的 2M 电路就称为局间中继。随着话务量的增加，交换机之间的电路越开越多，目前通常的做法是将 63 个 2M 合并到一个 155M ($2 \times 63 + P = 155$ ，其中 P 是电路复用的开销) 的光路 (光纤) 上，在 SDH (Synchronous Digital Hierarchy，同步数字传输体系) 技术中称为 STM-1 (Synchronous Transfer Module，同步传输模块)。

5.8 信令 (Signaling)

用户设备 (如话机) 与端局交换机之间，以及交换机与交换机之间需要进行通信。这些通信所包含的信息包括 (但不限于) 用户、中继线状态，主、被叫号码，中继路由的选择等。我们把这些消息称为信令 (Signaling)。

5.8.1 信令分类

信令主要有以下几种分类方式：

按信令的功能分

- 线路信令：具有监视功能，用来监视主被叫的摘、挂机状态及设备忙闲。
- 路由信令：具有选择功能，指主叫所拨的被叫号码，用来选择路由。
- 管理信令：具有操作功能，用于电话网的管理和维护。

按信令的工作区域分

- 用户线信令：是用户终端与交换机之间的信令。它包括用户状态（摘、挂机）信号及用户拨号（脉冲、DTMF）所产生的数字信号，以及交换机向用户终端发送的信号（铃流、信号音）。
- 局间信令：是交换机和交换机之间的信令，在局间中继线上传送，用来控制呼叫接续和拆线。

用户线信令少而简单，中继线信令多而复杂。

按信令的信道分

- 随路信令：信令和话音在同一条话路中传送的信令方式。
- 公共信道信令：是以时分方式在一条高速数据链路上传送一群话路的信令的信令方式。

随路信令信令传送速度慢，信息容量有限（传递与呼叫无关的信令能力有限）。而公共信令传送速度快、容量大、具有改变或增加信令的灵活性，便于开放新业务。

其他分类

其他的分类方式有带内信令与带外信令、模拟信令和数字信令、前向信令和后向信令、线路信令和记发器信令等，我们在这里就不多解释了。有兴趣的读者可以自行搜索相关的关键词进一步学习。

5.8.2 用户线信令

用户线信令是从用户终端（通常是话机）到端局交换机之间传送的信令。对于普通的话机，线路上传送的是模拟信号，信令只能在电话线路上传送，这种信令称为带内信令。话机通过电压变化来传递摘、挂机信号；通过 DTMF（Dual Tone Multi Frequency，双音多频。话机上每个数字或字母都可以发送一个低频和一个高频信号相结合的正弦波，交换机经过解码即可知道对应的话机按键）传送要拨叫的电话号码。另外，也可以通过移频键控（FSK, Frequency Shift-keying）技术来支持“主叫

号码显示”，俗称“来电显示”（Caller ID 或 CLIP，Caller Line Identification Presentation，主叫线路识别提示）。

与普通电话不同，ISDN（Integrated Service Digital Network，综合业务数字网）在用户线上上传送的是数字信号。它的基本速率接口（BRI，Base Rate Interface）使用 144k 的 2B+D 信道—两个 64k 的 B 信道及一个 16k 的 D 信道。其中 B 信道一般用来传输语音、数据和图像，D 信道用来传输信令或分组信息。

2B+D 的 ISDN 最初为了解决用户线上的语音与数据与同步传输问题，野心勃勃。但事实上，2B+D 的 ISDN 并不像传说中的那么美，而且需要专门的 NT1 终端设备，在我国并没有发挥出它应有的作用，后来很快被 ADSL 技术取代了。

5.8.3 局间信令

局间信令主要在局间中继上传送。一般一条信令链路通常只占用一个 64k 的时隙。一条信令消息通常只有几十或上百个字节，一条 64k 的电路足以容纳成千上万路电话所需要的信令。但随着技术的进步，话务量的上涨以及更多增值业务的出现，完成一次通话需要更多的信令消息，因此出现了 2M 速率的信令链路，即整个 E1 链路上全部传送信令。

目前常见的局间信令有 ISDN PRI(Primary Rate Interface，基群速率接口) 信令和七号信令。PRI 是在跟话路同一个 E1 上传送的，通常使用第 16 时隙，而 0 时隙传同步信号，其他 30 个时隙可传输通话信息，因此又称为 30B+D（与上面的 2B+D 相对）。

与 PRI 信令不同，七号信令除可以与话路在同一个 E1 上传送外，还可以在专门的用于传送信令链路的 E1 中继上传送的，因而它组网更加灵活，支持更大的话务量。我国的电话网络中有专门的七号信令网。

但支持七号信令的每个通信设备都需要有一个全局唯一信令点编码，而信令点编码资源是比较有限的。因而，七号信令主要在运营商的设备上使用，而运行商与用户设备（如 PBX）一般使用 PRI 信令对接。

5.8.4 七号信令

七号信令（SS7，Signaling System No. 7）是目前我国使用的主要的信令方式，用于局间通信。

我们来看一次简单的固定电话的通话流程。如图 1-8。用户 a 摘机，与其相连的 A 交换机根据电压、电流的变化检测到 A 摘机后，即送拨号音，同时启动收号程序。a 开始拨号，待 A 交换机号码收齐后，即查找路由，发送 IAM（Initial Address Message，初始地址消息）给 B 交换机。B 向 A 发 ACM（Address Complete Message，地址全消息）并通知话机 b 振铃，A 向 a 送回铃音。这时如果 b 接听电话，则 B 向 A 发送 ANC（Answer Charge，应答计费消息），a 与 b 开始通话，同时 A 对 a 计费⁵。

⁵B 也可以对 A 计费，称为中继计费，主要用于局间结算（话费）。如果 A 和 B 分属于不同的运营商（如移动和电信），则称为网间结算。

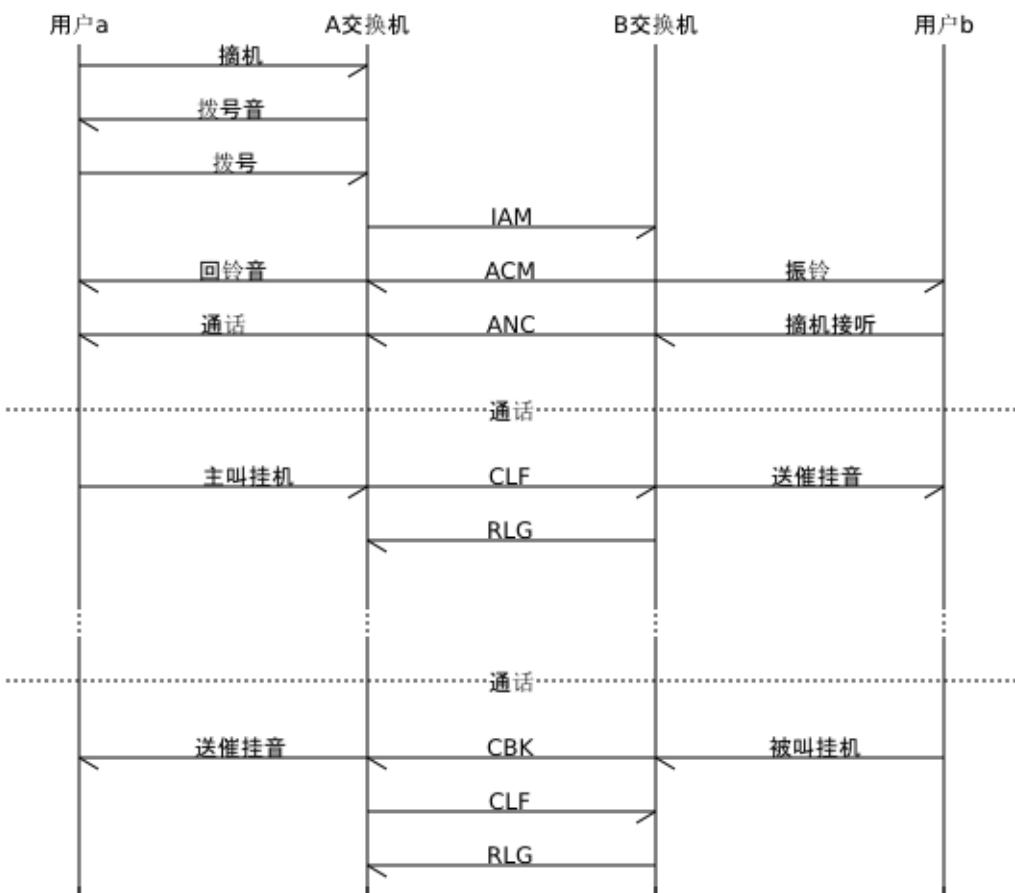


图 5.8: 七号信令局间呼叫流程

通话完毕，如果主叫挂机，则本端交换机 A 向对端 B 发送 CLF (Clear Forward, 前向释放消息)，B 向 A 回 RLG (Release Gard, 释放监护消息)，并向 b 送催挂音 (嘟嘟嘟…)。

如果被叫挂机，则 B 向 A 发送 CBK (Clear Backword, 后向释放消息)，A 回送 CLF，最后 B 回 RLG。

上面在交换机 A 与 B 之间传递的为七号信令中的 TUP (Telephone User Part, 电话用户部分) 部分。目前，由于 ISUP (ISDN User Part, ISDN 用户部分) 能与 ISDN 互联并提供比 TUP 更多的能力和服务，已基本取代 TUP 而成为我国七号信令网上主要的信令方式。ISUP 信令与 TUP 互通时的对应关系如图 1-9 所示：

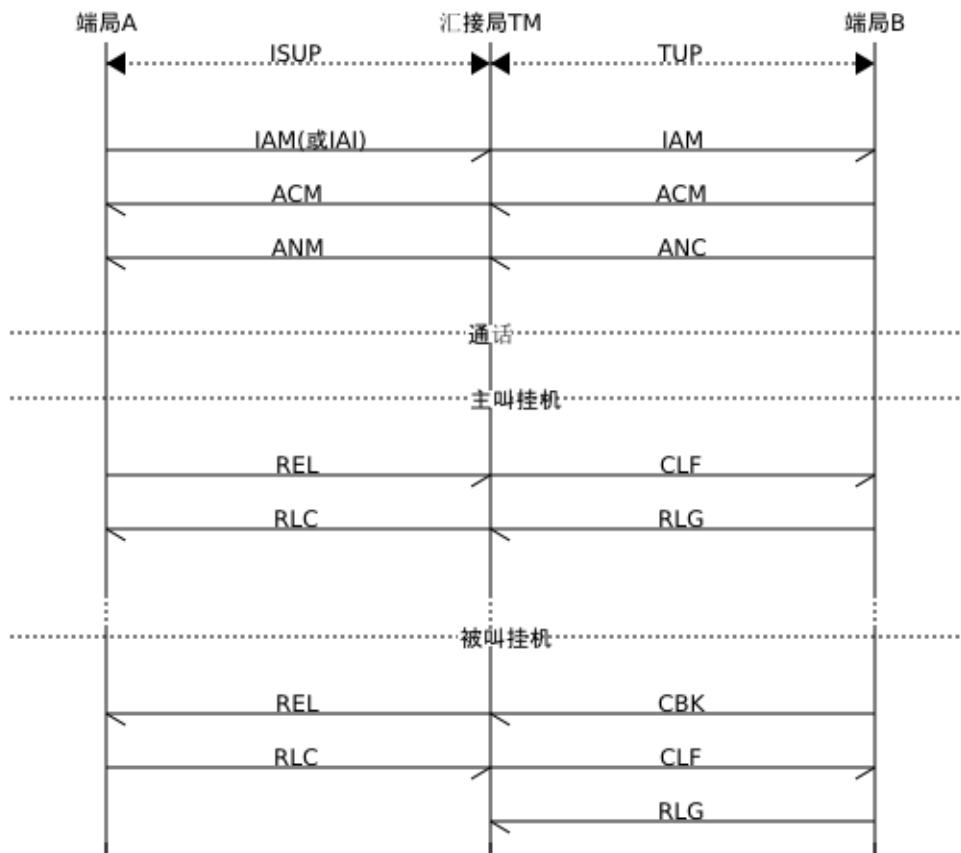


图 5.9: ISUP 与 TUP 互通信令流程

以上为端局 A 与端局 B 经过汇接局 TM 汇接通信中 ISUP 与 TUP 信令转换的例子。ISUP 信令的初始地址消息有 IAM 和 IAI (IAM With Additional Information, 带附加信息的 IAM) 两种，后者能提供更多的信息 (如主叫号码等)。另外 ISUP 信令的拆线信号不分前后向，只有 REL (Release, 释放) 和 RLC (Release Complete, 释放完成)。

在上一节中我们提到了 ISDN 信令，完整性起见，这里我们也来看一下 ISUP 与 ISDN 信令互通的例子。ISDN 使用 SETUP/CONNECT/RELEASE 消息分别对应 ISUP 里面的 IAM/ANM/REL 消息。如图 1-10：

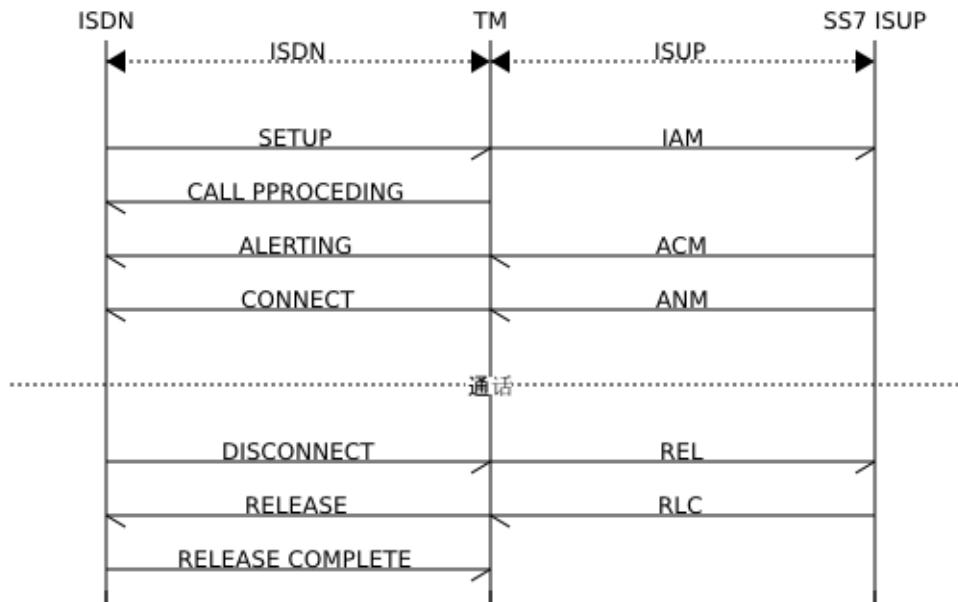


图 5.10: ISUP 与 ISDN 互通的信令流程

5.8.5 H323 与 SIP 信令

H.323 是 ITU 多媒体通信系列标准 H.32x 的一部份，该系列标准使得在现有通信网络上进行视频会议成为可能，其中，H.320 是在 N-ISDN 上进行多媒体通信的标准；H.321 是在 B-ISDN 上进行多媒体通信的标准；H.322 是在有服务质量保证的 LAN 上进行多媒体通信的标准；H.324 是在 GSTN 和无线网络上进行多媒体通信的标准。H.323 为现有的分组网络 PBN（如 IP 网络）提供多媒体通信标准。若和其他的 IP 技术如 IETF 的资源预留协议 RSVP 相结合，就可以实现 IP 网络的多媒体通信。

SIP (Session Initiation Protocol, 会话发起协议) 是由 IETF (Interne 工程任务组) 提出的 IP 电话信令协议。正如其名字所隐含的，SIP 用于发起会话，它能控制多个参与者参加的多媒體会话的建立和终结，并能动态调整和修改会话属性，如会话带宽要求、传输的媒体类型（语音、视频和数据等）、媒体的编解码格式、对组播和单播的支持等。

H.323 和 SIP 设计之初都是作为多媒体通信的应用层控制（信令）协议，目前一般用于 IP 电话。

它们能实现的信令功能基本相同，也都利用 RTP 作为媒体传输的协议。但两者的设计风格截然不同，这是由于其推出的两大阵营（电信领域与 Internet 领域）都想沿袭自己的传统。H.323 是由国际电信联盟提出来的，它企图把 IP 电话当作是众所周知的传统电话，只是传输方式由电路交换变成了分组交换，就如同模拟传输变成数字传输、同轴电缆传输变成了光纤传输。而 SIP 侧重于将 IP 电话作为 Internet 上的一个应用，较其他应用（如 FTP, E-mail 等）增加了信令和 QoS 的要求。H.323 推出较早，协议发展得比较成熟；由于其采用的是传统的实现电话信令的模式，便于与现有的电话网互通，但相对复杂得多。SIP 借鉴了其他 Internet 标准和协议的设计思想，有其突出的优点。

首先，它是基于文本的协议，而 H.323 采用基于 ASN.1 和压缩编码规则的二进制方法表示其消息，因此，SIP 对以文本形式表示的消息的词法和语法分析就比较简单。其次，SIP 会话请求过程和

媒体协商过程等是一起进行的，因此呼叫建立时间短，而在 H.323 中呼叫建立过程和进行媒体参数等协商的信令控制过程是分开进行的。再次，H.323 为实现补充业务定义了专门的协议，如 H.450.1、H.450.2 和 H.450.3 等，而 SIP 只要充分利用已定义的头域，必要时对头域进行简单扩展就能很方便地支持补充业务或智能业务。最后，H.323 进行集中、层次式控制。尽管集中控制便于管理（如便于计费和带宽管理等），但是当用于控制大型会议电话时，H.323 中执行会议控制功能的多点控制单元很可能成为瓶颈。而 SIP 类似于其他的 Internet 协议，设计上就为分布式的呼叫模型服务的，具有分布式的组播功能。

关于 SIP 信令我们在第 8 章详细讲解，读者可以先在这里参考一下以下 ISUP 与 SIP 间的信令转换关系图建立一个直观的印象（图 5-11）：

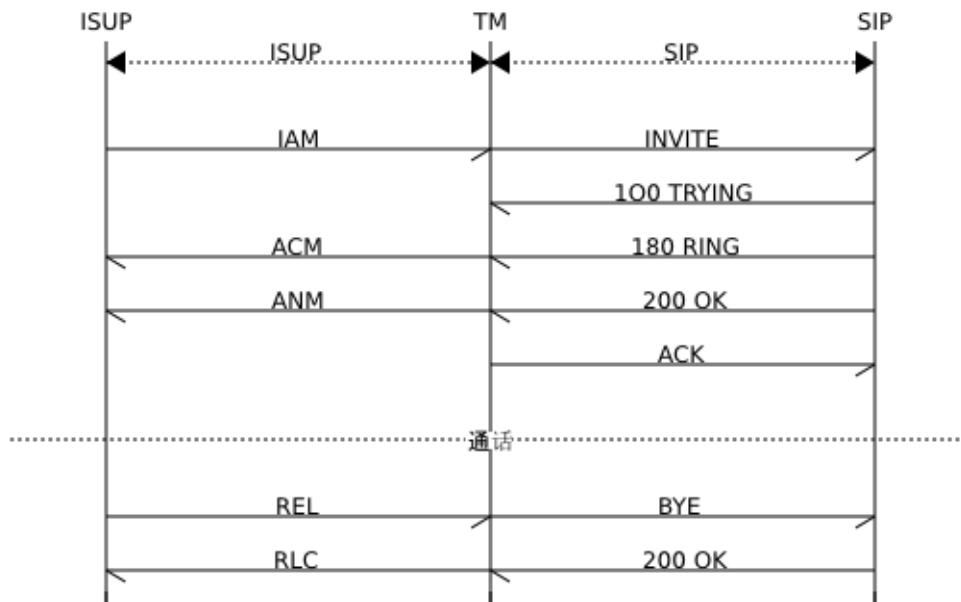


图 5.11: ISUP 与 SIP 互通的信令流程

5.9 媒体 (Media)

信令主要传输一些控制信号，而通信双方需要听到的是对方的语音数据，这些语音数据就称为媒体。随着通信系统能力的提高及更加高级、智能的终端设备出现，通信系统所能传输的媒体类型也越来越丰富，比较典型的有我们刚才提到的语音，还有视频、文字信息（短消息）、传真等。

在 SIP 通信中，除文字外，媒体是在 RTP 协议中传输的。由于媒体一般都是持续的传输，因此又称为“RTP 流”。

举一个通俗一点例子：假设张三乘火车从北京到南京，那么张三就是媒体。而火车称为承载（或载体，Carrier）。为了保证火车的正常运行，火车需要听从控制信号的指挥，而这个控制信号就相当于“信令”。如果信令出了问题，媒体就可能无法正常到达了。当然这个比方可能不是十分恰当，你也可以认为张三和火车都是媒体，只不过张三被火车“包”了一层，而铁路才是真正的载体。不管怎

么，在 SIP 通信中，SIP 相当于这里的火车控制信号，RTP 流中的语音数据相当于张三，它被 RTP 又“包”了一层，通过以太网承载到达对方。

5.10 电路交换与分组交换

5.10.1 电路交换

传统的电话都是基于电路交换。由于电路交换在通信之前要在通信双方之间建立一条被双方独占的物理通路（由通信双方之间的交换设备和链路逐段连接而成），因而有以下优缺点。

优点：

- 由于通信线路为通信双方用户专用，数据直达，所以传输数据的时延非常小。
- 通信双方之间的物理通路一旦建立，双方可以随时通信，实时性强。
- 双方通信时按发送顺序传送数据，不存在失序问题。
- 电路交换既适用于传输模拟信号，也适用于传输数字信号。
- 进电路交换的设备（交换机等）及控制均较简单。

缺点：

- 电路交换的平均连接建立时间对计算机通信来说较长。
- 电路交换连接建立后，物理通路被通信双方独占，即使通信线路空闲，也不能供其他用户使用，因而信道利用率低。
- 电路交换时，数据直达，不同类型、不同规格、不同速率的终端很难相互进行通信，也难以在通信过程中进行差错控制。

5.10.2 分组交换

我们熟悉的 IP 交换就采用分组交换的方式。它仍采用存储转发传输方式，但将一个长报文先分割为若干个较短的分组，然后把这些分组（携带源、目的地址和编号信息）逐个地发送出去，因此分组交换除了具有报文的优点外，与报文交换相比有以下优缺点：

优点：

- 加速了数据在网络中的传输。因为分组是逐个传输，可以使后一个分组的存储操作与前一个分组的转发操作并行，这种流水线式传输方式减少了报文的传输时间。此外，传输一个分组所需的缓冲区比传输一份报文所需的缓冲区小得多，这样因缓冲区不足而等待发送的机率及等待的时间也必然少得多。
- 简化了存储管理。因为分组的长度固定，相应的缓冲区的大小也固定，在交换结点中存储器的管理通常被简化为对缓冲区的管理，相对比较容易。

- 减少了出错机率和重发数据量。因为分组较短，其出错机率必然减少，每次重发的数据量也就大大减少，这样不仅提高了可靠性，也减少了传输时延。
- 由于分组短小，更适用于采用优先级策略，便于及时传送一些紧急数据，因此对于计算机之间的突发式的数据通信，分组交换显然更为合适些。

缺点：

- 尽管分组交换比报文交换的传输时延少，但仍存在存储转发时延，而且其结点交换机必须具有更强的处理能力。
- 分组交换与报文交换一样，每个分组都要加上源、目的地址和分组编号等信息，使传送的信息量大约增大 5% ~ 10%，一定程度上降低了通信效率，增加了处理的时间，使控制复杂，时延增加。
- 当分组交换采用数据报服务时，可能出现失序、丢失或重复分组，分组到达目的结点时，要对分组按编号进行排序等工作，增加了麻烦。若采用虚电路服务，虽无失序问题，但有呼叫建立、数据传输和虚电路释放三个过程。

总之，若要传送的数据量很大，且其传送时间远大于呼叫时间，则采用电路交换较为合适；当端到端的通路有很多段的链路组成时，采用分组交换传送数据较为合适。从提高整个网络的信道利用率上看，报文交换和分组交换优于电路交换，其中分组交换比报文交换的时延小，尤其适合于计算机之间的突发式的数据通信。

5.11 VoIP

维基百科上是这样说的：

IP 电话（简称 VoIP，源自英语 Voice over Internet Protocol；又名宽带电话或网络电话）是一种透过互联网或其他使用 IP 技术的网络，来实现新型的电话通讯。过去 IP 电话主要应用在大型公司的内联网内，技术人员可以复用同一个网络提供数据及语音服务，除了简化管理，更可提高生产力。随着互联网日渐普及，以及跨境通讯数量大幅飙升，IP 电话亦被应用在长途电话业务上。由于世界各主要大城市的通信公司竞争日剧，以及各国电信相关法令松绑，IP 电话也开始应用于固网通信，其低通话成本、低建设成本、易扩充性及日渐优良化的通话质量等主要特点，被目前国际电信企业看成是传统电信业务的有力竞争者。详细内容参见维基百科上的[IP 电话⁶](#)。

目前，VoIP 呼叫控制协议主要有 SIP、H323，以及 MGCP 与 H.248/MEGACO 等。H323 是由 ITU-T（国际电信联盟）定义的多媒体信息如何在分组交换网络上承载的建议书。它是一个相当复杂的协议，使用起来很不灵活。而 SIP 则是 IETF（互联网工程任务组）开发的（RFC3261），它是一种类似 HTTP 的基于文本的协议，很容易实现和扩展，被普遍认为是 VoIP 信令的未来。

⁶http://zh.wikipedia.org/wiki/IP_电话

5.12 IMS

IMS 系统涉及到的概念和名词术语相当多，在本节，我们简单加以介绍，对此感兴趣的读者可以进行参考，也可以根据这里提到的关键词到网上搜索或查找相关书籍进行更深入的学习；其他读者跳过本节。

IMS 的全称是 IP 多媒体子系统 (IP Multimedia Subsystem) 它是一个基于 IP 网提供语音及多媒体业务的网络体系架构。它最初是由 3G 标准化组织 3GPP⁷设计的，作为其 GSM 之后的未来移动网络远景目标的一部分。IMS 的最初的版本 (3GPP R5) 主要是给出了一种基于 GPRS 来实现 IP 多媒体业务的方法。在这个版本的基础上，3GPP、3GPP2 以及 TISPAN 进行了进一步的更新，以支持 GPRS 之外的，诸如 WLAN、CDMA2000 和固定等其他接入网络。从目前来看，IMS 是独立于接入网技术的，虽然 IMS 与底层传输功能有着很多联系。

从另外一个角度看，IMS 实际上是 IP 网上的一个应用系统。关于 IP 网的相关技术标准主要由 IETF 制定，包括应用层如 Email(POP3、SMTP)、文件传输 (FTP)、网页浏览 (HTTP) 等的相关协议标准。IETF 负责制定了实时应用 (Real-time Applications) 相关的协议标准，包括 SIP、RTP 等。IMS 基本使用了 IETF 的相关协议标准(SIP、Diameter 等)，同时在其基础上有着详细的描述和增强，以便于提供一种完整的、健壮的多媒体系统。这些增强和操作性描述为运营商控制、分责任、计费和安全提供了支持。

支持 IP 多媒体的全套解决方案是由终端、GERAN (GSM EDGE Radio Access Network, GSM/EDGE 无线通迅网络) 或 UTRAN(UMTS Terrestrial Radio Access Network - UMTS 陆地无线接入网)、GPRS 核心网和 IP 多媒体核心网子系统的一些特殊的功能单元。这些功能单元包括呼叫会话控制功能 (CSCF)、媒体网关控制功能 (MGCF)、IP 多媒体网关功能 (IM-MGW)、多媒体资源功能控制器 (MRFC)、多媒体资源功能处理器 (MRFP)、签约定位功能 (SLF)、出口网关控制功能 (BGCF)、应用服务器 (AS)、信令网关功能 (SGW) 等。

IMS 网元众多，其核心网络基本架构如图：

5.12.1 IMS 的特点

- 采用 SIP 作为呼叫控制协议
- 支持 Diameter 协议
- 采用归属控制方式
- 采用接入无关性
- 业务、控制、承载层完全分离

⁷第三代合作伙伴计划 (3rd Generation Partnership Project) 是一个成立于 1998 年 12 月的标准化机构。目前其成员包括欧洲的 ETSI、日本的 ARIB 和 TTC、中国的 CCSA、韩国的 TTA 和北美洲的 ATIS。详见：<https://zh.wikipedia.org/wiki/3GPP>。

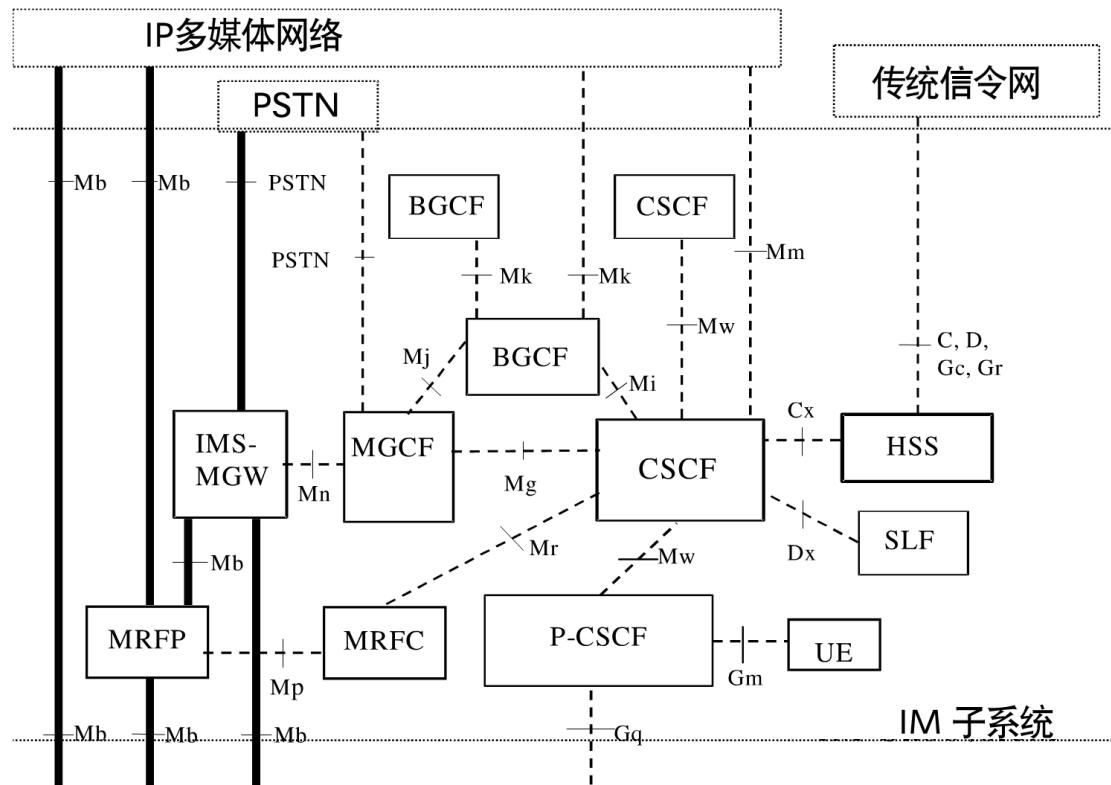
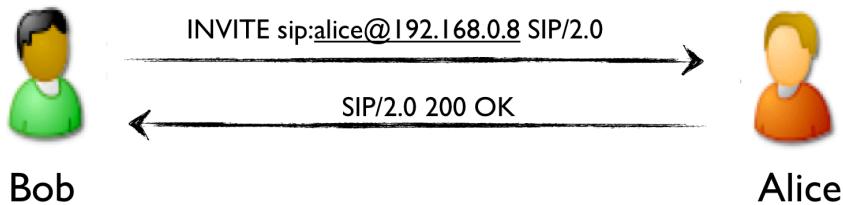


图 5.12: IMS 基本架构

- 增强计费功能：支持在线、离线计费
- 增强多媒体业务
- Presence：呈现
- Messaging：短消息
- Conferencing：会议
- PoC：Push-to-talk over Cellular，基于移动网络、采用 VoIP 技术的集群对讲业务
- MBMS：Multimedia Broadcast Multicast Service，多媒体广播多播服务

5.13 生活中的信令和媒体

杜金房/2016.06.04



什么是信令什么是媒体呢？有人问。

这还不简单，咔咔咔，我能讲一小时。但对于非技术人员来讲，我应该怎么解释呢？讲个故事吧。

由于我这几年比较专注做 FreeSWITCH，因此，有人邀请我去北京讲课。我的家在烟台，所以要坐飞机从烟台去北京。我平常比较忙，以下是对话内容发生在客户那边的负责人和我的秘书之间：

- 您好，不知道杜老师什么时候有时间来给我们讲讲 FreeSWITCH？
- 猴年马月（实际上快到了，听说是 2016 年 6 月 5 日（芒种）至 7 月 3 日（小暑前夕），就是传说中的“猴年马月”！）。
- 他坐飞机还是高铁？我好安排买票。
- 飞机。
- 好的，稍等。… 杜老师的机票已经买好了，航班号 XXXX，首都机场 3 号航站楼，到时候有人接。
- 好的，谢谢，我告诉杜老师。

上面的对话大家应该都很容易能理解吧。其实在上面，除了媒体和信令外，还有两个其它的概念。我们先不说，先卖个关子。

下面我们来看一个 SIP 通话。

Bob	Alice
INVITE Alice@example.com	
----->	
100 Trying	
<-----	
180 Ringing	
<-----	
200 OK	
<-----	
ACK	
----->	
<=====RTP=====	
BYE	
<-----	
200 OK	
----->	

在上面的 SIP 通话中。Bob 给 Alice 打电话。

首先，Bob 拿起话机拨号，Bob 的话机发送 INVITE 消息给 Alice，说 Alice，咱俩说说话…

Alice 的话机收到后，跟 Bob 的话机说，稍等… 100 Trying

Alice 的话机这时振铃，叮铃铃… 同时，它给 Bob 的话机回一个消息 180 Ringing

Bob 的话机收到 180 后，开始给 Bob 播放回铃音嘟嘟嘟… 告诉 Bob，再等等，Alice 的话机已经振铃了

Alice 听到后，接起电话，这里 Alice 的话机给 Bob 的话机发一个消息，200 OK。表示，Alice 已经接听了

Bob 的话机收到 200 OK 消息后，停止播放回铃音，这时候，Alice 和 Bob 就可以通话了。

Alice 和 Bob 通话的内容是通过电话线传输的，通话的内容叫媒体（Media），而电话线其实是一个传输媒介或载体，叫 Transport 或 Carrier。

好吧，现在我们知道什么是媒体了。那么，上面罗嗦了一大堆就叫做信令。因此，如果没有信令，Alice 和 Bob 就无法通话，可见，信令是为了通话建立服务的。它主要是通过一系列的消息，完成一个通话的建立。而这一系列的消息，就叫做信令。

当然，挂机后也要有信令（BYE）。

信令的传输是在两个话机之间发生的。这两个话机，分别代表 Alice 和 Bob，叫做 UA (User Agent)，也就是叫用户代理。

那么，这些跟我们上面讲的杜老师讲课的故事有什么关系呢？

关系大着呢。我们对比一下。

我们把杜老师当作媒体。因为对方是想把杜老师请过去。而 SIP 通话是把话音传过去，所以话音是媒体，杜老师是媒体。

那么，之前邀请杜老师的那些聊天消息就是信令。这些信令都是为了把杜老师请过去这个目标服务的。

传输媒体的媒介或载体是什么呢？当然，是飞机。

UA 是谁？就是客户那边的负责人和我的秘书。

一切都是来源于生活。

生活如此美好，我们再深入研究下 SIP 信令。下面是一个典型的 INVITE 消息：

```
INVITE sip:9196@192.168.7.6 SIP/2.0
Via: SIP/2.0/UDP 192.168.7.6:48808;branch=z9hG4bK-d8754z-eb76c76409fc5100-1---d8754z-;rport
Max-Forwards: 70
Contact: <sip:1000@192.168.7.6:48808>
To: <sip:9196@192.168.7.6>
From: "1000"<sip:1000@192.168.7.6>;tag=bfdeeb5b
Call-ID: ZDQ5ZTYyMDYyNmUxOTlhYzNjNTg3NTEyOGQ4MzJiZjc
CSeq: 1 INVITE
Allow: INVITE, ACK, CANCEL, OPTIONS, BYE, REFER, NOTIFY, MESSAGE, SUBSCRIBE, INFO
Content-Type: application/sdp
Supported: replaces
User-Agent: Bria 3 release 3.5.5 stamp 71243
Content-Length: 208

v=0
o=- 1465047960147349 1 IN IP4 192.168.7.6
s=Bria 3 release 3.5.5 stamp 71243
c=IN IP4 192.168.7.6
t=0 0
m=audio 59108 RTP/AVP 0 8 101
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=sendrecv
```

其中，第一行 INVITE 表示请杜老来讲课。9196相当于杜老师的电话号码，@后面是 IP 地址，相当于我家或办公室的地址吧。

Via: SIP/2.0/UDP 表示信令消息是用什么承载的，除此之外还有 TCP, WS (Websocket) 等。可以理解为北京负责人跟我的秘书是用微信聊天还是 QQ 或电话沟通的。

From一行，是说明这条消息的来源，可以理解成北京那边的号码是 1000。主叫号码。

To还是 9196，杜老师的号码，被叫号码。User-Agent 是说明北京那边用户代理的型号，比方男秘书还是女秘书，女性的名字等。

Content-Type更进一步说明消息的内容，可以理解成行程安排。

从 v=0 开始，是真正的行程安排。

c=IN IP4 192.168.7.6 这一行，是说明媒体要送达的 IP 地址，可以理解成北京首都机场。IP4 可以理解成做飞机，如果是 IP6 就代表坐火车。

m=audio 59108 RTP/AVP 0 8 101 中的 59108 代表媒体要送达的端口号，可以理解成第三航站楼的某个出口，接机的人就在那里等着。

RTP/AVP 代表什么呢？头等舱或经济舱吧。

后面的 0 8 101 其实约定媒体编码的类型，也是一个暗号，比方告诉说接机的人和杜老师都要拿一把雨伞（在这个特殊的日子里我忽然想到了这个词，请不要多想）方便互相认识。

好吧，一个 SIP 信令，就能把杜老师骗到北京去了。

SIP 的全称是 Session Initiation Protocol，即会话初始协议。以后，邀请我讲课请用 SIP :)。

第六章 小樱桃

小樱桃，我们要做最美的那一颗。

6.1 小樱桃是怎么来的？

杜金房/2015.12.17

本故事纯属虚构，如有雷同，投个票吧？

作为一个创业者，除了要搞定资金、技术、发展方向等问题外，一个不可忽略的环节就是就公司起名？起名也是有学问的，要符合公司主营业务、要三观正、要好听、要有逼格、有品味、要雅俗共赏、还要吉祥……

不管怎么样，绞尽脑汁起几个心仪的名字，还不算是太难的。再找风水先生一算，选一个顺风顺水的，便可以去注册了。但你还得过最后一关——要能通过核名，能注册上。

有重名肯定不行，跟别人家的公司名称太相似也不行，总之还有很多不行，最后，把想好的名字颠三倒四排列组合就出现了许许多奇怪的名字。

我们也不脱俗，做通信的，最后选了个名字叫信悦通，全称是北京信悦通科技有限公司。当初没觉得怎么样，后来用得多了就觉得这个名字还挺顺眼的。注册完公司还得申请域名，全拼音的域名已被别人注册了。最后我们选了个奇葩的x-y-t.com。事情就这么定了。

现在，拟在烟台成立一家公司，实在想不出有比信悦通更适合的名字。如果用烟台分公司注册的话，由于分公司不能有独立的法人，不在考虑之列。所以我们就很自然地想用烟台信悦通科技有限公司的名字。网上查了也没有重名，但核名还是不过，理由是前两个字跟其它公司有重音。重音又不重字，而且我们是三个字的名称，但无论我们怎么说都不让用这个名，实在是郁闷。

此外，烟台还规定科技前面必须有行业名称，比方是网络科技，所以，即使这个名字能注册也得是烟台信悦通网络科技有限公司。每次开发票时都需要多打两个字，累积下来浪费的时间也是挺要命的。

这个名字看来是没指望了，为了能抓紧注册，我们还得赶快想名字。工商那边建议我们把字的顺序颠倒一下，比如悦信通，但我们还要考虑兼容我们的公司域名啊。好在我们的域名是x-y-t.com，所以现在的基本想法是能套上这个域名就行，比如信远通之类的。

当然，一个名字不够用，我们又开动脑筋想了想，符合条件的还真不算少，有的还很有趣。至于哪个最好，我们也不确定，大家投个票支持下吧？

信远通 信云通 信要通 信又通 信义通 信也通
信已通 新语通 行远通 星云通 星云亭 行也通
讯悦通 小樱桃 小丫头

最后的名字将是『烟台 XYT 网络科技有限公司』格式。如果你能想的更好的，留言，一经采用有红包奖励啊:D。

更新：小樱桃就是这么来的。

6.2 如何获得 FreeSWITCH 培训优惠？

写于 2016 年 8 月

FreeSWITCH 培训班月底就要在北京跟大家见面了。这次，我们又将迎来一批新的学员。

时间过得好快，不知不觉，我们的 FreeSWITCH 培训，几年的时间已经搞了十几届了。

有很多同学比较关注价格，问如何获得优惠，老板今天请假了，今天，小樱桃就在这里给大家透露点秘方。

- 早报名会有优惠。杜老师以前是卖飞机票的，因此，培训的定价也跟机票类似，你买得越早，就越便宜。
- 团购有优惠。多来几个人一般可以申请更多的折扣，我们有一次培训班一个单位一下来了 9 个人，后来又追加两个…
- 老客户有优惠。杜老师老是觉得如果有重修的学生，就是老师教的不好，所以，优惠。我们真有几个学员上过三、四次课的，想想杜老师教的是有多不好啊。
- 搞好关系有优惠。经常给杜老师发个红包，打个赞赏，请杜老师吃饭什么的，到时候说不定也能优惠。(别听我说的啊)
- 走后门有优惠。杜老师有很多要好的朋友，如果你能找他们说上两句，说不定也管用。
- 为人民服务有优惠。杜老师花了很大精力建设和维护 FreeSWITCH 社区，如果谁在社区里比较积极，杜老师说不定也能看在眼里。
- 有过业务往来的有优惠。嗯，比如买过我们的技术支持服务的，打过广告的。
- 运气好，免费票也有。虽然很少，但历史上确实有过免费上课的学员。想必都是对杜老师帮助过很大的人，或者很厉害的后门。
- FreeSWITCH A 计划成员有优惠。
- 小樱桃公司的员工培训免费，还包差旅住宿旅游（可以申请加入我们 o）。

不知不觉写了十条。哪一条适合你呢?

如果没有适合的也不要紧，小樱桃并不是标题党。其实——

- FreeSWITCH 培训，是一个很好的线下交流机会。平常，哪能见到那么多搞 FreeSWITCH 的人呢？因此，培训更重要的意义不是跟杜老师学，而是跟武林同道切磋交流。
- FreeSWITCH 培训，是一个很好的合作机会。杜老师不会告诉大家其实他有一大部分业务来自曾经参加过培训的学员或单位吧？
- FreeSWITCH 培训，是一个很好的休息机会。我们的学员大部分都是公司的技术骨干，平常在公司里忙着救火，难得闲下来好好学学技术不是？
- FreeSWITCH 培训，是一个很好的展示自我的机会。学员也可以上台分享自己的经验、知识。
- FreeSWITCH 培训，是一个很好的展示风范的机会。如果你是一个领导，给公司的员工一个培训机会，员工会更爱戴你，会更加卖力的工作，增加的工作效率其实跟培训费比起来那简直是……，嗯，你懂得！
- FreeSWITCH 培训，是一个很好的“养兵”的机会。有很多公司平时不注意投资学习，用到的时候又发愁没有现成的 FreeSWITCH 技术人员可以用，如果能提前做好培训，不是比临时抱佛脚来的更有效率吗？

6.3 烟台小樱桃成功参加 2016 年南京企业通信与呼叫中心技术发展论坛

小樱桃/2016.04.11

“江南佳丽地，金陵帝王州”，六朝古都南京有着 6000 多年的文明史，拥有厚重的文化底蕴和丰富的历史遗存。一下飞机，清新的空气扑面而来，雨后的路面干净湿润，两边的树木郁郁葱葱，白墙青瓦的房子错落有致，小桥流水人家的江南美景映入眼帘，看到这样的场景我们也醉了……

错了错了，这不是重点，我们的重点是来参加“2016 年南京企业通信与呼叫中心技术发展论坛”的，上面那段是情不自禁，有感而发的。还是快让我们言归正转吧！

俗话说“台上三分钟，台下十年功”，这个论坛从策划到成功举办历时半年，在通信业前辈刘光大老师的全程指导和督办下，主办方江苏智恒从组织、开发、宣传到成功举办，每个环节都是颇费心思。先来看看亮点吧！首先就是高大上的会场，超大超酷的屏幕最让杜老师满意了；其次，本次论坛支持微信全程直播（神州大唐自主研发），我们也都及时把链接发到各个群上，让那些想来而没有来的小伙伴们一起分享。当然，有些人反应效果不好，可能是网络问题也可能是手机问题，还可能是人品问题：），不过没关系，查看我们的公众号 FreeSWITCH 中文社区还有重播，没过瘾的朋友再来看一次：）；再次，整个演讲过程中有三次抽奖活动，每个人扫码签到时会有一个编号，抽奖采用电脑随机抽取；最后，本次论坛是集技术型、企业型、学者型于一体，开发者、企业家、学者各有各的菜。

第一个演讲的人就是我们的杜老师，因为当天安装的抽奖系统，把我们原来调试好的课件比例给打乱了，导致现场演讲时界面不能全部展示，但这丝毫不影响他的发挥。有人说他更帅了，也有人说

他的头更亮了，呵呵，这都是朋友们的关爱。对了，不得不提的是他的幽默，他的幽默方式大部分我们还是可以接受的，但也有尴尬的时候。比如今天，当讲到烟台的四季时，他搞了张在海边穿比基尼的美女的图片，按照国外的习惯来说，一看到这张图大家会嘘声一片，本来是想调节一下会场气氛的，但结果却鸦雀无声，“好尴尬啊”！哈哈！

当然，后面神州大唐孙总的“FreeSWITCH 与 CTI 完美结合商业价值”、声网的实时云、易掌云峰的全媒体智能客服、云软的多渠道用户体验、基立讯的智能语音以及北大段博士的大数据在企业通讯中的应用等内容也都是很有前瞻性、实用性、变革性的干货，值得大家看看重播，好好消化一下（别说我没提醒你）。

让我们感到很开心的是，会议中间好多对 FreeSWITCH 感兴趣的人都来我们的展位上看我们的实时模拟演示，还有很多人是来找杜老师合影的，呵呵！会议中我们的《FreeSWITCH 权威指南》得到了好评和热卖，我们可提供现金、支付宝等多种支付方式噢！

整个会议是很成功的，这与大家的齐心协力是分不开的，首先就是刘光大老师（会前还在打吊瓶），前前后后组织多次讨论会，自费奔波往来各地；其次就是江苏智恒，前期的付出先不说，就会议前几天，每天都忙到很晚；另外，各家发起单位、赞助单位也都付出百分百的努力；最后，大家能从百忙之中抽出时间赶来参加这次论坛，也充分说明了大家对通信业及呼叫中心的一颗心。“老骥伏枥，志在千里”作为老一辈通信人，虽然都已退休，却一直为通信业及呼叫中心奔走操劳，默默耕耘，真是值得我们下一代好好学习！

在走完了玄武湖，品完了淮扬菜后，我们更爱南京了！



图 6.1: 玄武湖畔会议中

6.4 FreeSWITCH 走进校园之烟大行

小樱桃/2016.03.21

“沾衣欲湿杏花雨，吹面不寒杨柳风”，春天到了，如此的沁人心脾，为不辜负这大好的时光，应烟台大学互联网协会的邀请，开启了我们 2016 年的校园大讲堂，也让 FreeSWITCH 第一次走进了校园。

先来介绍一下，烟台大学是是国内距海最近、拥有海岸线最长的滨海大学，是山东省属重点综合性大学，东临黄海，西依青山，风景秀丽，气候宜人，最重要的一点是她是杜总的母校，这里有他的青春，有他的足迹，也有他对母校的感情。我们先来几张烟大的美景图养养眼吧！



言归正转，这次讲座的主题是“从业者眼中的专业、职业、创业”，主要面向大三和大四的学生，因为他们面临着就业的问题，也容易产生共鸣，但机缘巧合的是大一、大二的同学来的比较多。杜总先用几个技术词语作为引子，又分别从“职业、专业和创业”三个方面，结合自身的经历进行了详细的阐述。讲到专业方面，杜总讲了几个经典的案例，进行深入浅出的说明。当说到哪种语言好时，杜总引用了网上流行的“学不同计算机语言的人相互鄙视”的段子，引得满堂大笑。演讲的整个过程笑声不断。



杜总用他特有的幽默方式和风趣语言让我们明白了以下几点：一兴趣是最好的老师；二自己认准的事要舍得投资；三不懂没关系，要有快速学习的能力；四要主动一些，多学多做多尝试。杜总通过演讲，希望能给这些活在象牙塔里的莘莘学子一些启示，在以后的学生和职业生涯中少走弯路。

此次讲座是免费的公益性讲座，饱含了杜总对母校的那份感情。学生的热情和兴趣度超出了我们的想象，我们也被这青春活力所感染。学生的可塑性很强，一个不经意的引导，也许会给他们开启一扇窗。今天我们走进了烟大，明天我们还会走进更多的学校，让越来越多的学生了解 FreeSWITCH、知道 FreeSWITCH、使用 FreeSWITCH。

光阴似箭，不知不觉，又是一个轮回。漫步在校园中，抬头望，只见太阳透过朦胧淡薄的云层放射着光芒，把温暖和光辉一缕缕地洒满了校园、、、、、

精彩瞬间



6.5 FreeSWITCH 高手速成培训 2016 春季班（南京站）圆满结束

小樱桃/2015.04.12

“烟花三月下江南”，我们顺着春天的脚步来到了南京——这座古老而有着深厚文化底蕴的城市。三月的江南处处是美景，这好山好水好风光为我们拉开了 2016 年 FreeSWITCH 培训的序幕。

也许是城市的召唤，也许是季节的美好，把全国各地的学员聚集在了一起，他们有的来自黑龙江、辽宁，有的来自贵州、四川，有的来自广东，有的来自杭州、苏州、上海，当然还有南京本地的大部队，真的是五湖四海。其中西南地区的学员占比达到 31%，这让我们看到，除了北上广外，FreeSWITCH 的使用者越来越向全国普及，这真是一个好趋势。

四月的南京也是一个会议较多的地方，我们 7 号就到了南京，因为我们是 8 号“企业通信与呼叫中心技术发展论坛”的发起单位之一，也有不少学员提前一天到，参加了 8 号的论坛。关于 8 号的论坛，我们也有专门的汇报，此处不再赘述。

本次培训日程还是三天，三天的课程充实而愉快，大家白天上课，晚上还可以欣赏一下南京的风光，真是出差旅游两不误，呵呵！当然大家都知道，杜老师是非常负责和敬业的老师，他的课要百分百的投入，因为他讲的内容都是干货，信息量非常大，速度也非常快，稍不留神，你就 OUT 了。所以，一天的课程下来，小伙伴们还是很辛苦的。另外，小伙伴们都是“有备而来”的，课下也没让杜老师休息，一直讨论到上课，还有的同学午饭都没吃，和我们的技术人员一直做测试到下午上课，为了充分和我们的技术人员交流，这也是拼了。

按照惯例，在培训的第二天晚上，我们要一起聚餐，因为人数较多，我们分成了三桌，大家也真是太敬业了，都在讨论着技术问题。他们不止来自各地，也来自各行各业，所以，谈着谈着大家竟成了合作伙伴。到了南京不吃江鲜可就太可惜了，经典的淮扬菜不仅造型好，口味也很独特，让小伙伴们赞不绝口。这次聚餐结束的很晚，到了最后还玩起了游戏，直到酒店要打烊了，才不得不被迫结束。

最后，我们照了毕业照，也颁发了结业证书！本次培训圆满结束！但是大多数小伙伴都邀请我们去当地组织下次的培训，也建议我们周三开始培训，这样小伙伴们周六周日就可以……你懂得，呵呵！这些我们都会考虑的，谢谢小伙伴们的支持，培训我们还会再组织的，下次等你来约！

欣赏一下我们的照片吧！



图 6.2: 我们在上课

6.6 FreeSWITCH 开发者沙龙 2016

时间它走得快啊。转眼，又到了新一届 FreeSWITCH 开发者沙龙的日子了。今年，是第五届。



图 6.3: 难忘的聚餐



图 6.4: 美味的淮扬菜



图 6.5: 我们毕业了



图 6.6: 刘光大致开场辞

回首过去，2012/2013 年，在车库咖啡；2014 年，北邮科技大厦；2015，鹏润国际酒店。在过去的几年中，我们感觉到了成长，也感受到了大家的认可。不管是还未走出象牙塔的学生、刚刚入行的年轻人、资深的开发者、CTO、企业者、企业的带头人，还是游走在技术边缘的销售和客服人员、观察者、分析师，都有一些人切实地感受到我们沙龙的价值，默默地，始终如一地支持着我们。

今年，我们做了一些改变。

首先，我们这一次选择了一个超五星级的酒店—北京泰富酒店。酒店环境好，高大上，交通方便。

其次，我们把免费的活动改成收费的。根据报名时间早晚收费金额从 128 到 512 元不等（含午餐）。

不同于其它的行业会议，我们的活动是一场技术沙龙，根本宗旨是『源于开发者、用于开发者』。我们推崇以技术为主线，做一些分享和交流，顺便做一些社交和广告。是的，我们不拒绝广告。但是，我们希望大家把广告巧妙地植入你的演讲、易拉宝和 PPT 中。让听众觉得你是在技术中植入了一些广告，而不是在广告中植入了一点点技术。

FreeSWITCH 是开源的，但我们也喜欢跟任何商业产品对接，共同把世界变得更美好。

我们的活动不是以盈利为目的的。我们收门票和赞助费，只是为了能选择一个更体面的酒店，把活动做得更好。

这次，将有神秘嘉宾到场，也给我们的活动增色不少。Giovanni Maruzzelli 是 FreeSWITCH `mod_skypopen` 和 `mod_gsmopen` 的作者，他从遥远的意大利赶来参加这盛会，分享了他的经验、技术、和心得。

FreeSWITCH 沙龙一直以干货居多而闻名，来参会的大多数都是技术开发人员和技术带头人，他们关注的不仅是技术、而且是技术和技术，所以，今年不仅仅是干货问题，还有连连不断的惊喜。

呼叫中心行业资深评论家刘光大老师做开幕词，他表示“从车库咖啡到五星级酒店，每年一届的 FreeSWITCH 开发者技术沙龙让我感慨的，不是场地越来越豪华，也不是人气越来越火，而是越来越多的人对 FreeSWITCH 这款国际流行的开源软交换平台的认知、认可、研究、应用和受益。另一方面，我看到越来越多的武林高手华山论剑甚是欣慰。”

杜金房老师分享了他多年的 FreeSWITCH 心得和经验，以及从刚刚结束的 ClueCon 上得到的最新消息和照片。好像他每次演讲都不忘了说“开源不等于免费”。



图 6.7: 杜金房

接下来的惊喜就是七岁的冰冰，上台演示用可视化编程配置 IVR 语音导航，让来宾们惊奇不已。看来 FreeSWITCH 真的要从娃娃抓起。

我们的老朋友贾强讲解了《编码那些事儿》；Sangoma 大中华区总经理朱利中带来了《强大的 FreePBX》；来自草原的赵总给我们分享了他基于 FreeSWITCH 的创业经验；美女徐厅庭，用她那温柔的声音让我们了解了“Equiinet”；天才老总马天才，先让我们了解了呼叫中心技术史，又跟我们分享了他是怎么用 FreeSWITCH 和 Erlang DIY 一个呼叫中心的；上午的最后一位是来自昆山箭斗云信息科技有限公司技术顾问张弘，他在分布式高承载通信网络方面有很深研究，主要分享了“IMS 构架和接入方法”，让大家受益匪浅。

下午的活动以抽奖拉开了序幕，主办方设计了微信抽奖，场面热烈，奖品除了赞助商赞助的 IP 话机还有杜老师的新书。接下来是云联 CTO 王化春带来的《会话边界控制器在 IP 音视频通信中的重要作用》以及声网战略合作总监杨剑分享的《移动互联：连接人与服务》。



图 6.8: 冰冰

来自意大利的大虾 Giovanni 分享了《FreeSWITCH High Availability and Scaling for SIP and Verto》，为了能做到台上台下有互动，他讲的速度较慢并伴有解释与描述，下面的技术人员英语水平好像也不差，听得都津津有味。

上海澍品 CTO 分享了“驱动中小企业信息化”的相关知识，最后一个演讲的是曾连任 9 年的微软全球最有价值专家，多届微软技术大会讲师—衣明志，他演讲的内容是“用 Xamarin 进行跨平台开发”，现场演示了用 Xamarin 如何打电话，非常生动。

最后一个环节是圆桌会议，请各位大牛上台，台下技术人员对自己感兴趣的讲师提问。问到 Giovanni 的时候，或许是台下观众嫌杜老师翻译地太慢，直接就用英文提问了。英语不好的同学一头雾水，最后杜老师还是进行了些简短的翻译。



图 6.9: 圆桌会议

为了本次沙龙，杜老师特意准备了新书《FreeSWITCH WIRESHARK》、《FreeSWITCH 实例

解析》和《FreeSWITCH 文集》，参会人员每人可扫码送一本《文集》。

通信界的前辈刘光大老师，从第一到第五届，他一路伴随而来，默默的支持着我们。他的身体有些欠佳，但一直在给我们拍照、录像、上传照片、时时播报，真让我们感动。

小伙伴们也很给力，大家都知道，今年的沙龙我们是收费的，事实证明这种方式效果不错。一方面，提高了参会人员的质量，交了钱很少有随便不来参加的；另一方面，也多多少少能给我们抵消掉一些成本，使我们有更多的力量把活动组织好。报名的小伙伴们都能如期而至，并自始自终没有离开，认真听讲，并与我们的讲师良好互动，这是非常难得的。

我们可爱的赞助商们，个个英俊潇洒，精气神足，他们也跟随着我们很多年，一直支持着我们的活动。无论是会前会后还是休息期间，他们都尽心尽力的与参会的小伙伴们沟通交流，有一些也切切实实“卖”出了产品。我们今年的赞助商级别是按视频分辨率定的。

1080p	360p			
720p				

图 6.10: 赞助商

当然，我们需要感谢的人还有很多，感谢崔钢先生的主持活动，使一切井井有条；感谢远道而来的朋友，不管是专门为了沙龙来得，还是专门为了杜老师来的，还是专门为了刘老师来的，还是专门为了哪个帅哥大姐来的，都感谢；感谢泰富酒店的工作人员，他们前前后后保证了活动的圆满；感谢摄像、录像师，他们把我们的瞬间变成了永远；也感谢我们自己，数月的努力换来大家这么多的微笑，嘿嘿！

好了，主要内容播报到这，没有来现场的你，错过这次只能再等一年了。关注我们，下一年等你噢！



图 6.11: 合影留念

6.7 FreeSWITCH&WebRTC 高手速成培训 2016 秋季班（北京站）圆满完成

小樱桃/2016.09

和往年类似，在第五届 FreeSWITCH 开发者沙龙活动结束后，紧跟着是我们的三天培训。在沙龙的强大对比下，培训显的有些平淡，但这次的人还真不少，这次的内容也更丰富，除了培训 FreeSWITCH 外，还增加了当下最热的 WebRTC 的相关内容，含金量有多高，可想而知。

我们的学员大部分来自北京，当然也有来自沈阳的、大连的、上海的、厦门的、广州的、贵州的、郑州的、成都的，真可谓五湖四海，看来距离的远近都不是问题。有些学员是在参加完沙龙后现场决定报名的，有些学员是第二次参加的，有些学员实在是没时间，只能参加一天，人家也不远万里的赶来，还有些学员在开课第二天了打电话报名参加，这不是为难我吗？为这些有学习热情的同学们鼓掌！

培训共三天，第一天和第二天都是杜老师讲 FreeSWITCH，中间作为福利让沙龙的嘉宾 Giovanni 给大家分享了《FreeSWITCH High Availability and Scaling for SIP and Verto》，第三天由远道而来的胡凌云老师讲解 WebRTC。

在开课之前同学们都早早的到了，并领到了杜老师的新书作为礼物。开课后杜老师先了解了大家的基本需求，并做了记录，以便在后期的讲课中重点讲解。培训主要围绕 FreeSWITCH 和 vars 配置文件、Dialplan、SIP 协议、透传、Profile 配置文件、NAT 穿越、录音、playback 的参数、Lua 脚本等内容，另外，还重点讲了案例分析，并对现场学员提出的 H264、OpenSIPS、视频会议、隐藏电话、并发、SQLite/pg 问题进行了现场教授。信息量比较大，学员普遍反应“学习很辛苦啊！”。



培训第二天晚上，我们举行了晚宴，正好赶上 Giovanni 过生日，真是热闹至极了。每个人都吃好喝好，还有人直接就谈成了“生意”，大家都笑说“看来他培训的动机不纯啊！”哈哈。潮流的吴总与赵总的加入让晚宴更加的完美。

最后一天的培训胡老师带领我们了解了 WebRTC 的来龙去脉，明白了发展现状、发展方向及应用场景，还知道了开发中的那些坑。真的是没有白来啊！下午颁发了结业证书，还剩下一些时间，杜老师又对大家的提问进行了现场解答。直到最后，还有好多同学追着杜老师问问题和拍照。

三天的培训很快就结束了，大家学的都很辛苦，能够完全的放下手中的工作来安安静静的学习真的是很难得，效果也是相当的好。有些人会问，为什么不录视频呢？在网上卖不是一劳永逸吗？其实，不是这样的，“书非借不能读也”，如果有网上的培训，你可能一年也学不完，因为你没有时间去专门学习，或者学着学着被其他的事情所打扰，这样的效果也不好；另外，还有一个更重要的问题就是，杜老师每次讲解的重点不一样，他是根据现场学员的实际需求来讲解的。所以，为了效果、为了质量，也为了能见上一面，我们还是希望大家能到现场学习。

另外，我们想说说报名的事，如果你想来培训，请早点报名，因为早报名优惠多，到了后期，我们的优惠力度就很小了，再讲价，那就是难为我了，呵呵！另外，越早报名，我们能知道培训的规模有多少，以便我们确定培训场地的大小，我们想尽量给大家提供一个舒适的学习环境。比如这次，很多学员是到了最后才报名的，我们的场地就显的有点小了。

本次培训在大家的共同努力下，圆满成功，感谢学员的认可与参与，感谢潮流公司给每位学员提供的话机，感谢杜老师和胡老师的付出，感谢 Giovanni 的友情出演，感谢泰富酒店的工作人员。

第七章 不知所言

都云作者痴，谁解其中味？

7.1 1024，我也是一名程序员

杜金房 / 2015.10.24

不知道什么时候起，1024 这个毫无意义的日子被程序员们定义成了自己的节日。

我也是一个程序员。

对 1024 这个数字，我也是很有感情的。2 的 12 次方，应该是 2 的 x 次方中能跟日期对应上的最大的数了。其实我挺喜欢一个叫 1024 的游戏，也有一个 1024 的升级版叫 2048，都跟 2 有关。

比起 1024，其实 1240 对我更有意义一些。说到 1240，不知道在众多的读者有没有共鸣的。1240 是上海贝尔的一种电话交换机，传说是 1240 个工程师做成，因而得名。我毕业后在电信部门工作，就是维护 1240 交换机。1240 交换机是一个纯分布式的系统，单元和组建之间使用消息机制通信，有部分系统失败不影响其它部分，重要的模块和单板（硬件电路板）都有配对的模块，如果某一模块失败，跟它配对的模块就会接管服务……虽然 1240 已几十年了，但其架构在今天也都还是很先进。

1240 有很强大的命令行人机命令接口，4296、136、7473……都是我们常用的命令。当年，我还用 C 语言自己开发了很多跟交换机通过串口通信的程序，自动发送命令，接收统计报告等。现在都事过境迁，刚跟以前的同事核对过，当年我工作过的地方的最后一台 1240 交换机也早已下架了。

不过，好在，今天我们要讲 1024，而不是 1240。

我出身贫贱，上大学时才开始学习电脑。但我发现我是真喜欢这东西。从最简单的 CAI、DOS、UCDOS 开始学，最早的硬件是 80286。不过，那时我还不会编程，只不过总是用 pctools 改改 CMOS 之类的。后来，学习 Windows 3.1、3.2、Win95、Word7.2、Word 97 等，最早学的编程语言好像是 FoxBase，如果算是个编程语言的话。然后自学了 Linux 和 C，毕业设计是用 C++ Builder 做的。

我始终不喜欢 Windows，因此，学习基本集中在 Linux 上，当时做东西主要就用 PHP（那时是 PHP3）和 C。其实，我的工作跟开发无关，但是，我工作的环境里有很多电脑和服务器我可以创造

条件折腾。我使用的最多的真正的 UNIX 系统是 Tru64 UNIX，当然也玩过 SCO，其它大多数时间就是玩 Linux，从 Blue Point、Debian、Slackware、Madarake、RedHat 到 Ubuntu 等都玩过。

我写程序一般直接在服务器上写，用 Vi，尝试过学 Emacs，但没学会。

那时候我的工作机器主要是 Windows XP，那时好像还没有 Putty，用 Absolute Telnet 通过 SSH 连到 Linux 上。直到最后一两年的时候我才用 Linux 工作，用 VMWare 虚拟个 Windows 处理工作上的事情（因为有些终端系统只支持 Windows，以及我们的 OA 系统），后来我找到一我们 OA 系统一个 Linux 版，工作起来基本就不用 Windows 了。

我曾经在我的电脑上插了一块板卡，装了 Asterisk，插上电话线，这样，我就可以在家里用 SIP 终端通过单位的电脑往外打电话。当时大家都感觉很神奇的，邻居偶尔也过来打长途电话。

后来，我离开原来的单位到了首都北京，到了一家外企。终于可以完全用 Linux 工作了，我的主要工作是 Admin，维护服务器。我们是一个 Rails 团队，因此我很快学会了 Ruby 和 Rails。有时候开发人员忙不过来，我也可以帮着改 Bug，也开发一些新功能。后来，我喜欢上了 Erlang，我们的 CTO 也喜欢上了，他就三下五除二写了一个程序，上线了。我的 Erlang 知识基本上是在给他改 Bug 练出来的。

Mac 基本是 Rails 团队的标配，后来我的工作平台转移到了 Mac 上。

我们用 FreeSWITCH 的时候那时还不到 1.0，在使用过程中我们发现一些 Bug，都汇报上去了。后来，我发现其实我自己也会改，就慢慢地提交一个补丁了。我还写了一个 G729 编码的模块（基于一个 G729 库），只是，由于版权问题后来 FreeSWITCH 官方不支持，甚至因此我把 Wiki 上的 G729 页面都改成只读的了。我也写了另一个模块，把 G729 编码的数据原样录下来，然后用其它工作就处理录音。这个，后来也被 FreeSWITCH 在核心中实现了。

在改 FreeSWITCH 代码的时候，我才真正学会多线程编程。最早，在 DOS 下面编程，只能用 TSR 之类的手段，或者在 Linux 上用 SELECT 之类的来处理多数并发。

我尝试过学 Python，只是读了一遍书，基本没动手写过程序，因此也不会写。有了 Ruby，也没必要再去学 Python 了。

只是，Ruby 确实慢。所以，后来我离职创业后，我们开发 Web 系统主要的语言就是 Erlang。也曾想过用 Elixir，但 Elixir 对于非 Rails 程序员来说，其实也没有什么吸引力。而我们团队里的人都没有用过 Rails 的。

开发 Web 系统当然离不开 Javascript。最早基本就是简单的 JS，再后来使用 jQuery，再后来用 Sammy.js。Sammy.js 是一个很轻量级的框架，类似于现代的 Angular 和 Ember。后来，我也尝试过使用 Angular 和 Ember，但上手太难，对我来说尚且难，对团队更是了。因此，一直只是做了几个实验项目，也没有深入。以前还尝试过 Backbone，项目中也用到一些。现在，基本主要集中在 React.js 了。

CSS 就用 Bootstrap。

对了，为了开发 FreeSWITCH 脚本，还学过 Lua。为了给客户写 Demo，还写过一点 Java。其实我看一本数据结构的书，全部是用 Java 语言描述的，算是有点基础吧。也写过 C#，是基于 Mono

的，没在 Windows 上试过。

我还学过 Action Script，就是 Adobe Flash 和 Air 里用的，当时还写过一个 Action Script 版的 ESL 呢。Action Script 其实是一种很好的语言，只是，Flash，Air，注定要成为历史了。

我用过一段时间 QT。我很喜欢 QT，但 QT 编译起来太慢。最近，QT 也支持直接写跨平台的移动端程序了，看起来很 Cool，没再试过。

我也学过 Go，但是，从来没用它写过代码，相当于不会。而且，我们现有的项目全是基于 Erlang 的，替换成 Go 的代价太大，也没必要。对于 Web 程序来说，或许 Go 合适，但对于实时通信的程序，Go 缺少 Erlang 能提供的实时性。

其实这些年开发越来越难了。除非你在大公司，否则，随着你越来越资深你会的和必须会的东西就越多。Rails 从 1.2 到现在都是 4 了，每个版本升级都得吐点血，何况有时还得维护很多版本的项目。C 语言倒是稳定，但操作系统一直升级，Linux 上编译器在升级，Mac 更是升级频繁，从 GCC 到 Clang 等，无数的版本依赖都是非常令人头痛的问题。

大家在争论什么是最好的语言时都以 PHP 举例。其实真是这样，我认为 PHP 最好的一点就是——PHP5 基本上 10 年都没变。所以，如果你是 10 年前学的 PHP，到现在都好用。现在虽然有 PHP7 了，但我不知道有多少赶时髦的。

写了这么多年程序，我认为我写得最好的完完整整的一个程序还是一个 PHP 程序，那个程序七八年了，到现在用，还很好用，虽然后来很多人开发了好多替代版本，但对程序使用的人来讲，还是我写的那个最好用。我当年使用了最先进的 CakePHP 框架，支持 Ajax 操作。

最近几天，为了给一个项目擦屁股，我不得不学点 Andriod 开发，又写了一些 Java。在这个特殊的国度，学 Andriod 真麻烦，光搭环境就得搭好几天。

我还曾经写过一个 iOS 的 App，那时我用 iPhone 3GS，iOS 的版本都不记得了。当时是使用 Titanium 框架写的，虽然后来读了很多 iOS 的书，始终没学会使用 XCode 和 Objective C。

现在有了 React-Native 了，这应该是个正确的方向。但是，不管怎么样，想练成 Andriod 和 iOS 程序员，不会 Java 和 Objective C 是不行的。更甚者，我们这些搞通信的，还需要会 JNI 等集成更底层的 C 代码。

学习永无止境。今天看到 Fenng 发微博说，“会更多的语言并不会让你更聪明”。然。

学了这么多语言，我最头痛的就是我的 Mac Pro 硬盘空间不够大，现在每种语言的环境和框架环境都至少要几 G 甚至十几 G 的磁盘空间，真是吃不消。新的环境都倾向于用 npm 管理依赖包，npm 为了支持多版本，愣是在每个项目中都下载安装依赖包，不仅浪费了空间，而且，很多包都得翻墙下载，程序员真不容易啊。说到翻墙，说出来都是泪，要不然怎么出了前一阵震惊移动互联网 XcodeGhost 事件呢。

罗罗嗦嗦这么多，1024 即将过去，明天，还得写程序。不过，不被别的事烦着，天天能静来心来写程序，才是程序员们最快乐的日子，是不是？

7.2 待价而估，技术服务如何定价

杜金房/2015.11.03



前几天看了 caoz 的一篇关于技术服务定价的文章，深感定价不易，一直想写点什么。但这几天特别忙，一直没空下下。偏偏来咨询的又多，就报价问题纠结不已。我就索性忙里偷闲，写一写吧。

我这几年一直做 FreeSWITCH 相关的技术咨询（也有外包），做过甲方也做过乙方，当然乙方居多，接触的人和项目多了，一直也想写点什么。先从作为乙方开始讲。

技术服务这东西，看不见摸不着，确实比较难达成一致，我这些年也钻了不少坑。但不管怎样，吃这碗饭的，怎么也要定下一套规则，这样，便有章可循，知道自己该干什么不该干什么。

一般来说，甲方倾向于按工时定价，就是要评估工作量需要多少个人天之类的。这种定价方法虽然很普遍，但其实是对双方都不是很有利的。一是你的工时单价很难要上去，二是具体工作量很难评估，三是容易造成倒挂现象。

关于工时单价，我们就遇到了这样的问题。甲方总是倾向于按社会平均工资或程序员平均工资来给你定价，所以，不管怎么样都是没有多少钱的。

关于工作量评估，真的很难，甲方一般不会认同你分析问题，查资料，调试甚至改 Bug 的时间。我听说有些做外包项目的由于不方便评估工作量，就招聘了很多廉价的甚至不会写程序的程序员天天到甲方公司上班，还装作很忙的样子，凑人头，凑工时。

关于倒挂，那就是，如果你 1 分钟帮他解决问题，那么，他会说你 1 分钟就搞定了还收钱？如果你两天两夜没睡觉，他觉得你很辛苦，花钱也值了。鉴于此，乙方一般 1 分钟能完成的问题也会拖两天，对双方都不利。

那么，该如何定价呢？

我也不知道，我参考了 FreeSWITCH Solutions 公司的方法做出如下方案：

1) 有一个基础询价包，1024，1200，或 1240 元。我大约花一小时的时间通过 QQ、Email、电话等做一个简单的需求分析，然后评估一个合理的价格。

这一步看起来有些不可思议，好多人都会说这是什么逻辑？我还不知道你能不能做为什么给你钱？但我会告诉你，其实这一个小时的性价比是最高的，因为对你来说，你基本上可以问任何你想知道的东西，即使我最终评估的价格高于你的预期，你也会知道为什么，再去找别人做的时候也明白你到底需要什么。

这一步对我来讲也很重要，我觉得连一千多块钱也不想付给你的人或公司，大概也不会愿意付你几万块钱。

当然，对于有的公司或项目这一千多块钱也可以不收，毕竟，我们不是为了挣这点钱，而是要看后面的大头。

2) 不按工作量定价，而是按心情。

我干我喜欢干的事情，至于我是 1 分钟干完，还是半年干完，真的与你无关。你只需要告诉我你需要的完成的时间点和最终能得到什么结果就行了。

如果你要的越急，价格就越高。

当然，为了到时候甲方能向公司有个交代，我们会按照这个价格反推一些工作量之类的加上去。

3) 按知识定价

我上面说的定价按心情，未免过于有些扯，客户也不认可，这时，就需要给客户讲故事了。

故事一：我忘了是哪位画家，在街头给一贵夫人画像，画完后要价 X 元，夫人嫌贵，说你就用了 30 分钟却要这么多钱。画家说：我画像用了 30 分钟，但我是用了 30 年才练到 30 分钟完成画像。

故事二：我前几天在“You pay what I KNOW”里讲过的，大家都知道的故事。斯坦门茨去给福特修电机，在电机上画了一条线，要价一万美元（当时很贵），他解释到，画一条线，一美元；知道在哪儿画线 9999 美元。

不管怎么样，要让你的甲方认可你的价值才是重要的。其实，好多时候，如果客户认可我的价值，我都可以不收钱，因为有时候确实就是一句话的事，他们可能好几天都搞不定，告诉他，其实损失不了什么。但是，如果我感觉客户不理解不认可这一句话值多少钱，那么，我就不会告诉他。

我经常遇到不理解的客户，比方有一天一个人来咨询，我说我认识一个人可能能做，你花 1024 买个咨询包我给你评估联络一下，能做就做，不能做就拉倒。他立马睁大眼睛（我猜的）说：“1024 就是个问路钱？要不是朋友介绍我过来找你我真是要喷人了”。我再也没理他。朋友归朋友，何况我不认识他也不知道他是哪门子朋友介绍来的，他不知道我为了帮他联络这些东西需要花多少时间，这些时间我能做的事情的意义远超过 1024。

所以，按知识定价，而不是按工作量定价。

4) 按产出定价

除了按心情和知识定价外，另外一种定价方式就是按甲方定价。你可以理解成看见有钱的就多要，没钱的就少要，但其实不然。

假设我花了一小时时间帮助了一个小公司，对他们来说产生的价值比如说是一万元。或许我同样花了一小时帮助一个大型的互联网公司可能他们就可以轻松挣到 100 万元。人生来就是不平等的。对我而言，同样是花了一小时，我对后者多要钱是正常的。

如果有人向你哭穷，那就少要点呗，甲方也是别人的乙方，大家都不容易，互相体谅，把事情做成了大家才有希望。

但如果碰到有钱的甲方，那不要白不要。

当然，人家的钱也都不是大风刮来的，没有人会白给你钱，所以，你总要说服对方。

其实大部分时候都是这样，我的有些业务是挣钱的，有的其实是不怎么挣钱的。但不挣钱为什么还做呢？小的客户也需要服务，我们不能忘了他们。所以，有时候，其实服务也是一种情怀。对于连一点点钱也不想出的小客户怎么办呢？也有办法。我们有社区，为了社区的壮大和发展，我们经常会免费回答一些问题，所以，如果你在我们公开的 QQ 群，BBS 等公开的论坛上咨询问题，我们只要有时间，还是很乐意回答的。如果你想得到私下一对一的支持，对不起，这个真得付费。

所以，选择免费的方案还是付费的方案由客户自己去评估。

5) 年费定价

除了按项目定价外，我们还收年费。年费是什么，相当于保险费。如果你这一年出了什么问题，随时可以咨询（当然，大的问题我们也可能会额外再要钱），你可以睡个好觉了。不过，我们现在交年费的客户也极少。经常有客户过来咨询一个紧急的问题，能不能 3 天帮我搞定之类的。我们其实能搞定，也不好意思多收钱，但是，我们也不会少收钱，越急就应该越多交钱对不对？我们通常建议这样可能会发生紧急问题的客户买个年度的咨询包。年费的定价比一次性突发性咨询的性价比肯定要高得多。

为了刺激大家交年费，我准备明年对一次性的咨询大幅度涨价，当然，年费也会涨价。2015 年的报价标准还有不到两个月的时间，如果有需要买咨询的公司提早跟我联系。

6) 飞机票定价

说到这里又想起了我们的培训业务。我们每年不定期会开 FreeSWITCH 培训课，定价 5000 元，但越早报名越便宜，类似飞机票的定价，我们需要尽早知道一次课上有多少人报名。

7) 没有免费的午餐

社区里的技术讨论都是免费的，但如果你进行商业活动，就没有免费的午餐。

对于一个卖产品的，经常会拿着自己的产品去给客户演示，讲解，甚至是测试，希望甲方能买下自己的产品，这需要很多投入的，同时，也养成了甲方的坏毛病。

我们的甲方很多都是想基于 FreeSWITCH 做项目或产品，不懂啊，想让我们去讲讲，甚至说可能会买我们的中间件产品。并强调他们也是在调研阶段，没有立项就没有预算。简单来说就是你去给他白讲，后面有块大饼但你几乎永远都吃不到。

遇到这种客户我们一般不去讲。因为这不符合我们的理念。调研阶段凭什么就不能花钱？如果他们研究一个月实验过 N 个同类产品得到结论 FreeSWITCH 是否适合他们，我去讲一天就能顶他们研究一个月的，那省下来 29 天的时间以及人力物力不都是钱吗？

我们是做服务的，因此，只要有服务付出就要有钱，没有免费的午餐。

上面说的大多是乙方的情况，下面再说说甲方。

我也做过甲方。有一个项目把一套 Android 的程序包给一个朋友，结果后来他太忙，没时间干，又另找了一个朋友，也比较熟，但从没一起做过项目，心里没底，但我也太忙，也没怎么跟进，结果就出现了一个情况，在 deadline 最后的关头，我连续 5 天没见到他人影。

这 5 天大致是这样的：

- 什么时候见个面？
- 在外地，明天回
- 回来了？
- 没？明天
- 回来了？
- 回了，车坏了，去修车，下午吧？（下午再打电话找不到人了）
- 今天见吧？
- 不好意思昨天下午喝高了，今天上午不行，我得去做另一个项目，下午看看吧（下午无数个电话找不到人）
- 哥们，今天无论如何见一面
- 真不行，那边的项目还没忙完……

甲方也有甲方的难处，如果找到不靠谱的人，也真是没办法。我这么多年也没这么狼狈过。不管这个项目做没做，做得怎么样，技术过不过关，人最重要。如果第一天的时候他就告诉我做不了，我有 5 天的时间做 B 计划，但我就是这样生生的被闪了 5 天。

这个项目的最终结果就是我被迫放弃了好多工作去学习 Android 开发，彻彻底底的失败。

所以，作为甲方，选对人最重要。在不知道是否选对人的时候，只能先小项目试水，或者，有备用计划。有时候，如果一个项目预算 5 万，不妨想办法拿出 10 万来找两拨人做，有时候多花一倍的钱比起项目失败来真不算什么。

作为乙方，一定要按期交付，一定要诚信。如果真不能按期交付，尽早提前说明原因，对自己对别人都很重要。这样才能定价越来越高。

所以，比起实际的费用，甲方更应该注重是乙方是否能按期限交付，一旦找到正确的人，不要太吝啬并及时付费才能会有双赢的结果。

7.3 挂号费和飞机票



前几天写了技术服务如何定价的文章，收到不少反馈。同时我们在报价过程中也遇到不少实际的问题，在此，继续跟大家一块分享下。

我们有两项基本的业务报价，一个是 FreeSWITCH 相关的咨询，一个是培训。一般来说，遇到询价的，我们就会先发一个通用的报价单过去。一般客户会有两种反应：1) 直接消失 2) 讲价。

就目前来说，直接消失的客户我们暂时没有精力去跟进。也许是我们的方法不对，我们以后要投入更多的人力来增加这方面的转化率。但目前，在我们人力还比较少的情况下，这种方式也可能正好帮我们消除了一些噪音。

对于来讲价的客户，多半还是希望跟我们合作的。但问题是，我们的报价并不怎么虚，基本没有什么议价空间。不过好在我们可以比较灵活的变通，Case by Case 的谈每一个项目。但不管什么项目，我们都会收取最初的一个基本费用 1240 元。

有些用户很难理解这一点：“我就是来问问你能不能做，不能做还收我钱？”

直到有一天我忽然想到了一个参照物——挂号费。

我们这个基本的询价包就等于是医院的挂号费。不管你有什么问题，挂号后我们才帮你看，诊断过程中我们可能还会让你交钱做个 CT、血检、尿检什么的，或许最终你还得“住院”。不管用户是否认可我们这种收费方式，至少他们比较容易理解我们的解释。

那么年费呢？年费其实就是相当于预交一年的挂号费。有问题可以随时问，但我们不保证你交了年费就给你解决所有问题，以后该打针吃药花钱看病情而定。

我还记得很小很小的时候看到我们镇上人民医院门口的对联：但愿人间常无病，不惜价上药生尘。我们的理解也永远是这样的。我们的目标并不是收你多少钱，而是希望，如果你有问题的时候，我们能帮上点什么。

如果你可能经常遇到问题，建议交一个年费。这样，我们可以有预留更多的资源给你更好的保障，我们可以多招聘和培养一些工程师帮你解决问题。当你遇到一个突发问题挂急诊的时候，会不会后悔当初为什么没有买保险呢？会不会因为我们收了较高的服务费而指责我们的道德底线呢？

再说说培训业务。我们采用了类似飞机票的定价，就是为了更好的控制到培训的那一天我们该派个 737 还是 A320 去接您。早报名会享受更多的折扣。我们遇到很多人到最后关头还来找我们砍价，我们也遇到有的单位向领导申请了预算但过了两个月才批准的，但问题是两个月后票价变了。总之，做一场培训也是很麻烦的事情，我们很难去满足每个特定的人的要求。

不多说了，一切来源于生活。咨询、培训，挂号费和飞机票。

7.4 我在 Mac 上的写作工具链

杜金房 /2015.08.13

有读者问我写书时是如何画插图又如何生成 PDF 的，今天，就跟大家分享一下。

先讲个故事。

大约 09 年底 10 年初的时候，我就想写点关于 FreeSWITCH 的东西，也希望最后能汇集成书。因此，我从最开始就研究怎么排版。

最开始，是从第二章写起的。当时直接使用 Textmate 写，使用 Latex 语法，用 PdfLatex 转成 PDF。事实证明，写的很痛苦。因为，跟那些写小说的人不同，我的书中有好多代码。大段的代码还好说，直接嵌入段落中的代码就比较麻烦，必须用一堆 \$\$ 括起来，而且，如果代码中有 \$ 就得用 \\$ 转义。不幸的是，我最开始写的那一章中有很多的 \$。

后来，发现写书是很遥远的事，就开始把内容放到我的博客上。博客是自己建的，使用简单的 Markdown 格式。Markdown 写起来很方便，而且兼容 HTML，一切都好。

但我一直忘不了寻求如何生成 PDF 的方案。也尝试过 RestructuredText 等方案，但都不完美。

再后来，一个自称是 Tim Yang 的网友鼓励我把书印出来。排版的事就又提上了日程。由于我时间不多，一个老朋友答应帮忙。后来，还真让她整理出了一个 Word 版。但是，问题是 Word 版里有很多地方需要改，而修改 Word 版，不仅是很累的活，而且，无法同步回 Markdown，并且我也不怎么喜欢（极其讨厌）Word。所以，我只好把那么精心打造的 Word 版给废了。再次寻找从 Markdown 生成 PDF 的方案。很幸运，我找到了 Pandoc。

Pandoc 是用 Haskell 编写的软件，能从各种文件格式中互转。比较典型的就是从 Markdown 转成 PDF、HTML 和 Word。

当然，它还是先将 Markdown 转成 Latex，再从 Latex 转成 PDF。通过精心调校以及一些 Shell 脚本的帮助（用于调整 Latex），终于能生成比较美观的 PDF 了。书也印了出来，名为《FreeSWITCH：VoIP 实战》。在第一届 FreeSWITCH 沙龙上卖出去几本，后来又间或有人来买，书也一直在更新。

但，PDF 比较美观，跟完美还差一个数量级。所以，我也一直在改进。当然，在过去的几年里，Pandoc 也一直在升级。

后来，我从 PdfLatex 转成了 XeLatex，效果要好很多（具体差别细节不记得了）。我的书中有很多脚注，写起来非常方便。而且，在使用中，我也慢慢学会了如何通过在 Markdown 中适当地嵌入一些 Latex 标记来做交叉链接。现在，除了还有下列两个问题外，已经几近完美了。

1) 我还没有搞定字体（我希望某些引用使用楷体，不知道为什么使用楷体后有些副作用），好在我可以通过改变字号变相的区别。

2) 图文混排控制力较弱。如果不在 Markdown 中增加 Latex 标记（我不想增加），就只能实现简单的图文混排，即图片永远占一行，而无法使用文字环绕。这种情况的一个最大问题就是一个竖版的图片，往往会占一整页。如果不想让它占一整页，只好修改图片，用白色或无色背景加宽。好在我会点 PS。

同样的 Markdown 也可以生成 ePub 格式，从 ePub 可以继续生成 Kindle 格式。这两种格式与 PDF 格式比起来，还不是很完美（因为 PDF 有 Latex 帮助，而 ePub 版不能利用 Latex 标签，如果需要更好的 ePub，需要一些相关的插件才行），但是，一般来说也够用了，在移动设备上有比较好的阅读体验。

不过，即使你的 Latex 和 PDF 再精美，出版社也不会用的。后来，出书的时机成熟后我发现，我的出版社只接受 Word 格式的稿子。他们说中国的环境不足以养一个会 Latex 的编辑。另外，Word 的修改和批注功能确实没有什么很好的替代品。我虽然跟他们说『我们可以使用 Git 啊』，但他们只是笑笑。其实我自己也知道，Git 是基于行的，不适合 diff 大段的文本（更别提中文单词和标点之间没有空格了）。

所以，我只好将我的 Markdown 通过 Pandoc 转成 Word 格式。再跟编辑们一遍遍地校对和修改。当然，这些修改就只存在于 Word 版了。除非手工同步修改 Markdown，不可能再从 Word 版转回原来的 Markdown。好在，书出版以后，我不需要再继续打印了。因此，也没有必要再保持 Markdown 跟最终的版本同步。而且，出版社给出了一个 Word 模版，通过使用一些快捷键，可以很方便地格式化一级标题二级标题以及代码块等，手工按这些快捷键处理一章也基本上就是几分钟的事（Pandoc 似乎也支持模板，但我没有试过，因为反正在编辑的时候还是要再次阅读和修改的）。

书于去年 6 月出版，出版时的名字叫《FreeSWITCH 权威指南》。

后来，在读了《Producter》一书（电子书）后，我也深受启发，在 SelfStore 上发了几本我的电子书《FreeSWITCH 互联互通》、《FreeSWITCH 实例解析》等。

当然，如果我今天只讲故事，没有干货的话，你可能觉得我是在做广告了。所以，接下来干货。

在写书时，要有很多的插图。据说 Mac 上最好的画图工具是 OmniGraffle，然而，我始终没有冲动试用它。而是尝试了很多不同的软件：Skitch、Dia、yEd、XMind、Keynote 等。这些软件的问题

是画得不好看，而且风格不统一。所以，最后，我又回到了程序员神器——Graphviz。

通过 Graphviz，可以使用很简单的 dot 语言画出很复杂的图，比如今天的题图，对应的 dot 源文件如下（虽然长，但很直观）：

```
graph G {
    ranksep = 3;
    ratio = auto;

    A [label = ""];
    B [label = ""];
    C [label = ""];
    D [label = ""];
    E [label = ""];
    F [label = ""];
    G [label = ""];
    H [label = ""];
    I [label = ""];
    J [label = ""];

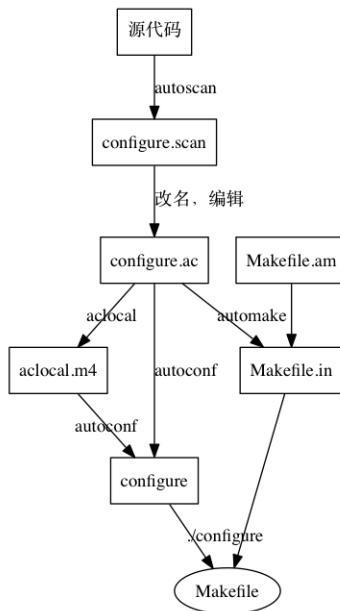
    A -- B;
    A -- C;
    A -- D;
    A -- E;
    A -- F;
    A -- G;
    A -- H;
    A -- I;
    A -- J;
    B -- C;
    B -- D;
    B -- E;
    B -- F;
    B -- G;
    B -- H;
    B -- I;
    B -- J;
    C -- D;
    C -- E;
    C -- F;
    C -- G;
    C -- H;
    C -- I;
    C -- J;
    D -- E;
    D -- F;
    D -- G;
    D -- H;
```

```

D -- I;
D -- J;
E -- F;
E -- G;
E -- H;
E -- I;
E -- J;
F -- G;
F -- H;
F -- I;
F -- J;
G -- H;
G -- I;
G -- J;
H -- I;
H -- J;
I -- J;
}

```

又比如下图：



源文件加上空行也只有 22 行（为了在手机上减少换行，取消了缩进）：

```

graph TD
    splines = false;

```

```

S [label="源代码" shape = "box"];
SCAN [label = "configure.scan" shape = "box"];
IN [label = "configure.ac" shape = "box"];
ACLOCAL [label = "aclocal.m4" shape = "box"];
configure [shape = "box"];
MakefileAm [label = "Makefile.am" shape = "box"];
MakefileIn [label = "Makefile.in" shape = "box"];

S->SCAN [label = "autoscan"];
SCAN -> IN [label = "改名, 编辑"];
IN -> ACLOCAL [ label = "aclocal"];
ACLOCAL -> configure [ label = "autoconf"];
IN -> configure [ label = "autoconf"];
MakefileAm -> MakefileIn;
IN -> MakefileIn [ label = "automake"];

MakefileIn -> Makefile;
configure -> Makefile [label = "./configure"];
}

```

而所有的 dot 文件，只是靠一个简单的 Makefile 和一个 make 命令就会全部都转成 PNG 图片了。同时，dot 文件还可以存在 Git 仓库里。

谁能告诉我，像下面这样的图（如图7.1）用 OmniGraffle 或 Windows 上的 Visio 画累不累呢？

当然，并不是所有插图都适合用 Graphviz。所以，除此之外，我还使用 Keynote。Keynote 比较适合画这样的图（是的，Internet 也是我用弧线画出来的，如图7.2）：

还有一个问题。那就是，以上工具都不适合画时序图。因此，我不得不又搜遍互联网，找到一个类似 Graphviz 的工具——mscgen。是的，跟我期望的一样，它可以使用类似 dot 的语法画时序图，大概就是下面这个样子，如图7.3：

搞过通信的人应该对上面那种图都非常熟悉。当然 mscgen 在 Mac 上生成图时对中文的处理有点问题。我只好首先用 mscgen 生成 svg，然后又用 Inksape 将 svg 转换成 PNG。

你还在用 Word 写 API 文档和产品说明文档吗？反正我们已经全部是 Markdown 了。虽然 Word 也可以进行『文件比较』，但怎么也不如 Git diff 来得顺手吧？

好了。这基本上就是全部的秘密了。如果大家感兴趣，我以后也可以跟大家分享一些相关的脚本和源文件。程序员万岁。

7.5 有求必应

Seven/2016.04.02

有的人说，开源社区应该是有求必应的，这样才能发展壮大。

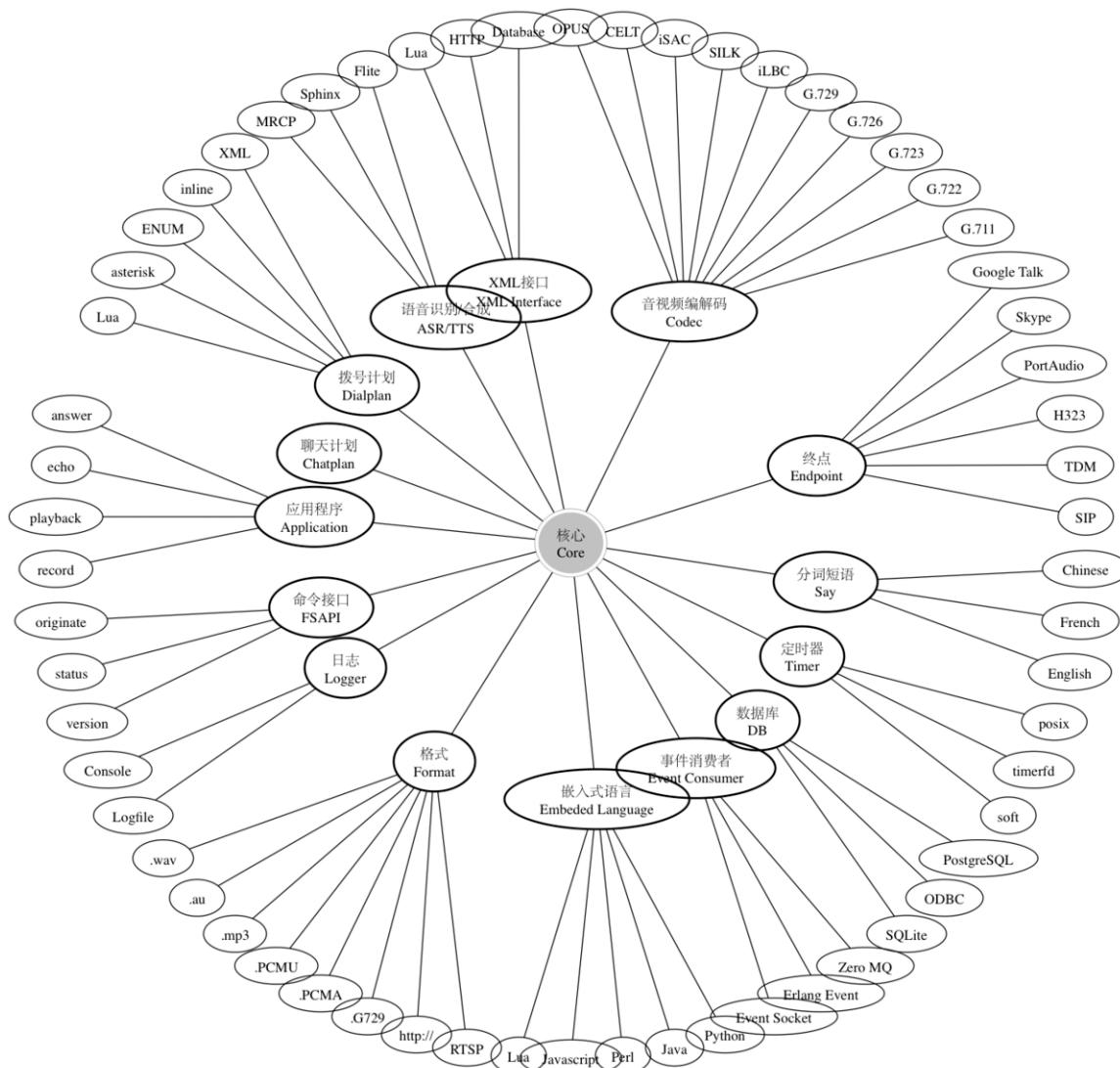


图 7.1: FreeSWITCH 模块结构

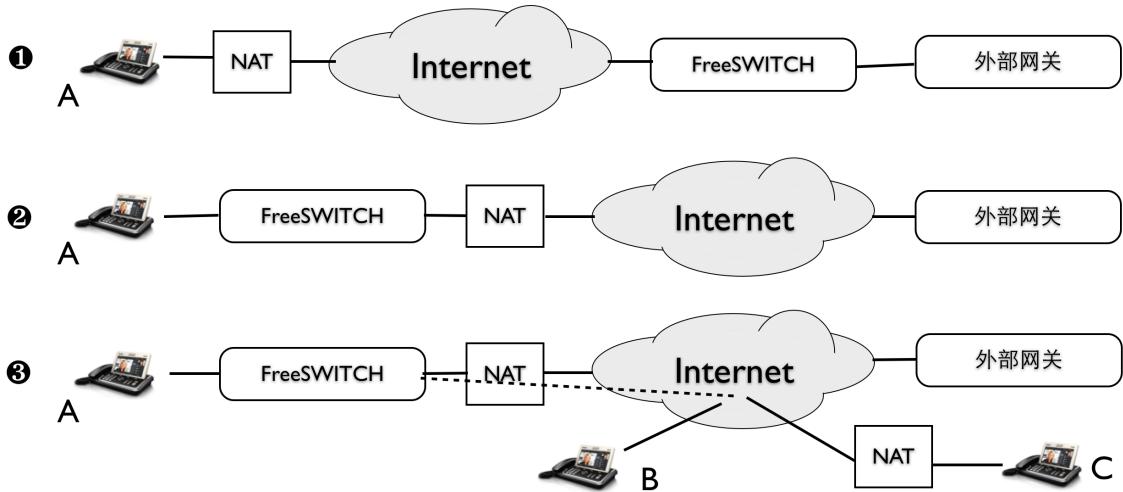


图 7.2: 用 Keynote 画的 NAT 示意图

我认为，也是，也不是。

有求必应的应，其实有两个意思，应答和应验。应答就是，不管什么问题，不管我会不会，我都会给你一个回答，应验就是，不管什么问题，我都会给你解决。

显然，我们不能做到后者，而且，在 FreeSWITCH 中文社区里，我们也没有做到前者。有的人声称他混过的开源社区都是有求必应的，我表示怀疑。

著名开源社区 OpenResty 的创始人春哥曾经说过：“相比其他开源作者，我自以为做得比较好的一个地方是，我总是尝试及时回复用户的各种邮件，即使我还有全职工作的时候。”我想，有求必应也不过如此吧（只是尝试而已，离“必”还远着吧）？

无论是全职还是兼职，春哥总是不遗余力的去推动 OpenResty 的发展，并且不求回报，不考虑短期利益，最终熬到了一个好的结果（罗永浩把 T2 发布会的门票都捐给了 OpenResty 基金会）。其实也很危险。

我们来看看我理解的有求必应。

首先看谁求。如果某个人在社区很活跃，经常帮助别人，经常资助开源项目，资助社区，那他提到的问题显然就会多收到一些回应。当然，他收到回应可能并不一定是因为他的资助，而是这样的人一般知道开源社区的规则，不会问一些垃圾问题。

什么是垃圾问题呢？典型的就是那些在 QQ 群里问 FreeSWITCH 打不通电话怎么办？谁能帮我配置一下 TLS 之类的需要用一本书去回答的问题。提问的人不知道自己去做功课，直接拿来问，即使我们想回答，我们也没法回答。

所以，“求”的人先要做一些基本的功课，才能指望有人会“应”。

其次，看你求什么。上面已经说了垃圾问题的事情了，如果你求别人帮你解决一个垃圾问题，自然你得不到好的答案。另外就是，你求答案的目的是什么。如果你是为了学习，那么大家很乐意回答你。但如果我说“急！！！，明天就给客户交差了，今天晚上必须搞定，跪求！！！”之类的话，那

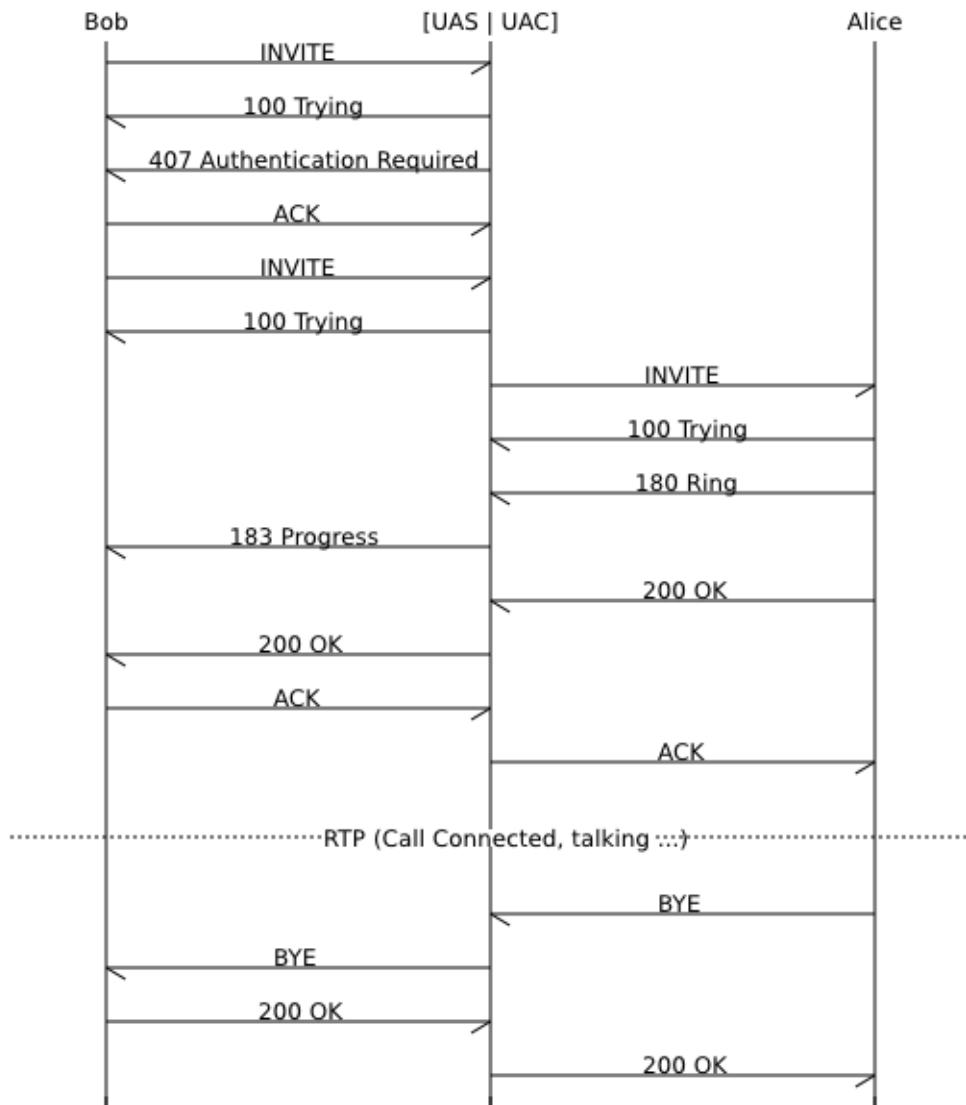


图 7.3: SIP 信令时序图

么，多半没人理你。原因自然很明了，你跟客户之间的协议跟别人没有任何关系。别人也没有义务去帮你甚至你的客户去解决问题（说白了，去帮你挣钱）。

再次，看应该怎么“应”。如果你提的问题，大家都不会，需要所有人都回答你“我也不回”吗？其实如果你提了一个所有人都“不会”的问题，那，要么你问错了地方，要么你是提了一个垃圾问题。

最后，永远不要要求别人“私下”帮你解决问题。这个说过多少遍了，原因很简单：1) 没有人欠你的；2) 私下回答你一个问题对社区没有任何好处，而在公共空间里的问题也会帮到别人；3) 实际上，即使有人愿意私下回答你，那么他在睡觉或修假的时候你不能及时回应，反正你在公区的空间有更多的概率有更多的人回答你。

说说我们社区。

这些年，我觉得我回答的最多的问题就是：

- 1) 什么操作系统？FreeSWITCH 什么版本？你改了哪些配置？…
- 2) 按 F8、sofia global siptrace on 然后把 Log 贴到 BBS 上
- 3) 看《新手必读》学学排版…

我们希望所有人在提问之前都自己做些功课。另外，我们的指导思想是授人以鱼不如授人以渔，我们希望通过我们的启发，能让别人自己找到答案，在以后遇到类似的问题时自己找到答案。所以，我们是为了最大限度的“教会”大家。但是，当一个人来到社区只是贪婪地索取时，我们坚决不开后门。当然，实际上我们也开了，后门就是花点钱找商业服务。

开源社区的壮大，靠的不是一个人的力量，而是需要大家共同的努力。我们希望，大家在学会之后，都会主动帮助新手，都会贡献一些文档、测试，汇报一些 Bug，甚至贡献一些代码等。当然，不会写代码的也可以贡献自己的资源，如服务器，带宽，甚至赤裸裸的金钱等，这样，社区才会发展壮大。

所以，即使有求必应，我们的应也是建立在为了社区的壮大和发展，而绝不是为了你个人、一个商业项目倾其所有的帮助你。

说到这里，我们的主题就结束了。但既然说的商业支持，我们就再多说几句。

就我们的经验来讲，不想交钱的一毛钱都不想交，所以，你给他报价多少钱他都嫌贵，这不是我们的客户。想交钱的，有的单位不给出钱，有的程序员自己掏腰包给我们交咨询费（当然我们感动他这种精神，会适当便宜些，其实我们呼吁所有有这样员工的老板都给程序员加薪）。有的单位能理解我们（或许不一定理解，只是有预算而已），我们就会不停地收钱。可能有人说你欺负老实人。说白了还真有那点意思，但是他们真亏了吗？未必，很多时候我们都会给他们提供超值的服务。

好了，不多说了。有求必应对开源社区来说是危险的，我们不会那么做。

7.6 钱

Seven / 2016.04.01



今天的标题直接就是赤裸裸的钱。

谁会跟钱过不去呢？

还真有。前几天，我们上线了『FreeSWITCH 高手』QQ 群（我承认这个名字起得不太好，好多人认为只有高手才能进，其实是进了才能成为高手:D）。收费 512 元。结果有的同学就不高兴了，说开源的社区还弄个什么高手群收钱，真让人恶心，云云。

辩解一下：

- 1) 谁说开源不能收钱？连开源（实际上是自由软件）的鼻祖 Richard Stallman 大叔都说开源可以收钱，愿意收多少收多少。
- 2) 现在社会，什么东西离了钱能转吗？我跟他说就连我们现有的 QQ 群，也是有人每年在默默地交群费，他还很惊异—开个 QQ 群还得交费？
- 3) 他还辩称他在过多少多少社区，都没听说过收钱的事。没听过不能说明没有，只能说他见过的太少而已。
- 4) 退一万步讲，我们弄个收费群，我们自然有我们自己的规划和定位，你不交钱不进就是了，怎么恶心着你了？

从 09 年开始，我们 FreeSWITCH 社区运营也有些年头了。除了有一年戴总带着募了点捐外，其它的钱都是从我们几个的个人腰包里来的。就连我们搞的每年一次的 FreeSWITCH 沙龙，虽然有赞助商，我们除了搭上人工外，多多少少还是亏些钱的，更别提好多朋友都是从外地自掏腰包过来捧场，我欠多大人情呢。

谁会跟钱有仇呢？

钱不是万能的，但是能衡量一些东西。举一个例子，有的人自己研究 FreeSWITCH 遇到问题半个月没有进展，或者雇了一个工程师搞了两个月问题没搞定，他觉得工程师笨，但是，我可以花几分

钟就能帮他解决这个问题，或者，退一万步讲，我告诉他此路不通，不用花两个月去钻牛角尖了。但是，他认为我几分钟也就值几毛钱，谈钱还伤了感情。却不想想他付给了他的工程师多少工资都打了水漂。

当然，这只是个例子而已，跟今天的主题其实无关。

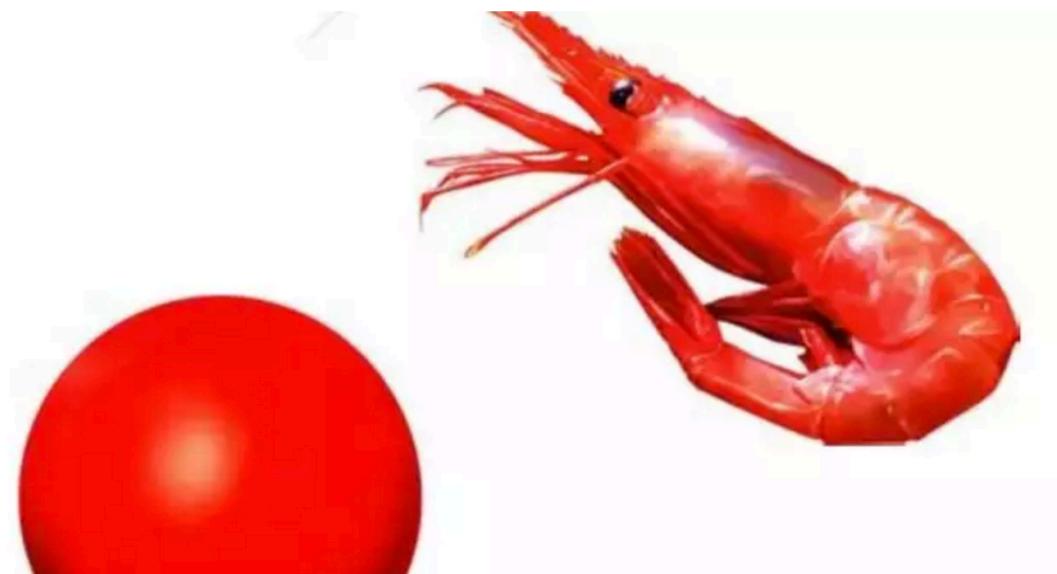
其实今天主要想说的是，开源跟是否收钱，没有半毛钱关系。解释得再清楚一点-没有必然的联系，也没有一丁点冲突。有人收有人不收而已。

感谢交钱加入我们 QQ 群的朋友们。有了你们的支持，我们社区才会变得更好。

话说到这里其实我们无意去打压那些质疑我们的人。相反，批评与自我批评，我们欢迎这样的声音。但同时，在对我们提出批评的同时，也请接受我们的批评。我们大家都是为了开源大业，还是那句话，对事不对人。

7.7 这事不归我管

Seven/2015.10.12



国庆几天我哪里都没去。不过，听说这几天青岛的大虾火了。

关于这些，网上的文章已经铺天盖地了，不想说太多。只想说说踢球的事。

作为一介草民，找政府办事是最头痛听到的一句话就是“这事不归我管”。

那这事该谁管呢？110？不，物价局？人家放假了。其实，关于价格的问题还是得工商部门管，不过，客人受到人身威胁，就该 110 管了吧？

十多年前吧，我还在电信部门上班的时候。那时候我们推行一种服务叫“首问责任制”。就是不管什么问题，只要找到你了，你就要负责到底。需要协调的你去找相关部门协调，而不是到处踢皮球。

我在想，如果我们的政府部门都实行首问责任制该多好呢。

但是，实际上这个制度是很难执行下去的。因为这要求每个人每个部门都对相关部门的业务比较了解。还有，你必须能协调的动。

所以，协调所有部门真的不是每个部门的本职工作。

不过这些天看网上大家发的文章，我都没看到关于 12345 的。

12345 是为了为人民服务开通的市长热线。我对其运行规则不是很了解。但是，我有一次由于技术原因确实去了一个市长热线的呼叫中心。里面的人很少，电话很少，呼叫中心业务软件也很简陋。

为什么电话会少？我没有答案。不过，从跟他们话务员了聊天中了解到，其实话务员也大多都是从各机关部门抽调上来的，服务一段时间然后换人。

但不管怎么换。话务员在的时候就要了解相关业务。从我看来，他们一般的职能就是接到电话以后，把相关情况反映到相关部门，让相关部门进行跟进。

好了，既然他们能把电话跟相关部门反应，那么，他们是否知道什么事该哪个部门管呢？当然。有些事情确实需要多个部门联动呢？很简单啊，12345 从中协调呗。

所以，真希望 12345 能协调所有部门，如果有的部门踢皮球，让他们踢到 12345，如果 12345 管不了，我觉得让市长亲自值班是个很好的选择。

所以，12345 呼叫中心系统简陋，电话少，我觉得真正原因是，它没有受到应有的重视。

当然，简陋的电话系统不是我们做的。如果我们做，做好后能解决踢球问题的话，不给钱也要给他们做好啊，这是关于我们民生的大事啊，是不？

读者中，有做过 12345 的没，给大家说说呗？有用过 12345 的没，也给大家说说呗？

第八章 精英

总有一部分人会成为精英，总有一部分人会成为精英。

8.1 FreeSWITCH 精英群上线了

小樱桃/2016.03.21

在一些准备之后，我们成立了 FreeSWITCH 精英 QQ 群（收费）。

在精英群里，您可以：

- 认识一些跟你一样愿意付费获取知识的人；
- 你在为开源事业做贡献；
- 你提的问题会比社区支持优先得到回答，或者会得到更耐心的回答；
- 如果你需要买 FreeSWITCH 商业支持，可以免收 1240 元“评估询价”费；
- FreeSWITCH 专家 Seven Du 会定期为大家提供群内实时交流的机会；
- 优先知道一些内部消息，或优先获取 FreeSWITCH 新品试用机会；
- 禁止不经允许乱发广告，可能会被无警告地踢出群（但还可以继续交钱重进）；
- 如果 QQ 群中毒了乱发消息可能会被清除出群，如果重进需要重新交钱；
- 一般来说不要闲聊。

会员费 1024 元/位/年。2016 年 7 月 1 日前进群 512 元/位/2 年（自然年）。

而且从前几期的群内活动来看，大家的收获都是很多的，杜老师也时不时的总结给大家看了，关键是进了群之后，大家的水平都有所提高，不论是提问的水平还是其他方面，总之，这个群还是很不一样的噢！想进来看看吗？呵呵

感兴趣的可以到[有赞商城](#)购买进群门票。付款后会有专人联系加你 QQ 并拉你入群。

8.2 FreeSWITCH 精英群第一次活动小记

小樱桃/2016.04.28

整个过程历时一小时，一开头杜老师就开启了“AMA”的模式即“Ask Me Anything”。在紧凑的时间内，主要回答了 FreeSWITCH 视频录制的原理、存在的资源损耗问题；FreeSWITCH Proxy 模式过于简单，并给出一些组合建议，如 Mediaproxy、OpenSIPS 等。

本次活动好多人迟到了，好多人不知道，还有好多人参加不了，呵呵，没关系，以后我们每周都会。FreeSWITCH 精英群已经诞生好久了，成员也在不断的壮大，在经过一个月的准备后，今天下午两点我们组织了第一次群视频活动。

由于老杜第一次用 QQ 群视频聊天，再加上用惯了 MAC 系统，换到 Windows 上有点不习惯，折腾了半天。另外，由于这是第一次活动，大家也没有太多准备，再加上还是上班时间，不能畅所欲言，大部分人只能通过打字的方式把问题提出来，杜老师再根据大家的提问进行专项回答。但整个过程还是非常顺利的，有准备的同学都得到了满意的答复，这种只针对你一个人的专项快速回答还是很让人激动的。组织一次类似的活动，大体安排是这样的：一周一次，时间段分为下午 2 点到 3 点和晚上 8 点到 9 点。比如，本周是下午的活动，下周就是晚上的活动了，听说有的同学周四晚固定加班，因此我们把活动暂时安排到每周三。下周的活动是周三晚 8 点-9 点，如果临时有变，我们会及时通知的。

大部分人晚上的活动比较自由，可以自由发言，如果聊的嗨了，还可以延长一下时间，呵呵！所以，大家可以提前准备一下，除了提问的方式外，大家也可以上台分享一下自己的经验。总之，我们的形式是多样的，最终目的是想让大家在这个群里真正成长起来。



图 8.1: 视频中

如果还有没来得及加入 FreeSWITCH 精英 QQ 群的，加我们公众号“FreeSWITCH 中文社区”吧。

8.3 FreeSWITCH 精英群第 N 次活动小记

杜金房/2016.05.25

不记得是第几次了。今天晚上，我跟大家在 QQ 群里聊了聊。

主要聊了 mod_verto 这个模块，以及 verto communicator，后来又有人来，没赶上，我就简单的写一写吧。

mod_verto 这个模块以前讲过，是 FreeSWITCH 实现了一个私有的信令（其实也不算私有，人家都开源了啊），使用 websocket 传输一些 json 消息建立通话。

verto communicator 是一个 verto 的客户端，运行在浏览器里面。

安装 verto communicator 很简单，进入源代码的 verto communicator 目录

```
cd html5/verto/verto_communicator/
```

如果你是用 Debian 8 的话，直接执行下面的脚本就好了：

```
./debian8-install.sh
```

比较悲催的是，我运行了，但是没安装成功。

verto communicator 是用 Angular 框架写的，因此需要 nodejs, npm, grunt 等一大堆东西，在这个神奇的国度里，很多依赖都下载失败了。

我的 Debian 在阿里云上，没架过 VPN，后来还是把我本地打包好的文件上传到服务器上去了。话说阿里云本来不提供 Debian 8，我是从 Debian 7 上手工升级的。最新的 FreeSWITCH 需要用 Debian 8 才好，题外话。

我们近距离看一看脚本都干了些什么。

```
cat ./debian8-install.sh

#!/bin/bash
apt-get update
apt-get install npm nodejs-legacy
npm install -g grunt grunt-cli bower
npm install
bower --allow-root install
grunt build
```

很简单一个脚本，做前端的应该都很熟悉这些命令，这里就不多说了。总之，如果顺利安装并且 grunt build 的话，会在当前目录下产生一个 dist 目录，里面就是 html 的根目录。

话说，放 html 要有个 Web 服务器。典型的有 Apache 和 Nginx。官方的 Nginx 有 Apache 的配置例子，我则比较喜欢 Nginx，都可以配好。不过，由于最新版的 Chrome 要求使用 WebRTC 必须使用 https 和 was，所以，还需要在 Web 服务器里配置证书。这一步有些麻烦，先不讲了。

下面说一个简单方法：

将下列内容放到 FreeSWITCH 的 `verto.conf` 里，放在 `</profile>` 之前：

```
<vhosts>
  <vhost domain="localhost">
    <param name="alias" value="freeswitch.org"/>
    <param name="root" value= "/var/www/html"/>
    <param name="index" value="index.html"/>
  </vhost>
</vhosts>
```

这样，FreeSWITCH 就成了一个简单的 Web 服务器了，支持 http 和 https，这样，测试起来不用配 Nginx 或 Apache，方便多了。

用 Chrome 浏览器访问 `https://你的 IP:8082/` (注意 s) 就可以看到 Web 页面了。注意，要设置正确的 root 目录。我一般把 `verto communictor` 放在 `/var/www/html/vc` 目录中。

第一次访问时 Chrome 说证书不可信，继续就可以了。因为 FreeSWITCH 默认生成了一个自己签名的测试证书。注意，有些浏览器如 FireFox 可能会挑证书。如果实在不行那你得买个真证书了。

打开上述网址就可以登录了，保持 FreeSWITCH 运行，输入你的姓名和电子邮件，登录后呼叫 3500 就可以进入一个测试的视频会议。如果有多几台电脑的话将看到更好的效果。

好玩吧。

更高兴的是今天晚上找到了直播写代码的方法，以后直播写代码有望了。

由于我一直用 Mac，Windows 好多年不用都不熟了，而且，Windws 上各种弹出广告总会让我找不到北。但没办法，QQ 群视频只能在 Windows 上用。所以，我就用它直播。群视频中可以共享桌面，我共享了一个 Putty 客户端 (SSH 客户端)。ssh 到我的服务器上，然后，我在 Mac 上也 ssh 到服务器上，执行 tmux (终端共享软件，类似于 screen) 以及 tmux attach，这样，就可以在 Windows 和 Mac 上看到同样的服务器界面了。我在 Mac 上敲代码，在 Linux 上运行，在 Windows 上直播出去，Yeah!

上图：

三人行，必有我师。今天还有人告诉我 QQ 手机版也可以看群视频，我还是刚刚知道呢，效果还不错呢:)。

```

2. dujinfang@seven.local: ~ (ssh)
top - 21:03:37 up 2 days, 4 min, 2 users, load average: 0.22, 0.39, 0.42
Tasks: 90 total, 1 running, 89 sleeping, 0 stopped, 0 zombie
%Cpu(s): 14.5 us, 0.7 sy, 0.0 ni, 84.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.4 st
KiB Mem: 504828 total, 356616 used, 148212 free, 13220 buffers
KiB Swap: 0 total, 0 used, 0 free. 74812 cached Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1308 root 20 0 682012 5432 3880 S 0.3 1.1 3:36.67 AliHids
1 root 20 0 28696 3192 1480 S 0.0 0.6 0:02.72 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root 20 0 0 0 0 S 0.0 0.0 0:04.64 ksoftirqd/0
5 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/0:0H
6 root 20 0 0 0 0 S 0.0 0.0 0:02.98 kworker/u30+
7 root 20 0 0 0 0 S 0.0 0.0 0:37.89 rcu_sched
8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_bh
9 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
10 root rt 0 0 0 0 S 0.0 0.0 0:01.09 watchdog/0
11 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 khelper
12 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kdevtmpfs
13 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 netns
14 root 20 0 0 0 0 S 0.0 0.0 0:00.00 xenwatch
15 root 20 0 0 0 0 S 0.0 0.0 0:00.00 xenbus
17 root 20 0 0 0 0 S 0.0 0.0 0:00.06 khungtaskd

[0] 0:bash 1:bash 2:bash 3:bin/freeswitch 4:top* 5:bash- "AY131214074648467915Z" 21:03 25 May 16

```

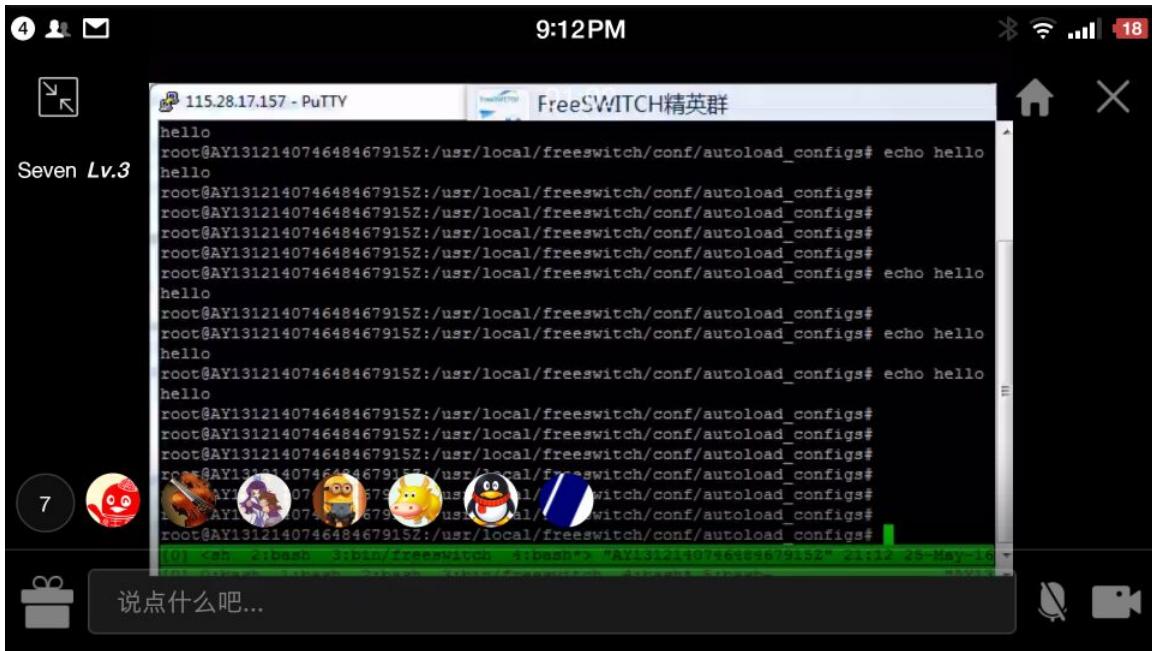
```

115.28.17.157 - PuTTY
top - 21:04:19 up 2 days, 4 min, 2 users, load average: 1.07, 0.55, 0.47
Tasks: 90 total, 2 running, 88 sleeping, 0 stopped, 0 zombie
%Cpu(s): 15.1 us, 0.7 sy, 0.0 ni, 83.4 id, 0.4 wa, 0.0 hi, 0.0 si, 0.4 st
KiB Mem: 504828 total, 363384 used, 141444 free, 14096 buffers
KiB Swap: 0 total, 0 used, 0 free. 78604 cached Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
1119 root 20 0 84532 55408 1212 S 0.3 11.0 4:31.61 tmux
1 root 20 0 28696 3192 1480 S 0.0 0.6 0:02.72 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kthreadd
3 root 20 0 0 0 0 S 0.0 0.0 0:04.64 ksoftirqd/0
5 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/0:0H
6 root 20 0 0 0 0 S 0.0 0.0 0:02.98 kworker/u30+
7 root 20 0 0 0 0 R 0.0 0.0 0:37.90 rcu_sched
8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_bh
9 root rt 0 0 0 0 S 0.0 0.0 0:00.00 migration/0
10 root rt 0 0 0 0 S 0.0 0.0 0:01.10 watchdog/0
11 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 khelper
12 root 20 0 0 0 0 S 0.0 0.0 0:00.00 kdevtmpfs
13 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 netns
14 root 20 0 0 0 0 S 0.0 0.0 0:00.00 xenwatch
15 root 20 0 0 0 0 S 0.0 0.0 0:00.00 xenbus
17 root 20 0 0 0 0 S 0.0 0.0 0:00.06 khungtaskd

[0] < 2:bash 3:bin/freeswitch 4:top* > "AY131214074648467915Z" 21:04 25-May-16

```



8.4 FreeSWITCH 第 N+1 次活动小记

这次，到场的人数较少，但杜老师兴致丝毫不减。直接就讲起了代码。

跟上次一样，在QQ群里直接一个视频窗口，窗口用Putty连接到一个Linux服务器上，杜老师在Mac上用SSH连接服务器，通过tmux共享屏幕。

杜老师先分享了一个Bug

<https://freeswitch.org/jira/browse/FS-9266>

这个问题是早上解决的

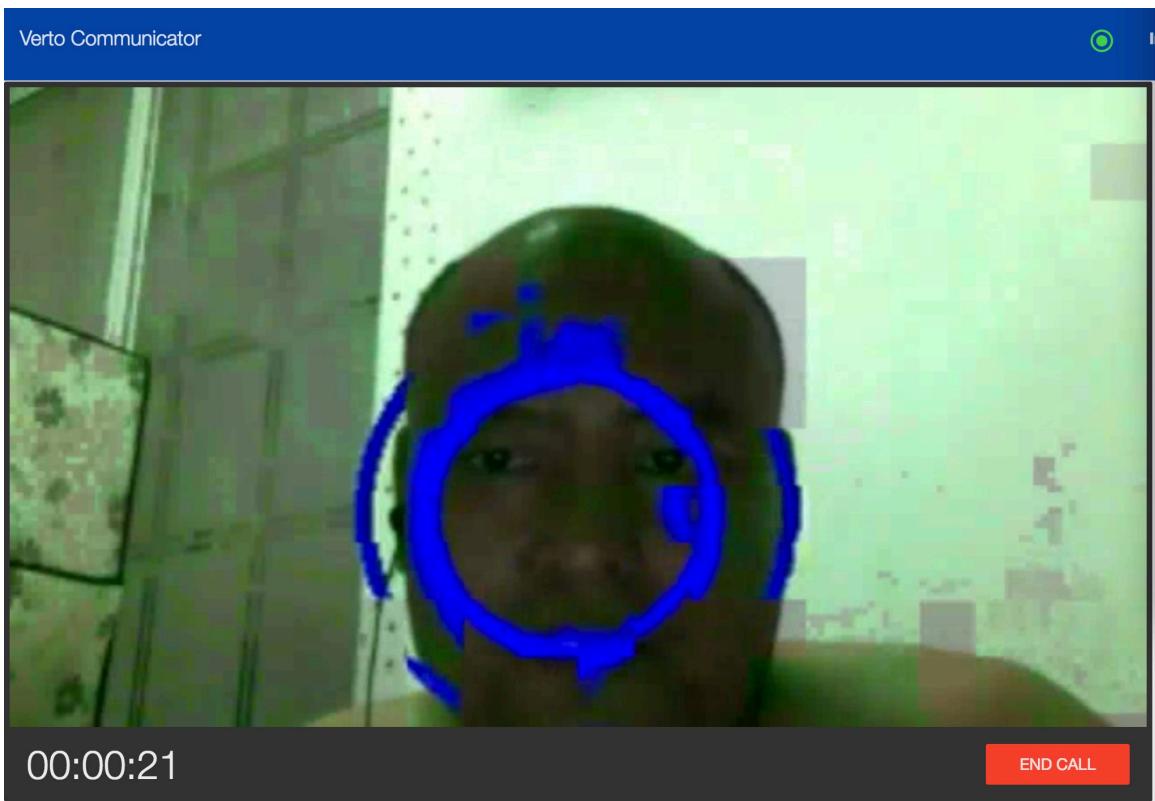
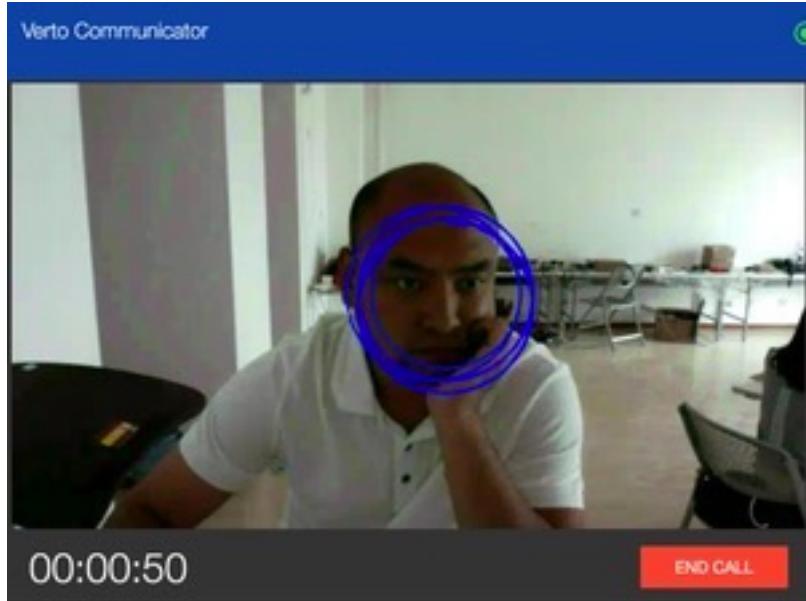
<https://freeswitch.org/fisheye/changelog/freeswitch?cs=c0499fbb22b42839b7df8c7bcc8a15dfcb6c1c4e>

在某一次重构代码的时候，引入了这个Bug，只是，由于mod_cv这个模块用的人少，竟然一直没人发现。

解决完这个Bug，又发现了另外一个问题：在用mod_cv做人脸识别的时候，返回的画质会越来越差，直到遇到下一个关键帧，至少看起来是这样。

杜老师跟大家把代码过了一遍，加入了一些日志，不过，发现可能是画圈的时候引起的问题。

是的，FreeSWITCH 支持人脸识别。甚至能通过检测指定的动作控制 FreeSWITCH 的行为。



时间所限，并没有在这次代码分享活动中解决该问题。不过，后来该问题还是解决了。确切的说是 Anthony 解决的。

<https://freeswitch.org/jira/browse/FS-9267>

修复在这里

<https://freeswitch.org/fisheye/changelog/freeswitch?cs=b0be5d67375fb8b23bccf2e4e544bc8a2afeb4e1>

后来查明问题的原因并不在于`mod_cv`，而是…

好吧，解决问题的过程是这样的。该问题以前是不存在的。但现在出现了。经过一番调试，怀疑这个跟 VPX 有关，而不是 FreeSWITCH 本身。后来用 H264 编码试了一下，果然，没有此问题。

所以，怀疑可能是 libvpx 库升级后进行了一些优化。由于`mod_cv`会改变图片数据的内容，但这个改变 libvpx 可能不知道，因此，libvpx 就傻了…

最后，为了解决这个问题。只好不让`mod_cv`改变 libvpx 原始的图片。每个图片都复制一份。

嗯，是的，libvpx 所做的优化起了副作用。当然，不能说是 libvpx 的错，但是，确定在使用`mod_cv`的时候，就不能享受这些优化的好处了。

好在现在还没有人用`mod_cv`识别很多 Channel …

第九章 旧文

从微信公众号上找到的一些旧文章。这些文章有些历史印记，因此，把它们整理放在这里。

9.1 圣诞快乐

Seven/2013.12.25 本文是 FreeSWITCH-CN 微信公众号上第一篇文章

今天是圣诞节，对我来说，是个有纪念意义的日子。它的纪念意义足以让远在千里之外的我在圣诞前夜赶回家。

时间过得很快，FreeSWITCH 从 2005 年开始构思，到现在，都快有 9 年了。而我从 2008 年春接触 FreeSWITCH，到现在，也已经快满 6 年了。还有，FreeSWITCH-CN 中文社区是在 2009 年底成立的，到现在也整整四年了。

在这些年里，我常常记起朱自清《匆匆》里的句子——“当我察觉他去的匆匆了，伸出手遮挽时，他又从遮挽的手边过去了。”

远的就不说了，从 FreeSWITCH-CN 社区创建开始说吧。当时，我很幸运，找到一个在国内还从来没有人干过的事，误打误撞成为了所谓的“FreeSWITCH 中国第一人”。便对于这个称谓，我是失职的。朋友跟我说过，这些年，你挑着 FreeSWITCH 的大旗，都为社区干了些什么？

说起来，在创建之初，我还真为社区干了些什么的。至少有了一个网站，跟大熊同学共同维护着一个 QQ 群，写了一些关于 FreeSWITCH 的中文的文章，并有一本初具规模的 FreeSWITCH 实战电子书。QQ 群也已有近千人。但近两年，我变得出奇得忙，因此，社区的成长速度不快，我有很大的责任。

因此，值此圣诞节来临之际，在 FreeSWITCH-CN 整整四岁之时，我决定再努力一把，为社区多做些事情。

从何开始呢？这还得从另一个人说起。

如果大家不熟悉的话，可以关注一下新浪微博上的 @Fenng，以及在微信中搜索“小道消息”，就能找到他。

我从 Fenng 先生以及他的小道消息中学到不少。因此，现在，决定像 Fenng 先生那样，也隔三差五地在 FreeSWITCH-CN 微信公共平台中发一些消息。到现在为止，我的工作仍然很忙，所以我不能保证每天都发，也许隔三差五也发不了，但，无论如何，我决定开始了。

初步的内容打算是——我逐渐分享一些这些年我对 FreeSWITCH 的一些认识、经验。并且，我会把大家在 QQ 群中问到的问题，有选择性地回答几个。当然，如果大家有好的意见，也欢迎跟我提，如果我有时间我会跟大家互动。当然，我也可能会分享一些跟 FreeSWITCH 完全不相关的想法。

有一件事我不确定的是，代码、命令行以及 Log 在微信中的排版会是什么样。我也是第一次在微信后台学习排版。因此，不管好不好，大家多给一些鼓励，并给我一些时间，我希望我的排版在不久的将来能达到“小道消息”的水平。

其实 FreeSWITCH-CN 的微信公共平台很久以前我就申请了，只是，还是因为忙，一直没有张罗，因此粉丝也不多。今天在 QQ 群中我说了我的想法，并说只要粉丝达到 50 名我就开写，刚才看时，粉丝已经 59 了。想着我的粉丝，今天晚上带小孩出去吃饭，不到 8 点就赶回来了。

好了，罗嗦了这么多，今天先不谈技术了。大家有什么想问的，留言，我慢慢回答。当然，我也可能不回答。

最后，祝大家圣诞快乐！

9.2 我 Donate，我快乐

2013-12-30/Seven Du/FreeSWITCH 中文社区

昨天由于太忙，没有发文章。今天看了一下统计信息，3 篇图文信息的阅读人数分别为 34、41、20，虽然最后一篇发的比较晚，但也过去两天了，看来，周末时间看的人比较少。大家可能都去娱乐了吧，看来周末我小休一下也是对的。

另外，自从我加了主菜单以后，有的网友已经在使用该功能了。不过，大家只是试了一下，选了一两篇文章，没有三篇全部都看完的。这可能说明标题还不是很吸引人，以后还得改进。

话说前天终于有时间试一下新的清歌五笔输入法了。这是一款 Mac 上的五笔输入法，好像刚发布不久。我知道，现在的五笔用户很少，用五笔又用 Mac 的更少，用五笔又用 Mac 又用 FreeSWITCH 的就更少了。但无论如何，我就是这么一个用户，如果早发现这个输入法的话，我写书的输入速度至少能提升 10%（主要是以前的输入法中英文混合输入太差了）。如果有跟我同类的用户，不妨看一下<http://qingg.im>。当然，即使不使用五笔，关注一下也是不错的。这年头，真正好好做东西的人少了。要不然想一想，那么多做输入法的大公司，为什么这么个小小的五笔就做不好呢？在简单试用了该输入法以后，我毅然决定捐赠一把。捐赠这个文化吧，在国内看起来流传还不是很广。FreeSWITCH-CN 开办以来，整整 4 年了，到现在只收到一笔捐赠，是资助我写书的。当然，既然话说到这里，就公布一下我的收款主页吧，地址是<https://me.alipay.com/freeswitch>（已失效），也可以点击本文左下角的阅读全文进入。如果 FreeSWITCH-CN 对你有用，不妨捐赠一下。在收到下一笔捐赠的时候，我们也搞个贡献者名单，至少，让捐赠者也感到一点点光荣。

好了，下面说点技术问题。今天谈一谈前天在 QQ 群中有网友问到的一个问题，说他的 Windows 版本的 FreeSWITCH 启动时出现如下错误，问是怎么回事：

```
[ERR] switch_nat.c:201 Error checking for PMP [general error]
[ERR] switch_nat.c:365 Unable to initialize NAT thread
[ERR] sofia.c:2573 Error Creating SIP UA for profile: external
```

其中，前两个错误是 Nat 错误，表示你的路由器不支持 NAT-PMP 或 UPNP，而这两个协议会帮助 FreeSWITCH 自动设置 NAT。关于 NAT 以后有时间了我们慢慢讲，所以，如果现在你遇到这个问题，或者看看你的路由器是否有这两个选项（或之一），或者直接使用 freeswitch -nonat 启动以防止出现这个错误。其实你在启动时不加-nonat 参数也是无害的，只是启动速度稍微慢一点。

最后一个错误是说明 FreeSWITCH 无法启动 external 这个 Profile，错误的原因可能有很多，最可能的是这个 Profile 配置的端口已经被某个程序占用了。在我又进一步询问的时候，该网友并没有回答。因此，接下来的问题就无从判断了。这里顺便说明一下，在 QQ 群中提问时，最好不要走开，或者走开前说一声，否则别人热心的帮助也没有结果。当然，这里并不单纯针对某个网友而言，希望大家都这么做，必然能更好地得到帮助。

这篇文章就是用清歌五笔打的，非常爽。再次向清歌的作者致谢。

9.3 说说 NAT

2014-01-10/Seven Du/FreeSWITCH 中文社区

昨天没有发，今天发早一点。

这两天我一直在做视频转码，试了 ffmpeg、libpx 以及 Cisco 新开放的 OpenH264。尤其是最后的 OpenH264，文档很少，刚刚开源，也找不到什么参考资料，代码嘛，写得也不怎么清爽，还是 C++ 的。因此，这几天我的 FreeSWITCH 应该是 Crash 了不下 100 次，因此，便没有时间写微信了。

视频转码有无数的参数，因此，是个任重道远的活，所以，这里就先不说了。今天说一说 QQ 群中网友提到的一个问题，比较典型。

首先是一个网友私下问我问题，我让他到群里问，后来我看到群里有人问一个在 NAT 环境下电话不通的问题（不知道是不是跟私聊的一个人，私聊跟群里对不上号，呵呵）。问题是，如果不开视频，则经过 NAT 设备的通话是通的，一开视频就不通了。看起来很奇怪，原因也可能有很多，因此，我提议他打开 SIP Trace（使用 `sofia global siptrace on` 命令）将抓包的数据放到 Pastebin 上。接着他问什么是 Pastebin，我说先看新手指南，接着他问什么是新手指南，我说那得看。是的，**我建议所有提问题的人都先看新手指南**。

说到这里，群里还有个人说他也有同样的问题。这就是**在群里问问题比私下问的好处，你能找到与你有同样问题的人，他们会跟你分享经验**。

当日志贴到 Pastebin 以后，我看了一下，客户端发了 INVITE 以后，FreeSWITCH 回了 407 要求认证，这时候客户端回了 ACK，然后应该重新发带认证的 INVITE。结果 FreeSWITCH 等了半天没有收到，因此报 WRONG_CALL_STATE 错误，呼叫失败。(有对 SIP 呼叫流程不清楚的读者可以看《FreeSWITCH: VoIP 实战》第四章，点击左下角的查看原文查看)

这里说一下，有了这个日志我们马上就定位到问题了，所以，**贴日志很重要**。

出现这个问题的原因可能是客户端根本没回下一个 INVITE (这不大可能)，或者是路由器或 NAT 设备将该 INVITE 包拦截或丢掉了。

由于现象是音频电话通，视频电话不通。而这两种电话的区别一般是 INVITE 包中的 SDP 不同，后者比较大一些。从收到的第一个 INVITE 包来看，大小已有 1265 字节，极有可能是后续的 INVITE 包更大而超过了 MTU，被路由器分包或导致了其它问题导致 FreeSWITCH 收不到完整的 INVITE 包。日志中的内容如下：

```
recv 1265 bytes from udp/[10.0.10.1]:62468 at 14:24:47.759160:
```

因此我让他换 TCP 试试，TCP 是面向连接的协议，具有丢包重传机制，因而能保证 IP 包的完整。但他试了以后问题依旧。

后来，我忙别的事情，再回来看时群里已经有朋友帮他解决了，说是路由器有 ALG (Application Layer Gateway)，他改了 SIP 服务端口就好了。ALG 是一个看起来很美好但到处都是 Bug 的 NAT 解决方案，因此在使用 FreeSWITCH 的时候，我们都建议关掉它。不过，不知道该方案中的 ALG 为什么只对视频请求有问题，音频却没问题。总之，根本原因还不知道。但这里我要说的不是这个问题。而是——**有问题在群里问，我不在的时候别人也能帮你**。

关于 NAT 我们先说到这里，后面有空我会详细说 (可能得写一个系列)。最后，贴上 Anthony 在《FreeSWITCH1.2》中说的关于 NAT 的部分，英文好的读者可以读读：

To be honest, the original stance of the FreeSWITCH developers on the NAT issue was, “Not our problem!” In an ideal world, every device behind a NAT firewall is well aware of its circumstances and can successfully solve its own problems. Unfortunately, we do not live in an ideal world (of course if we were in an ideal world, none of us would have a job because there would be no problems to solve). So we decided, “OK fine, we’ll give it a shot!” We soon learned that our users had a myriad of devices that they wanted to use with FreeSWITCH, but these devices had absolutely no idea how to deal with NAT. Soon, we began the monumental task of developing techniques to allow these devices to work despite their shortcomings. NAT is a vicious opponent and the faint-hearted do not stand a chance to survive.

9.4 要有光

今天，在QQ群里提问的同学终于有两个把Log贴到Pastebin上了，我帮忙看了下，没有解决，但当我回去再问的时候，人已经走了。

顺便说一声，如果在问问题的时候呢，时间长了没人回答可以走开，但是，这刚刚问完了就再也沒反应了是怎么回事呢？如果真有急事要走开，打上个**brb**(Be Right Back)也来得及吧？

另外还有一个同学比较强，上来直接问了个问题就是“freeswitch 怎么安装”，这个问题不难，可是，但没法有针对性地回答，我只好给了他一个链接，上面有安装的方法。可是得到的回应是：“我要完整的软件安装教程”。好大的口气，这让我想起了创世纪——神说，要有光，便有了光。

不过，即使如此，黑猫警长还是给了他另外两个链接，但是，跟上面的一样，提问的人就再也没有下文了。

还是那句话，我们在群里帮助大家，不在乎你是否说声谢谢。但是，至少告诉我们，我们的回答对你有帮助没有，这些，我们才能持续进步，帮助后面的同学。

今天的技术话题聊聊并发吧。看大家今天在群里讨论的挺热闹的。

FreeSWITCH 单机到底能支持多少并发，这个，没有准确的值，但据说在 ClueCon 2009 年就支持 10001 个并发了（参见），而且，在 2011 年 ClueCon 上，Anthony 演示了 1000cps，3 万个并发呼叫。（到 Google 上搜“Tony demonstrated FreeSWITCH running 1000cps 30sec call duration with media for a total of 30K concurrent calls @ ClueCon last year (just a few weeks shy of a year ago)”。当时服务器的 RTP 端口都用完了，因而不得不在一台机器上绑定多个 IP。当然，我本人从来没达到那么大的呼叫量，在我的测试中，最大的还是在一个 4 核 CPU 4 内存的机器上跑的 3000 路并发（跟华为 SoftX3000 对接），以及在某一国外咨询项目中在另外一个更强一点的服务器上跑过 5000 并发测试。

当然，只讨论这个数值没有什么意义，重要的还是在真实的业务场景中能达到多大的并发。每个人能达到的并发数因人而异。FreeSWITCH 官方一直对性能测试不是很感冒，我也是。做性能测试需要花很大的精力去调优，出了问题查起日志来也异常麻烦。因此，希望大家多把时间学习基本功能上，把 FreeSWITCH 用好再说。说实话，真有这么大的并发量了相信你已经能挣很多钱了，到时候再优化也不迟。

基于这些原因，一般遇到如 SIPP 怎么用，WinSIP 怎么用之类的问题我也不怎么乐于回答。

不过，我也不是阻止你自己去测，有条件的话学习一下也没什么错。这里，说两个重要的参数：一个是 SPS，又叫 CPS，即 Call Per Second，也就是一秒钟发起或接收多少个呼叫；另一个叫 Max Sessions，即最大并发数。这两个参数是相关的，它们又跟每通电话的呼叫时长有关，公式是并发呼叫数 = sps * 呼叫时长，如果所有电话都持续 1 分钟，sps = 100，则，并发呼叫数就是 100 * 60 = 6000。好像业界的 PBX 设备能支持 50 个 cps 就算不错了。

这两个参数默认的值分别是 30 和 1000，可以用以下命令显示这两个参数：

```
freeswitch> fsctl sps
freeswitch> fsctl max_sessions
```

也可以动态地调整

```
freeswitch> fsctl sps 500  
freeswitch> fsctl max_sessions 10000
```

当然，也可以配置到配置文件中，以便在下次 FreeSWITCH 重启时也能生效。在此，我就不说这两个参数的具体名称了，到 `switch.conf.xml` 中找一个 30 或 1000 应该就能找到了。

需要注意，这两个参数是为了保护你的服务器的，因此，不要设得太大。

9.5 如何得到我的帮助

2014-01-03/Seven Du/FreeSWITCH 中文社区

今天，看到粉丝数量已经超过 100 了，有一点小小成就感了。在昨天的文章中，放了一个小小的彩蛋，好像大家都没有看到，一个留言的都没有。今天，重新放一个：有什么问题可以向我提问，我会挑一些有代表性地在后续的微信文章里给大家回答。提问有效期一天。

既然没收到留言，今天也不谈技术问题了。还是谈一谈社区的问题。今天，在 QQ 群里又有人私下问我问题了。说实话，这给我的第一印象就很不好。更不要说，连自我介绍都没有，直接上来就问，显得很不礼貌。一般来说，在 QQ 群里问问题，会比较好些，别人看见了也可以帮助你。如果实在是想直接跟我私聊的话，至少来一句：“您好，冒昧打扰，我是 XXX，来自 XXX，我想学习 FS，听说您在这方面懂得比较多，能不能问一个问题？”说实话，我不是个喜欢听恭维话的人，但如果我听到这样的话，至少让我知道你理解我平常很忙，让我大体了解你的谁，你学习 FS 多久了，我才能知道该怎么回答你。

另外，我在 QQ 群里回答问题时，总有人对我的回答根本不听。我不知道是我表达有问题，还是我的思路不对。这里说明一下，提问时，我希望提问者是以一个小学生的态度问问题，因此，应该看 FreeSWITCH 新手指南。我花了很长时间写了新手指南，如果他看了再提问题，是节约他自己的时间，同时也节约我或别人的时间。

我在 QQ 群里一再主张大家把日志贴到 Pastebin 上或者 Gist 上，或者任何可以贴代码的地方。因为，好多人提问时根本不知道他想问什么，也描述不清楚问题，但有时候，我光看 Log 就能看出问题所在，因此，把 Log 贴出来，把链接贴到 QQ 群里，我很快就能帮到他。但是，有时我说了 3 遍，好像问问题的人根本就没看见！

这里再强调一下为什么要贴到 Pastebin 上。因为，Log 可以有彩色显示，比较容易看。我非常讨厌在 QQ 群中屏幕截图，看不清。大段的文字也不适合直接贴到 QQ 里，因此，Pastebin 是最好的选择。

好了，多了就不说了，下面是我在知乎上的回答，一块贴上，供自己参考。读者可以通过点击左下角的阅读原文在知乎上看该问题。

——华丽的分隔线——

关于 FreeSWITCH 的学习，我很想帮助大家，但大家却好像不知道怎么得到我的帮助。

我于 2009 年创办了 FreeSWITCH-CN 中文社区，非常喜欢跟大家一起学习和提高，交流经验。对于新手而言，我也非常喜欢帮助大家。但是，大家也应该考虑到，我工作比较忙，因此，在向我提问时最好把问题准备的完善一些。以便得到快速的解答，节约你的时间同时也节约我的时间。

我最喜欢的方式是在 FreeSWITCH-CN Google Groups 里向我提问，但是很遗憾，大部分用户都无法加入该邮件列表，因此，好几年了，没有什么人气。

不过，现在有了知乎了，你可以在知乎上提问，并邀请我回答。

最后一种才考虑在 QQ 群(190435825) 中向我提问。提问时最好把 FreeSWITCH 版本号，操作系统，以及问题的现象等都说清楚，如果我要看 Log，那么，在 FreeSWITCH 控制台上多打几个回车，然后打电话，把 Log 内容粘贴到 Pastebin 上。注意不要直接给我发 Log 文件。如果不知道 Pastebin 怎么访问，先看一下 FreeSWITCH 新手指南，在我们的官方网站上可以找到。

如果我要看 SIP Trace，就在控制台上打上下列命令开启 SIP Trace，并连日志一起贴到 Pastebin 上：

```
sofia global siptrace on
```

另外注意一点，如果是在 QQ 群中提问，尽量不要贴图，图很难看清，而且在移动设备上很浪费流量（导致我从来不在手机上用 QQ），所以尽量贴文件的 Log，如果 Log 比较长，就贴到 Pastebin 上。

当然，除了问我以外，QQ 群中也有很多其它的高手，礼貌的问问他们，一般会有人帮你解决（我相信他们也喜欢用我的方式帮你解决问题）。另外，如果没有回答你的话，也不要重复灌水，那样起不到积极的效果。还有，在 QQ 群中提问时，如果不是特别熟悉的话，最好不要直接上来就私聊。原因有三：

- 你丧失了 QQ 群中别人回答你的机会；
- 我不一定在，在也不一定有时间；
- 有些人连招呼都不打，上来直接就要求我回答一个问题，很无礼。

最后，声明一下，我并不是什么都会，会的也并不一定有时间回答。但是，如果我恰好会，又不会占用我很长时间，我很乐意回答你。

最后，不管是我的回答有没有帮助你解决问题，回头把结果告诉我一声（说声谢谢也好，不是必须的），好让我知道我说的对还是不对，以便帮助下一个有问题的人。

最最最后，为了能更有效地找到你想要的答案，磨刀不误砍柴功，推荐看一下下面几篇文章：

- 如何在知乎提一个好问题?
- 如何提问: <http://www.freeswitch.org.cn/2012/10/19/ru-he-ti-wen.html>
- 提问的智慧: <http://www.wapm.cn/smart-questions/smart-questions-zh.html>
- 橡皮鸭子问题: <http://www.freeswitch.org.cn/2012/09/21/jie-jue-xiang-pi-ya-zi-wen-ti.html>

9.6 无题

2014-02-13/Seven Du/FreeSWITCH 中文社区

从这几天微信的后台统计信息看，好像讲模块的话看的人就少，乱讲一气的看的人就多一些。今天换换口味，再验证一下。如果大家不喜欢讲模块呢，我少讲点也行。

年前，有个公司跟我联系说让试用一下他们的 IP 话机，我说，也好，我试用一下好的话没准也可以给大家推荐。不管是软广告还是硬广告，一般来说，推荐我亲自实验过的产品，心里踏实点。

后来联系了一下，对方同意提供试用。我看着产品外观还不错，问能否直接赠送两台话机，我平常测试也好用。而且，我也不是白拿，我可以在微信上给他做介绍嘛，甚至可以写个公开的博客嘛，而且，我看到他们产品兼容性列表里没有 FreeSWITCH，如果把 FreeSWITCH 列上去，还算是不错的事情吧？互惠互利的事情，不知道他们是否愿意干。

可是，对方说公司没有这样的规定，只好做罢。不送就不送吧，测测也行。但是，在签试用合同时还要求我付全款押金，我只好跟他说不测了。不是我付不起这个押金，我只是嫌麻烦，一来二去耽误我时间，送几台话机都换不回来。

说到这里顺便做个调查，如果我用某个厂家的设备写个与 FreeSWITCH 对接的技术贴或者软文，或者有时候写的不软被大家看出来的话，大家会不会烦感？如果没有意见或意见不大的话，我决定在合适的时候尝试开通软文广告，先让这些设备厂商支援支援 FreeSWITCH 社区。

当然，如果你正好是设备厂商，或者你有什么产品愿意赞助我搞搞活动的，也不妨联系我，我或许可以搞搞抽奖类的给大家发发福利。

今天在 QQ 群里听到大家聊性能的问题，有人对 FreeSWITCH 支持 2000 并发的集群赞叹不已，有人说他的 Asterisk 好几台机器才到 600 并发。当然，使用场景和业务逻辑不一样，单纯比并发也没太大意义。不过，如果你有 FreeSWITCH 的成功经验，愿意与我分享的话，我还是很愿意听的。

9.7 说说开源软件

2014-02-09/Seven Du/FreeSWITCH 中文社区

由于过年放假，疏懒了一阵。看到大家的留言，还真有些读者每天都等着看我的文章。最近脑袋还没回到工作状态，因此思路有些乱，要不，还是从明天起接着讲模块吧，今天，先随便说点别的。

年前，做了个小小调查，让大家跟我分享一下自己的 FreeSWITCH 故事。倒真有几个朋友分享了。另外有个朋友留言说，能不能把所有分享跟大家都再分享一下，如果那样的话整个微信公众账号就互动起来了。

这里，我想说一下，如果大家反馈比较多，故事也比较有代表性，我可以不时的总结一下。但让粉丝之间互动却不是微信的初衷。如果粉丝之间要互动，那大家都到 QQ 群或发微博好了，我们也有 FreeSWITCH-CN 新浪微博账号。

当然，也不能说粉丝互动在微信上是不可能的，微信好像出了一个微社区，只是，好像没有公开测试。如果开放了，我一定试试。

前几天，在知乎上看了一个关于中国开源软件现状的问题，我忍不住回答了一下，今天就不贴具体内容了，如果感兴趣的话请直接点击「查看原文」阅读。

<http://www.zhihu.com/question/21965679/answer/22151518>。

9.8 穿越

2014-02-25/Seven Du/FreeSWITCH 中文社区

知乎上有个问题，内容是：如果有机会让现在的你遇到刚毕业初入职场的你，你会对他说什么？

噢，如果真的能穿越到从前遇到从前的我就好了。但不管能不能，我当时的回答是——找个有钱的另一半。

不过，那只是占位用的，下面是修改稿。与大家分享一下。提前说一句，那个问题是值得一看的，尤其是对刚刚毕业的同学们来说。感兴趣的同學可以点击左下角的「阅读原文」查看，当然也包括很多高手的回答。

修改：

上面的答案是以前调侃用的，主要是为了占位，没想到还被点了赞。呵呵，缘分这事不好说，我们还是说说工作的事吧。关于这个话题，大家分析的都很到位很全面了，我也感觉插不上嘴了。简单说说我的经历吧。

毕业后顺利进入国企，一呆就是 7 年。期间，N 次都想离开，但一直没有如愿。当时打错了算盘，想先成家后立业，但由于一直没有成家，因此也一直没有立业。后来，成家了，却失去了立业应有的天不怕地不怕的精神和精力。所以，我想说的是：想好了就去做，不要在乎别人怎么说。

当然，在原单位我还是挺努力的，工作之余，我自己也学编程，我开发的一些程序到现在还在原单位跑着呢。所以，**即不走，则安之，工作要努力，学习自己感兴趣的东西。**

后来，去北京进了外企。那里环境不错，同事朋友们也都很 Nice，我还努力创造了一次去美国的机会，虽然是自费。说到外企，**建议大家有机会的话还是去干几年**。不一定成为终生的职业，但是，那里的氛围和思路以及你能接触到的人跟国企还是不一样的。

说实话，从小地方一到北京，才发现 7 年攒下来的钱，根本买不起房子。也没法把全家接到北京。本想着再攒攒再说吧，可是，却发现，又干了 N 年，却离房子越来越远，永远也赶不上了。这里，没什么想说的，如果说有的话，**不建议在京买房**。

不过，留在北京也不算什么坏事。后来，我逃回烟台，现在，虽说烟台也有雾霾，但空气指数与北京比起来，跟北京的房价是成反比的。想说什么呢，**不一定挤在大城市**。

我创业了。公司在北京，人在烟台。做的不好，但也不坏，唯一的就是很累，占用了许多本应该陪老婆孩子的时间。想说什么呢，**不一定非要创业**。

这些年，我一直坚持做开源的 FreeSWITCH，并写了一本书，估计很快要上市了。不管它将给我带来什么，这都是我这么多年经验的积累和总结。使我感觉到我过得充实。想说什么呢？**做自己喜欢的事情，同时也要坚持做好至少一件事**。

这些年来，我待过的公司也好，处过的同事也好，都没有自私自利勾心斗角的。我也把我会的都毫无保留地与同事们分享和交流。大家共同学习和成长，是我愿意做的。一路走来，感觉还不错。**希望大家都跟我一样，多与大家交流和分享**。

不多说了，比较乱，仓促间写下这些，希望对原来初入职场的我有所帮助。

9.9 屁大点事

2014-03-22/Seven Du/FreeSWITCH 中文社区

很久以前就想写这篇文章，只是一直没倒出时间。

事情的起源是这样的，在 FreeSWITCH QQ 群中有位网友有个 FreeSWITCH 的需求，正好我能做，便留了一个邮件地址让他联系。没想到，他却说“屁大点事”还用发邮件，不能 QQ 或电话直接联系吗？

说实话，如果真是屁大点事，那么不值得我们做。但我没想到在全民互联网时代电子邮件竟然成了一个障碍。

诚然，并不一定所有人都习惯用电子邮件。但我却是一个电子邮件的重度用户。每天，我都会收到上百封的邮件，虽然这里面大部分都是订阅的一些邮件列表，我也不是每封都看。

事实上，我们公司的网站上也并没有留任何电话及 QQ 之类的即时通信信息。在强调顾客就是上帝，强调用户体验的年代，为什么我还这么依赖电子邮件呢？

其实这件事的初衷就是我们不希望把时间花在整天接各种骚扰电话上（有恶意的，也有不经意的），当然也不想在 QQ 上收到任何乱七八糟的询问信息。如果一个人连发个邮件这么“屁大点事”都不愿意做的话，他多半没有跟我们合作的诚意，我又为何在这种人身上浪费时间呢？

当年我做在线英语教学的时候，我们的英语口语教员都是在美国招聘的，而我们利用我们的语音技术，做了自动面试的平台，应聘者需要填一系列的表格，填完后还需要经过我们好几个步骤的自动

电话面试（当然，那些自动电话面试的程序都是用 FreeSWITCH 做的），自动面试通过以后才会有专人进行远程电话面试。

当时我也向领导质疑过——如此复杂的步骤不会将很多优秀教员都挡在门外吗？他说，申请我们教员的有很多，事实上，最后能被我们录用的比率很小，也就是说对成功率而言能进入我们公司当教员比进入哈佛大学还难。在这种情况下，我们必须通过一些手段过滤掉一些人。而这种复杂的注册和面试流程恰恰能过滤掉一部分人，把机会留给那些真正有耐心和实力完成这步的人。

我恍然大悟——其实，大部分申请者说英语都很标准，我们恰恰需要那些有耐心的人。

无独有偶，其实 FreeSWITCH 的作者 Anthony Minessale 也干过同样的事。他在他的《FreeSWITCH 1.2》这本书里是这样说的：所以我就不停地写代码。时间很快到了 2006 年 1 月，那时我有足够的代码可以与公众分享了。我们向开发者们开放了我们的代码库。通过让他们注册一个开发者账号才能获取代码的访问权限，我们确保只有非常严肃的开发者才愿意完成整个注册过程… 原文如下：

I just kept plowing through the code and by the time I reached January of 2006, I had enough to share with the public. We opened up our code repository to developers asking them to register for a developer account to gain access to the code as a way to make sure only those who were serious would bother to complete the registration process…

9.10 那些故事

2014-03-26/Seven Du/FreeSWITCH 中文社区

过了年，我的书正式定稿后，我确实清闲了一阵。不过，一年之计在于春啊，因此，我最近又忙起来了。

但不管怎么忙，我还是回间或在 QQ 群中回答一下大家的问题。

非常欣慰的是，QQ 群里现在也有很多网友在回答大家的问题了，似乎，春天来了，我们的 QQ 群也热闹起来了。

但我每天回答问题的时间是有限的。因此，我一般是回答新手学习性的问题，至于涉及的商业运营、稳定性等问题我一般不回答。一是由于我的时间有限，我公益性的时间大多数只能留给刚开始学真正需要帮助的同学；二是，大多数比较高级的问题并不是一两句就可以回答的，需要对应用场景分析，看日志，甚至进行调试等，我自然没有这么多时间；三是，既然搭好了 FreeSWITCH 环境来挣钱，就不能留点预算用于商业咨询和支持吗？

由于我能抽出来的时间有限，所以我定了一些规则，比方新手在提问之前先好好看看新手指南啊，按我说的去做啊之类的，这样能节约你的时间，也节约我的时间，况且，你同时学会了怎么在社区里提问并快速得到答案不是？这个话题以前讲过了，对此有好奇心的同学可以查看“我在学习 FreeSWITCH，遇到问题时，如何得到 Seven Du 的帮助？”一文。

<http://www.zhihu.com/question/22412945>

说到花钱找支持，有的人也许会说，我也是刚刚学了 FreeSWITCH，搭好了还没挣钱呢。诚然，挣钱不是那么容易的，这就像我做 FreeSWITCH，挣钱还很难一样。但是，你或许可以换一种思路想一下——1) 我自己折腾，什么时候搞定了什么时候挣钱 2) 我花点钱找个人搞定，早弄好了早挣钱。想清楚了之后，再怎么决定就在于你自己了。

说到挣钱，确实，我有一部分收入是靠商业的咨询挣钱的，但那不是我主要的经济来源。我也做点培训，同样，去了成本也就够个吆喝钱。那么，我为什么还要做呢？

1) 我喜欢开源，喜欢 FreeSWITCH，既然我现在在 FreeSWITCH 方面在国内还有一片阵地，我希望把它做大、做好；

2) 几年前，我跟他们一样，对 FreeSWITCH 一无所知，是靠 FreeSWITCH 社区和 Wiki 的帮助，当然，还加上我自己的努力，我现在才对 FreeSWITCH 略知一二。所以，如果你也像我一样，我希望能指导你少走些弯路；

3) 都说国内开源难做，都说服务不值钱。我还是想拿出一部分精力来试一试。毕竟，我们这几年还是靠开源收入了一些钱；

4) 如果你自己折腾 FreeSWITCH，又想快速搞定，那么，我们提供一个良好的商业支持，帮你搞定。省了你的时间，同时建立良好的合作关系，同时我们也有一点点收入，保证我们能活着继续为你服务；

5) 如果你们公司基于 FreeSWITCH 做业务，又非常需要技术支持，那么，我们提供一个强有力的支持，包括个性化的修改和增强，成功了，你挣大钱我们挣小钱，不成功，你花点钱早点知道再改行做别的，不是更好吗？

总之，我们的定位很明确——我们不是暗示你或说服你给我点钱让我帮你服务，我们是勤练内功，让那些想花钱找人帮忙搞定 FreeSWITCH 的人或公司能找得到我们。

在这个宗旨下，我们要做的就是，普及 FreeSWITCH 知识，让更多人学会和使用 FreeSWITCH，这样就扩大了 FreeSWITCH 的使用群体，同时能增加“想花钱帮忙搞定 FreeSWITCH 的人或公司”的数量。所以，如果你是初学者或者新手，你就赚了。我们在 QQ 群里回答问题或者我们在网上公开发表的资料都是免费的。

培训呢，也是这个目的——我们想让那些愿意花钱参加 FreeSWITCH 培训的人能找到培训班，而不是我坑蒙拐骗让你来参加培训，培训完了还找不到工作。我昨天也在微信里这么写道：

无论怎样，我们不能改变我们生活的环境，但多学一些总是好的，知识掌握在手里永远是自己的。

今天也有人在跟我讨论培训收费贵不贵的问题。我觉得，怎么说呢，办一场活动是不容易的，我已尽力到靠你最“近”的地方，并联合了上海软件中心给学员提供更好的学习环境。靠培训吃饭的话我目前还不敢指望。因此，机会很难得。不管我水平怎么样，我愿意尽我所能，把我会的都告诉你，至于学的怎么样，就得看你自己了。

所以呢，其实培训劳民伤财连准备带培训得一个月时间，我们真不挣钱。因此，我也说我们的培训班机会难得也是说我们的培训班是有限的，不是你想有就有。

当然，在我刚刚自给自足之时，又有高人来指点我了。我不能光自己能吃饱就行，还得想想所有这些学习 FreeSWITCH 的人怎么吃上饭，还得想想所有那些使用 FreeSWITCH 挣钱的人怎么挣到钱。只有大家都挣钱了，我才能挣更多的钱。

是的，我承认，我一直是做技术的，我一直想做的也就是如何把技术做好，没有那么高瞻远瞩地去想整个生态链的事情。不过，我想，至少现在，所有在做 FreeSWITCH 的人都应该比我想得多看得远了，各种前景和风险也比我看得全，我还得向他们学习。

当然，我确实也在想这个问题。

————华丽的分隔线————

其实，今天的内容本来是起源于 QQ 群里的一则故事。因而引发的一些我的思考，但后来呢故事又颇有戏剧性，所以，今天就准备写长点，再讲几则故事。

还是从这则引发血案的故事讲起吧。今天在 QQ 群里有个网友问问题。其实呢，这个网友说起来有些熟了，因为他这几天一直在问问题，前几天我也帮过他一点，他后来把问题搞定了，还把他的配置晒出来让大家共同学习，因此，对他还算颇有好感。

但他今天问的问题有些复杂，虽然这已超过“自己学习”的范围，我还是准备免费帮助他。但确实我需要看到他的 Log 才能更好的定位问题，因为他已经不是默认安装的 FreeSWITCH 配置，自己做过好多修改，谁知道他改了什么，谁知道他描述的问题跟真实的问题是否一致呢。因此，我让他把 Log 贴到 Pastebin 上。

至于为什么贴到 Pastebin 上呢，原因就是 Log 很长，不能很 QQ 群里猛灌啊。另外，FreeSWITCH 提供的 Pastebin 能对 Log 格式化，不同级别的 Log 用不同的颜色显示，因而看起来比较顺眼。但是我说了三遍，他还是没看新手指南，不贴 Log，后来，我就说，我今天不能帮助你了，因为，时间到了。

期间他也问到能否直接把日志发给我，我说可以，但直接往我邮箱里发邮件咨询是要收费的，在 QQ 群里以及 Pastebin 上支持则不收费。后来，他选择了不交费，自己去看新手指南。

为什么我用这种策略呢？因为，私人咨询相当于一对一支持，理应收费的。而且，我帮了你，别人也学不到东西，所以，我花了时间等于私人教练，没起到更大的本应有的价值。因此，收费只是一个借口，我更希望大家在公共的 QQ 群里交流，大家都多学知识。

当然，他选择了不交费，而看新手指南，这让我感到，其它新手指南也是值钱的，以后是不是想个办法“卖”新手指南？

呵呵，回归主题。我们不是说了该故事有戏剧性吗？主要是 @ 昆山出来调侃了几句，引出来好多潜水者。话题也更热闹了。我就怀疑，为什么我在群里一说话大家就不做声了呢？难道 @ 昆山比我魅力大嘛？

长话短说，后来请求帮助的这位哥们居然说愿意出 500 块钱让昆山帮他搞定这事，昆让则让他把钱捐给我。

话说，昆山是个好同志，过年那时候就带头捐钱。他从他去年在 FreeSWITCH-CN 沙龙上的演讲到他今天在阿里云上的成功故事，都说明他是真的干出来的，也是踩着 FreeSWITCH 成功的，因此，他是我们群里极好的一个典范。

不过，由于我有言在先，今晚不能帮助他了，而且，我也确实没时间，忙着写微信的文章了，因此，这件事情只好留到明天再解决。

但他说他用 Ubuntu 使用 WebQQ 跟我们聊天，又让我想起年轻时候的我。一时想起好多旧事，因而又想写一篇文章，不过，今天肯定来不及了，记在这里，做为后话。

好吧，这一则故事就讲到这里了，下面，再讲一则。

某年月日，也有个来咨询的，一来二去，最后确定了需求，我们团队的小弟帮他搞定了，我们告诉他账号，话音未落就收入 500 大元，几个兄弟午餐每天都加肉，如果称一称，说不定每个人都胖了好几斤呢。

我是山东人，喜欢直来直去、爽快的人。所以，这种人，我喜欢。

当然，也有不怎么地的人，说的就是下面这一位：

有一次有个家伙在邮件列表中问问题，我回答他一下，接着就像中了风一样，每次问我都发到我的私人邮件。我一次次地很礼貌地回答他，并告诉他，这些东西下次到公共的邮件列表中去问，我回答都是免费的云云，可人家就是不听。也是我那阵比较闲，因此呢，每次我都回他，并告诉他如果再往私人邮件发的话就要收费了，他还是一再的发，也不提交钱的事。后来，我就真不理他了，但他死心不改，后来提到费用，我说了一个价格，他砍去一半。一半就一半吧，我帮你解决了，省得老烦我，一来二去讲价花的时间比我解决问题的时间都多，那我干得也太不值了。

结果，又是戏剧性的一幕。我后来详细问了他一些问题，他回答之后，我只一句话就把他的问题解决了。但这时候，他不是高兴，反而说我这一句话不值那些钱了！你说有没有天理，我免费支持的时候，无数次写邮件让他告诉我这个告诉我那个，问一个告诉一个，总之就是没把关键的问题告诉我。早管着干什么了，现在收钱了，解决了，又说一句话不值那么多钱了。

这个我还得好好分析一下：

1) 他的意见是，我以前数十封邮件的时间不算时间，时间应该从定了价格开始算，而我一句话不值那么多钱

2) 当时我也是太年轻，早知道我就应该（装做）研究两天以后再告诉他！

所以，这人的脑子是极有问题的，我知道答案，立即告诉他，想让他早点解决问题，他却不领情。另外，这人品德是极有问题的，我到现在也没收到这笔钱。

常在水边走，哪有不湿鞋的时候。这样的人，我不理就是。

再说另一则故事吧，这回尽量讲短点：

某年月日与人约了个三天的培训（远程），讲了两天之后，后来一直没联系我。也是我当时疏忽，没记下他电话，后来费了好大劲才查到他电话，打十几个都不接，或挂机，后来总算联系上，原来公司倒闭了，钱呢，没了（当然，我不知道这是真的假的）。

真没想到还有这样的人。如果想赖账，第一个电话接起来说不付就完了，跟我玩失联，什么意思呢？

当然诸位也可以说是我的错，当初为什么不先收费呢？为什么不走公司流程呢？是啊，是谁求着我不要走公司流程，按个人收费的？当初是怎么跟我说怎么这个还能信不过嘛之类的话的？总之，我都懒得跟他叽歪。直接拉黑省心。

话说呢，我甚至没问他是什么公司的，是干什么的，小时候武侠小说看多了，拿人钱财，替人消灾，总之呢，最后我落下个不是，为人不查，谁也不赖，赖自己。

这说着说着又说到另一则故事，话说我这故事还挺多的啊。

去年，一个朋友介绍一个活，要搞定 OpenSIPS 和 FreeSWITCH。当时是周末，我就说等周一再搞吧，也不能周末签合同啊。这不，说是着急，又有中间人之类的，我心就软了。我当时说的也不明确，我说，现场环境不熟，我不确定能否搞定，如果当晚搞不定呢，我就不搞了，让他们另请高明。实际是，我下午就搞定了全部的问题，但由于他们的网络设备有问题，业务还不正常。但我也给他们定位出了问题，只是问题的真正原因不知道。这不是到了第二天了吗，又帮他们查，最后查出真正原因是网络设备的原因。妈的（我可以骂人吗？）网络原因管我什么事，我要解决的问题是在网络设备好的情况下解决 OpenSIPS 和 FreeSWITCH 的问题。而且前一天晚上我已明确告诉他们是网络设备的问题要找网络的人来解决啊。况且，如果真是加上查网络设备的问题的话也不是这么个价啊！

还好，由于有中间人，最后付了一个百分比。

从这三件事之后，我痛下决定，2 万块钱以下的私人项目，预付款就必须是 100% 才做。否则，走公司流程。否则，免谈。

话说呢，最后我还发现该公司刻意隐瞒了一些信息利用了我，丑事就不跟大家说了，呵呵。

当然，世界还是美好的，如果所有人都这么流氓我也不愿意讲故事了。

去年做了个印度的项目，最开始付款也没及时，但后来款到之后我发现比约定的多了好多美元，经查证后是对方觉得我做得好，并且由于当时付款问题延误（其实也就延误几天，与国内公司动辄延误一年相比那真不就延误），就多给了一点。世界上还是好人多啊，你多给 1 美元我也开心不是？

国外的也有私人项目，请我远程讲课，我说了个价钱二话没说 Paypal 上就付了，而且是每次上课前都付，讲起来也觉得爽，我就给他多讲了些，后来导致他预计一周的课程我三天就把他教会了，你说是谁赚了还是赔了？

去年做过一个欧洲的项目，先预付 1 小时的钱我给他做需求分析，在分析中我说了 N 点，交换了几次邮件，每次得到的答复都是 Perfect、Exactly want we want。后来做项目说好了分 4 次付款，每次都时间还没到呢，款就到了，当然，他最后得到的功能也比约定的多。

好了，今天时间比较晚了，就写到这里吧，讲故事也挺花时间的。今天就不总结了，仁者见仁，自己体会吧。有什么批评建议，尽管提啊。

题外话：那些在微信上给我留言让我回答，却又关闭了消息接收的人，是怎么想的呢？

9.11 FreeSWITCH-CN 高手速成培训 2014 春季班（上海站）圆满结束

2014-04-30/Seven Du/FreeSWITCH 中文社区



FreeSWITCH-CN 高手速成培训 2014 春季班（上海站）已于 2014 年 4 月 21 日圆满结束。

从去年 6 月在北京成功举行了 FreeSWITCH 高手速成及实战课程的技术研修班以来，很多没赶上上次培训的朋友都纷纷要求开下一次培训班。本着上次的培训理念并尽量符合学员的要求，我们（FreeSWITCH-CN 及北京信悦通科技有限公司）与上海软件中心、谷声（上海）信息科技有限公司联合展开了上海站为期三天的 FreeSWITCH 培训课程。

本次参加的学员数量是去年的两倍。参加培训的学员都是来自各公司的技术骨干，他们年富力强又热爱新技术。大部分学员来自长三角地区，但也有部分学员不远万里，从遥远的重庆、哈尔滨、北京、深圳、广州、武汉等城市赶来。而且，更令人欣喜的是，我们还迎来了自 FreeSWITCH 培训开天辟地以来的第一位女学员。另外，有几个去年已经参加过培训的老学员也加入我们的培训，他们对我们课程的认可使用我们倍感温馨。上海当地的一名老学员也专程赶到现场为大家分享他上次的学习体会以及这一年的 FreeSWITCH 使用经验。

在 Grandstream 的支持下，我们增加了实际的 IP 话机的演示环节，使场面更加生动，并且由于本次的培训加入了有趣的互动抽奖项目，很好的调动了学员的积极性和创造性。

为增进学员间的相互交流，我们也专门于培训的第二天晚上为所有学员准备了晚宴。通过大“吃”大“喝”，也更拉进了大家的距离。美中不足的是，由于我们通知比较晚，一些提前做了其它安排的同学未能参加。

无法避免的是参训学员普遍反映培训时间过短，希望组织更长时间的培训，学习到更多的相关知识。FreeSWITCH 本身是一个庞大的系统平台，3 天的技术培训时间只能够起到开头引路的作用，我们希望通过培训的这样一种形式，让学员可以快速入门，并具备进一步研究与开发的能力，同时为学员和讲师们之间建立一个交流的平台，共同研究和学习。

为了满足大家对于 FreeSWITCH 学习和研究的需要，我们会不定期的举办相应的培训课程或主题研修班。请大家持续关注我们的网站。

我们通过培训的反馈问卷收集整理学员的意见和建议，同时希望大家可以将自己的需求和建议发给我们，我们会不断的改进完善培训内容，增加更为关心关注的课题和应用，为大家提供更好的实效课程。

作为培训计划的一部分，我们也将于今年 6 月 16-18 日在北京开办 FreeSWITCH 高手速成培训夏季班，提前报名可享受优惠。

9.12 锤子

2014-08-30/Seven Du/FreeSWITCH 中文社区

锤子本身跟 FreeSWITCH 的关系甚远，但是，事物总是相互联系的，至少我今天就把他们扯到一起了不是？

说锤子得先说说罗永浩，人称老罗。知道老罗是由于前些年看了他在海淀剧院的演讲，好像是关于《一个理想主义者的创业故事》(或许还是一个系列，有 1、2、3 之类，未考证)。当时打动我的主要有这么几点：1) 追求设计，一个海报都做得那么精美 2) 跟别人不一样，总是做些出人意料的事，敢于反其道而行之，如公司上下全部用正版软件，一块钱听八次课，那个关于 FUCK 的英语广告等 3) 那就是老罗说的“**通过干干净净地赚钱让人相信干干净净地赚钱是可能的、通过实现理想让人相信实现理想是可能的、通过改变世界让人相信改变世界是可能的——即使是在中国！**”。

受我前同事方舟的影响，我对界面设计尤其是交互设计产生了浓厚的兴趣，无疑，老罗的设计是我见过的设计里面最好的，也是最用心设计的。

从某种方面讲，老罗是个流氓（没有贬义），他一直在用堂堂正正的流氓手段在做事情。在看了“其实内在的一些东西是一模一样的……”之后，更加了解了这些动机的源头。是啊，谁又不一样呢？

他跟方舟子的斗争最终（好像）没有结果，但他却从做教育改行做了手机；西门子冰箱事件闹得很大，那把最有象征意义的锤子也最终成了他公司及产品的名字。但不管怎么说，锤子这个名字不是怎么响亮，至少，锤子科技有小姑娘给我打过几次电话，每次都是说了好几遍我才听出来是“锤子科技”。而前几天，他与王自如的现场对质又成了人们饭后的谈资。

我对这场辩论并没有什么看法，或者说，没有什么高深的见解。老罗还是一如既往的那么流氓（如，不允许别人打断他却总打断人家，当然大家都只是说风度不够，我认为够流氓），但他确实是有备而来，一大摞白板直接就占了上风。他在对质一开始就直接拆掉了王自如将视频重放一遍的大

招，不管谁对谁错，王自如在老罗面前还真不是对手。整个对质过程很好看，大家的评论也都很多了，我也说了我没什么高深的见解。不过我有一点小小的疑问：在王自如明知苹果配件供应属于灰色地带的情况下，怎么还那么振振有辞？我还记得从小学就学过，如果明知某件东西是小偷偷来的（赃物）而购买的话，本身就是违法的。这难道不能类推到王自如的配件吗？即使他跟供应商之间所有的交易都是合法的又能证明什么呢？

我也买了锤子手机。

我最早用的智能手机是 iPhone 3GS 阴割版，没有 wifi，就从来没有爽过。去年这个季节，3GS 实在是跑不动了，又买了一台华为的双卡双待的 Andriod 手机。当然，买这个还有一个原因，那就是电信的信号通常都比联通的好。换号的成本太大，只好同时用两个号（其实用两个号的成本也不小，除了交两个套餐费外，你还得跟别人解释你为什么有两个号，人家还得判断该拨你哪个号码……）。一年后，也就是今年 7 月份之前，华为的屏幕摔坏了，但新买的锤子还没到，只好将就着用。

去年，有幸参加了 SmartisanOS 的发布会，得到了优先购买码。本来说是 7 月初就能发货的，结果由于产能不足整个 7 月份都没拿到货。我也没法拿着去美国 ClueCon 上吹了，还是拿着碎了屏的华为去的。

好在，从美国一回来就收到了手机。看起来，非常好。当然，我没有用过 iPhone 4 和 5，无法比较。但比起我以前用过的任何手机来，都好太多了。

多等了一个多月，也许是值得的，网上有人反映的各种机械问题、漏光问题，在我的手机上都不存在，一切都很好。

我不是锤粉，也不是黑，它对我来说，只是一部手机。我用过的手机也不多，但我是做 FreeSWITCH 的，是做通信的，还是要整天跟手机打交道的。

锤子的各种好就不说了，反正，我只关注是否好用，也不是很在乎价格，总之我很喜欢这款手机。

下面说几个我用的锤子的问题。

邮件：

锤子的邮件客户端不支持 thread，我工作严重依赖邮件，看起来比较麻烦。而且，我还订阅了很多邮件列表呢。

邮件提醒不对，经常误报收到几十封几百封邮件，看时却没有新邮件。

装了原生的 Gmail，又得装 gws，又得装 Google Play，Google Play 闪退。也许不是锤子本身的问题，但，期待有解决方案。

浏览器：

支持有限的 Tab，打开多了页面时经常提示需要先关闭几个不用的 Tab，这对我经常在桌面版的 Chrome 打开几十个页面，在手机上也常常不关页面的习惯确实是一个挑战。

不支持 WebRTC。WebRTC 在今天已经是很重要的了，我还得下载 Chrome。

调试：

不知道为什么，在打开 USB 调试之后，Andriod 的开发环境（MAC 版）中认不出手机。

不知道我的算不算问题，只是，我没看到其它人提到这些问题（大家好像都在关注屏能不能摔坏，照片是不是精美），也希望能尽快找到解决办法。

怎么把锤子跟 FreeSWITCH 联系起来呢？我在锤子上装了几款软电话，跑起来挺流畅的:)。有朝一日，我希望能实现自己的软电话。

9.13 写在 2014 年最后一天

2014-12-31/Seven Du/FreeSWITCH 中文社区

著名博主月光博客发布了中国互联网 2014 大事记，微博上也有人有回忆自己的邮箱了。做为通信或呼叫中心行业的程序员，我们也来点回忆接龙吧？

我先来，这些都跟通信有关。

邮箱：

我的第一个邮箱是 263 的，大约 1999 年前注册的，当时还注册了好几个，后继传闻 263 要收费，就用我们名字加数字一下注册了 10 个，后来就真收费了，我就不用了。还注册了 163，新浪，Yahoo（不是中国）的邮箱，hotmail 邮箱主要是为了 MSN 用的，当然后来还试过 outlook 邮箱。但我最常用的邮箱是 Gmail 的。现在用 Gmail 是越来越累了，但就跟 Google 搜索一下，没有一个国内的替代品能代替了 Gmail 的，还得坚持用。

IM:

我最早是用 ICQ 的，后来用 oICQ，后来才是 QQ，不过，早期那些账号都不记得了。后来上班后一个同事送给我个 QQ 号，自用至今。但我前些年并不怎么用，因为早些年我用 Linux，而 QQ 在 Linux 上表现不好，还有人因为写 QQ 插件被抓。到外企上班后流行用 MSN。后来 MSN 不行了，我也不在外企工作了。而我的平台转到 Mac，Mac 版的 QQ 也渐渐好用了，我就会自然的转到 QQ，现在是 QQ 重度用户。另外，在移动端我几乎从没用过 QQ，不过我用微信。当然，Skype 我一直用，主要是跟老外交流。有一些专业的技术讨论群（如 FreeSWITCH）也用 IRC。

电话：

记得我是 97 年左右因为高考的事情才第一次打电话。由于电话里的声音很小，我听不清，所以我说话的声音很大，基本就是吼。那时打电话基本是到亲戚家里或单位上或很远的邮电局去打。刚上大学时，全宿舍楼共用传达室里一个电话，后来才发展到每个宿舍一个电话。有时到外面排很长的队用 IC 卡给远处的女朋友打电话。毕业后，买了一个 BP 机，那时的 BP 机已经有汉显了。由于在电信上班，于半年后买了一个小灵通，用了好几年。再后来买了一款 LG 的 CDMA 翻盖手机。去北京后还用过一款忘了牌子的有手写笔的手机。再后来买了一个阉割版的 iPhone 3GS，没有 Wifi 别提有多别扭。后来因为联通信号在我们办公楼里太烂，买了一个华为的双卡双待机，办了一个电信卡。在杭州西湖边上把屏幕摔坏了（但还能用），我也不需要双卡了，现在一直用锤子手机。

上大学时话机制限制只能插 IC 卡才能打电话，我们也用过连续拍叉簧的方法绕过它的限制用 200、300 卡拨打电话。也曾在公用电话上用拍叉的方法触发反极性而使计价器不计费，不过，有一次被发现了，这种方法就一直没再用。

其它：

我有幸用过几个著名的服务，但好就好像被 curse 了一样，只要我一用，大部分服务现在都不能用了，如 Google Search、Google Reader、Google Voice、DropBox、Twitter、Facebook、人人、新浪博客、新浪微博（还在用）等。

在 2014 年终之际，我们的 FreeSWITCH 中文论坛正式上线了，点击「阅读原文」可访问论坛并进行故事接龙。

<http://bbs.freeswitch.org.cn/t/2014nian-zui-hou-tian-liao-dian-huai-jiu-de-ba/186?>

9.14 致 FreeSWITCH-CN 公众号所有订阅者

2015-07-03/Seven Du/FreeSWITCH 中文社区

FreeSWITCH-CN 微信公众号自开张以来，到现在差不多有一年半了。

最初，写得比较勤快些。后来，就懒了。所以，一直到现在也没有机会能有微信上的原创标志和打赏功能。

至于懒得原因有很多，但我觉得主要的原因还是用户活跃度不高，我写作的积极性就少了。我最初写文章的时候，是从几十个订阅者的时候写的。后来，很快就有了一百多个订阅者。每篇文章大约有一百多阅读，虽然比起那些自媒体大牛动辄上万的阅读还差了几个数量级，但自我感觉还不错的。可是，直到今天，我有了上千的订阅者了，每篇的阅读量也还是那么 100 多人。

当然，阅读量和订阅者的数量主要跟我的水平有关。第二原因是选择了 FreeSWITCH 这么一个小众的话题，而我又不会向其它大师那样会讲成功学和各种鸡汤。在技术层面，我虽然涉猎面也挺广泛，但几乎没有真正的实践，因而，对其它话题也不难妄言。因此，本订阅号在可以预计的将来还是将专注于 FreeSWITCH。不过，我现在开始学习 OpenSIPS 了，也许将来可以写几篇。

我也一直在反思，是不是哪里我做的不对。

首先，不管在 QQ 群、BBS、还是在微信后台，我都很认真的对待大家的问题。但是，知道不？大部分问题我都不知道怎么回答。为什么？因为我知道的太多，而你的问题给的信息量太少。

这就是典型的信息不对称。举个例子，你安装了 FreeSWITCH，遇到了一个问题，就上来问——FreeSWITCH 打不了电话有人遇到吗？FreeSWITCH 不能录像是怎么回事？XXXX 问题有没有人遇到？

其实，你说的绝大部分问题我都遇到过。但是，某些问题只在某些版本里有，某些问题只是你配置的问题。你至少要告诉我你是用 Windows 还是 Linux 吧？如果是 Linux，是 Ubuntu、CentOS、

还是 Debian? 如果是 CentOS, 那么是 CentOS5, 6, 还是 7? FreeSWITCH 是什么版本的? 你的配置是默认配置还是有更改? 你是怎么遇到你的问题的, 如何重现你的问题? 等等。无论如何, 提前想一下, 如果你有问题需要我问你十个问题才能帮你解决, 那多半我没有耐心回答你的问题。

还有另外一种情况。即便是如上所说, 你只提了一个简单问题而没有任何信息, 我知道是怎么回事。因为我遇到过, 我知道它会在什么情况下发生。但是, 我不告诉你。

为什么?

今天就在人在 QQ 群里质疑说: 要说就说, 不愿意回答拉倒, 对一个小白用户提个问题就带来一大番说教……

我们社区定位是一个技术社区, 我希望大家都能学习和提高, 我希望授人以渔。如果我直接告诉你一个答案, 你多半会说我好, 但你不会思考, 下次遇到同样的问题还是解决不了。而我希望的是告诉你, 怎么才能解决你的问题, 下次该怎么提问, 该怎么自己想办法找答案。

那又有人说了, 小白嘛, 那又不知道这些, 你总得慢慢让他知道啊……

是的。我们为小白准备了「新手指南」, 「新手必读」……大家都是成年人了, 我不认为我需要告诉大家「新手必读」是需要读的。退一万步讲, 很多 QQ 群里的人不知道我们的官方网站是 www.FreeSWITCH.org.cn, 不知道不要紧, 我相信你能自己找出来。很多人找不到我们的微信公众号入口, 就在官方网站上啊, 那么大的二维码。很多人找《FreeSWITCH 权威指南》不知道到哪里买, 就在官方网站上, 那么大的广告……

好了, 你找到「新手指南」了, 但你没有好好看。

很多人说我们的 BBS 跟其它 BBS 不一样, 不会用。我知道, 改变习惯可能需要一代人的努力。但是我们的 BBS 是新一代的 BBS, 如果你想到 BBS 上发问, 最好看看「新手必读」。我们为了大家适应, 专门设立了新手灌水区, 可是, 你连上来练几把的时间都不舍得花, 上来就问把版面排得乱七八糟, 别人有心情回答你吗?

好像是我在教小学生一样。但是, 这些, 应该是你自己去学的。现在网络这么发达了, 这些东西都是现成的。如果这些东西都需要有人教的话, 那么, 我认为你在公司的表现也不过而而。

所以, 你想成为 FreeSWITCH 高手, 成为一个好的开发人员, 先从自己做起, 从这些小事做事。我希望我们社区里的每一位技术人员都有这样的基本素质, 同时, 传达给每一位新加入的成员。

好了, 言归正传。在摘要里用「有彩蛋」把大家忽悠来了, 就把彩蛋说一下。

其实, 还有好几个呢:

- 1) 主要目的, 我想看看有多少活跃用户。
- 2) 真彩蛋。本篇文章的((阅读量 + 点赞数量) x 1 元 + 截止周一晚上全部赏银) 用来发红包
- 3) 想收红包, 你需要加入一个微信群。下面有码, 人数有限制, 晚了进不来
- 4) 接下来我准备写一写 FreeSWITCH 里的视频功能, 连载
- 5) 「我和 FreeSWITCH 有个故事」有奖征文开始啦, 见 BBS

- 6) FreeSWITCH 沙龙将于今年 911 在北京某五星级大酒店举行，会场情况见题图。想成为赞助商或者想上面 * 分享演讲跟我联系
- 7) FreeSWITCH 培训将于 9.12-14 日在北京举行，现在在有赞上报名（顺着微信菜单能找到）最优惠。

9.15 FreeSWITCH oTo 阿里巴巴站回顾

2015-07-27/Seven Du/FreeSWITCH 中文社区



虽然航班延误，但我还是在当天凌晨 2 点多钟赶到附近的酒店睡下。

早上起来，吃完早餐，打车到阿里巴巴西溪园区门口，小伙伴们已经在门口等待了。我的老朋友老戴带着墨镜端坐在等待室里，不怒自威。

登记入园，拍照留念。很快就到达既定的会议室门口。门口有更多的小伙伴，只是我基本都不认识。

接待我们的是钉钉团队的技术骨干。简单跟他们交流下活动日程，调好投影，活动就开始了。

最开始，钉钉团队的技术骨干就给大家介绍了钉钉应用。虽然我们一直在用钉钉，但各种功能还是没有发掘到，听了也大有收获。话说钉钉确实上了一款很不错的应用，定位于企业团队高效协作，很好的平衡了即时通信以及语音通话、电话会议等功能，用起来相当方便。

接下来，我分享了一些我对 FreeSWITCH 的理解。发挥过头了，用得时间比预想的都长。

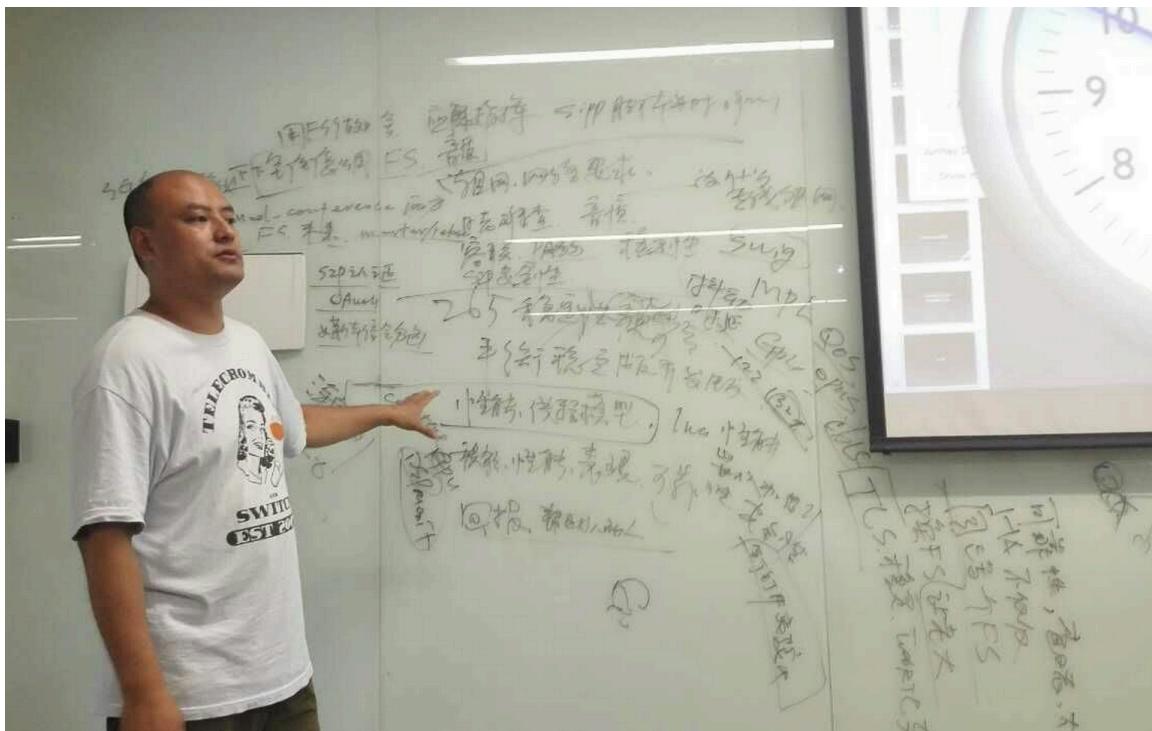
再接下來，是一个 1 分钟介绍自己活动。现场的每个人都用 1 分钟的时间介绍了自己与 FreeSWITCH 的渊源，以及参加本次活动希望讨论的 FreeSWITCH 相关的问题和主题等。活动现场



大约有 50 人左右，由于阿里是主场做战，因此，有很大一部分名额被他们占据了。其它的是杭州本地科技公司的居多，北京次之，还有人来自上海、深圳、昆山、无锡等。远道来的兄弟们，有的是正好在杭州出差，也有的是专门来的。



这个节目是非常好的一个节目，不仅增进了大家的相互了解，也带出了好多很好的话题。后面，我又针对各话题进行了点评，跟大家做了一些交流。这些话题大多集中在 FreeSWITCH 的架构、性能、视频、H265 等高端的技术话题，也有一些对 FreeSWITCH 社区的探讨、对传统电信通信与现代化互联网通信融合的分析和展望等。



最后，钉钉团队的技术人员也分享了他们使用 FreeSWITCH 的经验，他们遇到的特别的问题，

以及他们的解决方案。

2个小时的时间很快就过去了。中午阿里巴巴管饭，大伙吃了个食堂，荤的素的都有，爱吃多少吃多少，感觉进入了共产主义。

饭后，大家意犹未尽，三三两两地聚在一起又聊了起来，我不敢耽搁太久，匆匆赶往下一场活动（PostgreSQL 峰会）……

感谢阿里巴巴提供场地、水果和午餐，感谢钉钉团队的热情接待和技术分享。

9.16 一个真实的 FreeSWITCH 问题

2015-07-31/Seven Du/FreeSWITCH 中文社区

在 BBS 上有一个问题。我花了很多时间写『新手必读』、『新手指南』之类的，但是，大家就是不看。

好吧，就事论事，我准备用实际的例子来说明问题。

请听题：

外部打入一个电话到 freeswitch A 我想把这个电话重定向到 freeswitch B 看了下文档有 redirect 的 app 直接采用以下 app 直接定向到 B 机器然后日志报错 Redirecting to “unknown” 不知道我的思路是否有问题，书里没有发现相关重定向内容[哭]

这样的问题，你知道怎么回答他不？反正我不知道。实际上，看到这个问题，我自己试了一下，没有遇到他所说的问题。那么，我们能否把问题描述更清楚一些呢？

让我们试试：

今天遇到一个问题。外线打电话进入 FreeSWITCH A，我想让电话重定向到 FreeSWITCH B。

实际环境如下：FreeSWITCH 的版本是 1.4.20，操作系统是 Linux Debian 8。电话打到 FreeSWITCH 192.168.7.6:5060，被叫号码是 redirect，我想转到 192.168.7.6:5080，被叫号码改成 8888。我使用的 Dialplan 如下：

```
<extension name="redirect">
<condition field="destination_number" expression="^redirect$">
    <action application="redirect" data="sip:8888@192.168.7.6:5080"/>
</condition>
</extension>
```

然后，日志报错，……，日志和 SIP Trace 如下：

(此处省略 5000 字)

如果有人像上面这样提问题，是不是很愿意帮他解决呢？问题背景、流程、日志这些一定要有。你多花点时间描述你的问题，你就能更快的得到答案。否则，如果是原题，我只能告诉他：『我试了一下一点问题都没有，你需要把日志和 SIP Trace 贴出来。另外提问时最好用完整的 Dialplan』。

好吧，看我是怎么做的。

首先，我的 FreeSWITCH 版本是 Git 库的 master 版本。我用 Mac。然后，我构造了上面的 Dialplan。电话打到 redirect 后就调用 redirect 命令转到其它地方去。日志如下：

FreeSWITCH 收到 INVITE

```
recv 812 bytes from udp/[192.168.7.6]:65034 at 07:26:36.287912:  
-----  
INVITE sip:redirect@192.168.7.6 SIP/2.0
```

发送 100 Trying ...

```
-----  
send 369 bytes to udp/[192.168.7.6]:65034 at 07:26:36.288130:  
-----  
SIP/2.0 100 Trying
```

查找 Dialplan，执行 redirect，挂机

```
[INFO] mod_dialplan_xml.c:637 Processing 1000 <1000>->redirect in context default  
[NOTICE] mod_sofia.c:1920 Hangup sofia/internal/1000@192.168.7.6 [CS_EXECUTE] [NORMAL_UNSPECIFIED]
```

发出到302重定向到8888

```
send 946 bytes to udp/[192.168.7.6]:65034 at 07:26:36.321772:  
-----  
SIP/2.0 302 Moved Temporarily
```

收到 ACK

```
-----  
recv 336 bytes from udp/[192.168.7.6]:65034 at 07:26:36.322595:  
-----  
ACK sip:redirect@192.168.7.6 SIP/2.0
```

流程到这里就应该结束了。不过，由于我并没有两个 FreeSWITCH，而是只用了同一样 FreeSWITCH，只不到是重定向到另外一个端口上。可以看到，客户端收到 302 后又重新向重定向的端口发起了 INVITE 请求

```
-----  
recv 1076 bytes from udp/[192.168.7.6]:65034 at 07:26:36.322911:  
-----  
INVITE sip:8888@192.168.7.6:5080 SIP/2.0
```

一切正常。

如果提问者也像我这样有耐心，也许自己就知道问题出在哪里了。这又引出一个话题『橡皮鸭子问题』。点击『阅读原文』看一下吧。

<http://blog.csdn.net/haoel/article/details/4914403>

另外，本文就事论事，对事不对人，对原 BBS 问题作者没有不敬，如有冒犯，多多原谅。

写在最后

本书将持续更新，这就是电子版的好处…

如果你对书中的内容和章节安排等有什么意见或建议，欢迎与我联系。如果你建议的内容适合放在本书里，我会考虑写进去；如果不适合放到本书中，我也会考虑写其它主题的书。

如果你的公司想在本书中植入广告或者赤裸裸地做广告，也欢迎与我们联系。

电子邮件：info@x-y-t.cn。

作者简介

杜金房 (网名: Seven) 资深网络通信技术专家, 在网络通信领域耕耘近 15 年, 精通 VoIP、SIP 和 FreeSWITCH 等各种网络协议和技术, 经验十分丰富。有超过 7 年的 FreeSWITCH 应用和开发经验, 不仅为国内大型通信服务厂商提供技术支持和解决方案, 而且客户还遍及美国、印度等海外国家。

FreeSWITCH-CN 中文社区创始人兼执行主席, 被誉为国内 FreeSWITCH 领域的『第一人』; 在 FreeSWITCH 开源社区非常活跃, 不仅经常为开源社区提交补丁和新功能、新特性, 而且还开发了很多外围模块和外围软件; 此外, 他经常在 FreeSWITCH 的 Wiki 上分享自己的使用心得和经验、在 FreeSWITCH IRC、QQ 及微信群中热心回答网友提问, 并不定期在国内不同城市举行 FreeSWITCH 技术培训; 自 2011 年起每年都应邀参加在美国芝加哥举办的 ClueCon 大会, 并发表主题演讲。

此外, 他还精通 C、Erlang、Ruby、Lua 等语言相关的技术。

著有《FreeSWITCH 权威指南》, 2014 年出版。

创办了[北京信悦通科技有限公司](#)和[烟台小樱桃网络科技有限公司](#), 提供 FreeSWITCH 培训和商业技术支持服务。

版权声明

本书版权归作者所有，任何人未经书面授权均不得分发此书。

本书电子版仅在 SelfStore (<https://selfstore.io/~dujinfang>) 上发布。如果您不小心从其它渠道获得本书，请删除您的版本并到 SelfStore 上购买正版。

本书纸质版仅随电子版赠送。SelfStore 支持买家自由定价。如果您购买电子书时支付超过一定金额，可以通过电子邮件联系我们赠送一本精美打印的纸质书。

关于 SelfStore: 就在本书即将付印之时，SelfStore 宣布关闭。笔者实感遗憾。如果届时 SelfStore 真的关闭，我们将会寻找其它方式发布本书。关于 SelfStore 的声明见这里：

<http://blog.selfstore.io/2016/08/22/selfstore-will-be-closed/>

广告

关于广告的广告

请允许我在本书中发布广告。广告合作联系邮箱：info@x-y-t.cn。

FreeSWITCH 第五届开发者沙龙将于 2016 年 8 月 27 日在京举行

<http://www.freeswitch.org.cn/2016.html>

FreeSWITCH 培训 2016 夏季班（北京站）将于 2016 年 8 月 28-30 日在京举行

<http://x-y-t.cn/fst1608.html>

烟台小樱桃网络科技有限公司提供商业 FreeSWITCH 及 OpenSIPS 技术支持

网址：<http://x-y-t.cn> 邮箱：info@x-y-t.cn

烟台小樱桃网络科技有限公司是潮流网络（GrandStream）山东总代理

深圳市潮流网络技术有限公司（Grandstream）是全球知名的统一通讯和整体解决方案厂商和全球 TOP3 VoIP 终端供应商，是国内 IMS、统一通信、呼叫中心、调度市场、视频监控的主要 SIP 终端供货厂商，成功案例包括京东、网易、凡客诚品、平安、中集、东航、国家电网、中石化、中石油、

外交部、中移动、中电信等等，公司是中国三大运营商 IMS 市场主要核心合作 SIP 终端厂商，入围中国电信集团 IMS SIP 电话短名单，持续配合运营商研究院 SIP 硬终端的核心业务及协议定制，参与及进入中电信、中移动多个省试点应用和产品供应厂商。在全球与渠道及增值合作伙伴成功服务将近千万终端用户，包括全球主流运营商数十万套终端以及美国共和党和美国民主党数万套 SIP 电话和网关等大型成功部署案例，连续 10 年保持近 50% 年复合增长率。

烟台小樱桃网络科技有限公司，是潮流全线产品在山东省的总代理。

联系方式：邮箱：info@x-y-t.cn 电话 0535-6753997

FreeSWITCH 相关图书

《FreeSWITCH 文集》收集了一些 FreeSWITCH 文章，相比其它 FreeSWITCH 书来说，技术内容比较少，便于非技术人员快速了解 FreeSWITCH。

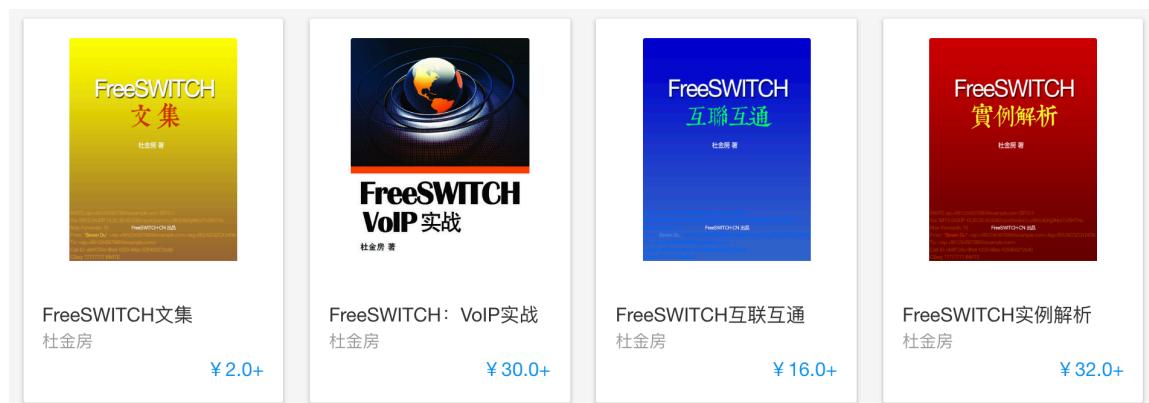
《FreeSWITCH 互联互通》主要收集了一些互联互通的例子。书中有些例子来自《FreeSWITCH 权威指南》。

《FreeSWITCH 实例解析》收集了一些如何使用 FreeSWITCH 的实际例子，方便读者参考。书中有些内容来自《FreeSWITCH 权威指南》。

《FreeSWITCH 实战》是《FreeSWITCH 权威指南》的前身，不再更新，但该书有其历史意义。

以上图书均为电子书，可在 SelfStore (<https://selfstore.io/~dujinfang>) 上购买。SelfStore 支持买家自由定价。如果您购买电子书时支付超过一定金额，可以联系通过电子邮件联系我们赠送一本精美打印的纸质书。

《FreeSWITCH 权威指南》是正式出版的纸质书，纸质版和电子版均可在以下网站购买：<http://book.dujinfang.com>。





作者简介



杜金房（网名：Sevan Du）资深网络通信技术专家，在各种通信领域耕耘15年，精通VoIP、SIP和FreeSWITCH等多种网络协议和技术。经验十分丰富。有超过6年的FreeSWITCH应用和开发经验，不仅为客户提供过各种语音通信解决方案，而且客户遍及全国。他是专家，FreeSWITCH-CNP中文社区创始人之一，被誉为国内FreeSWITCH领域的“第一人”。他的FreeSWITCH开源项目非常出色，不仅在国内有很高的知名度，而且在国际上也有影响力，还开发了更多的国际版本的客户端。此外，他已经将FreeSWITCH(Wall)上早已经使用的心得和经验，自费SWITCH INC 和 QSLab 公司的资助下，翻译成中文，贡献给国内的FreeSWITCH爱好者。2011年，2012年RSO大会应邀参加叽叽喳喳办的CicaCon大会，并发表主题演讲。此外，他还精通与C、Erlang、Ruby、Lua等语言相关的技术。



FreeSWITCH 权威指南

FreeSWITCH: The Definitive Guide

FreeSWITCH是我2006年的一个梦想，到现在为止已经近10年了。它从最初的一个由5个文件组成的概念验证程序已经成长为一个拥有400个核心代码，另外还有其他300万行以下开源核心的分布式电话系统。感谢大家，因为有了你们的贡献，使这个项目能够发展到今天。感谢所有参与FreeSWITCH项目的志愿者，是你们不计报酬的贡献，使FreeSWITCH社区更加壮大繁荣。我感谢本书的合作者加入我的FreeSWITCH中文社区，与所有FreeSWITCH爱好者一起成长，并一起见证我们迈向下一个里程碑，迎接自由电话的强大、成熟。

我在2006年第一次遇到Sevan，从那时起一直到现在，他做了大量的工作——帮助我们学习了所有的FreeSWITCH相关的技术，并贡献了很多的bug报告及修复代码，甚至数不清为了写这本书对FreeSWITCH进行调试，还贡献了至少5个bug。没有他，我们可能永远不会有这么大的中国读者基础。特别的，感谢FreeSWITCH的合作者。

——Anthony Minasale II FreeSWITCH之父

—— Jonathan Palley Siprius公司CTO

出版地：010-88376004
邮局代号：80-1711-1
印制地：北京中海星润印务有限公司
书名：FreeSWITCH权威指南
作者：杜金房 张令芳
定价：65.00元

FreeSWITCH 权威指南

杜金房 张令芳著



FreeSWITCH 权威指南

杜金房 张令芳著

机械工业出版社

本书是FreeSWITCH项目“第一人”、全球FreeSWITCH社区知名专家编写，FreeSWITCH之父Anthony Minasale倾力打造。全书共12章，涵盖FreeSWITCH使用、维扩、二次开发、源码分析等各方面；实践性强，适合大量实践，从单个独立应用的实现到集群的部署，适合所有对FreeSWITCH感兴趣的读者。

内容简介

FreeSWITCH是世界上第一个平台的、模块化的免费开源的、多协议的软交换系统。本书是FreeSWITCH领域最具权威的著作之一。在这本书面前，FreeSWITCH无秘密！

由中国的FreeSWITCH项目“第一人”，全球FreeSWITCH开源社区知名专家，FreeSWITCH-CNP中文社区创始人兼执行主编Sevan Du编写，FreeSWITCH之父Anthony Minasale倾力打造。全书共12章，涵盖FreeSWITCH使用、维扩、二次开发、源码分析等各方面；实践性强，适合大量实践，从单个独立应用的实现到集群的部署，适合所有对FreeSWITCH感兴趣的读者。

本书是FreeSWITCH项目“第一人”、全球FreeSWITCH开源社区知名专家，FreeSWITCH-CNP中文社区创始人兼执行主编Sevan Du编写，FreeSWITCH之父Anthony Minasale倾力打造。全书共12章，涵盖FreeSWITCH使用、维扩、二次开发、源码分析等各方面；实践性强，适合大量实践，从单个独立应用的实现到集群的部署，适合所有对FreeSWITCH感兴趣的读者。



ISBN 978-7-111-46624-7

THIS PAGE INTENTIONALLY LEFT BLANK.



鬻宇王 宝金卦 : 十岁面桂
斐 高 茲 芮 宾 : 辜嗣玉责