链滴

百度不限速下载器 BND2 技术架构简介

作者：88250

原文链接：https://ld246.com/article/1535277215816

来源网站：链滴

BND2 简介

BND2 是一款图形界面的百度不限速下载器，支持 64 位 Windows 和 Mac，下载地址。

高速下载原理

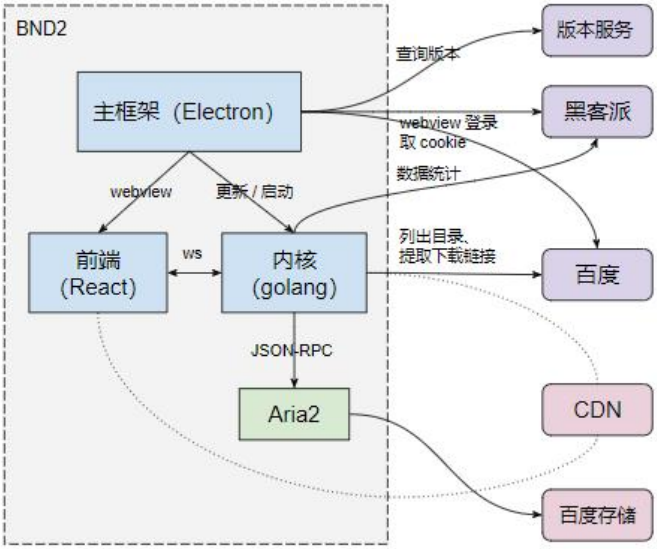
- 通过 PCS API 获得文件、下载链接
- 通过 Aria2 实现高速下载

和百度网盘下载相关的内容我们就介绍到这里。下面我们主要介绍框架和实现相关的东西，如果你刚想做一个桌面软件，可以参考借鉴一下。

技术架构

BND2 基于 Electron、React、golang 实现，通过主框架 - UI - 内核的分离实现 UI 以及内核的自动新。

- 主框架：Electron 主进程，负责版本检查、账号登录以及管理内核
- UI：React 实现主界面，编译后上 CDN，通过 Electron <webview> 加载
- 内核：golang 实现的 HTTP 服务，负责响应 UI 请求，管理 Aria2 进程。 编译后可执行二进制上 DN 提供给主框架拉取更新



为什么选 Electron、React 和 golang

在选择 Electron 之前，我们大致看了下 NW.js 和 CEF，他们肯定都可以实现我们想要的，仅从文档社区上看，Electron 比较和胃口 yum

React 和 golang 的选择是因为我们开发团队对这两个技术比较熟悉，特别在工期紧任务重的情况下选撸起来快的准没错。

自检更新

主框架启动时会从远程更新服务器上获取版本，主要包括两个版本信息：

- 主框架版本：如果有升级，则提示用户需要从指定位置手动下载安装包
- 内核版本：如果有升级，则自动从指定位置下载内核二进制

Electron 内建有热更新机制，但不支持 Linux（虽然目前 BND2 也没支持 Linux，但未来会考虑支持，并且我们有自己的产品版本管理机制，所以就没有考虑使用 Electron 内建的热更新了。

另外，无论给 Windows 还是 Mac 的升级包都是使用 zip 包，主要是为了统一。但 zip 包不会保留执行权限，Linux/Mac 解压后需要再给二进制赋一下可执行权限。

账号登录

为了“复用”登录，我们是通过 webview 直接引的待登录站点的 web 登录界面。没有在本地做登录主要是考虑到：登录逻辑复杂，特别是登录异常处理。比如二维码、验证码、短信校验等等，这些如通过后端对接非常繁琐，外部站点改一下实现就要跟着变，维护工作量巨大。

登录后我们只需要获取一下关键的 cookie 就行了，然后把这个 cookie 传给内核，后续由内核负责外部站点的接口进行交互。

webview

通过 webview 从 CDN 加载，如果要升级只需要编译并发布 CDN。

每次启动加载的 HTML 是通过在客户端加入 ?时间戳来引入，也就是说这个文件肯定不会在 CDN 上中，都是回源加载最新的，CDN 主要是加速其引入的其他资源文件。

前后端交互

UI 和内核的交互没有通过传统的 HTTP AJAX 实现，而是统一通过 WebSocket 来实现全双工的异步讯。这样设计主要是考虑到 BND2 在异步推送的场景比较多，比如全局统计、下载进度统计、反馈核报错等。

实现上我们做了一个简单的抽象封装，通过命令模式实现指令分发和执行：

```
// WebSocket 消息处理
m.HandleMessage(func(s *melody.Session, msg []byte) {
    request := map[string]interface{}{}
    json.Unmarshal(msg, &request) // 反序列表指令

    cmdStr := request["cmd"].(string)
    cmd := command.Commands[cmdStr] // 指令路由分发
    param := request["param"].(map[string]interface{})

    go cmd.Exec(param) // 异步执行
}
```

前端也非常类似，在 ws.onmessage 时分发并执行指令。

安全性

桌面软件容易被破解主要是因为整个运行时都是在客户机上，破解者想怎么调试都可以。

比如破解者可以通过网络抓包对关键的请求接口进行修改，从而影响后续的执行逻辑，这一点可以通加密请求响应数据来 稍微 加大破解难度。再比如，破解者可以通过反汇编调试，找到并修改关键变来跳过某些判断检查，这一点可以通过加壳来 稍微 加大破解难度。

总之，没有绝对的安全，防君子不防小人。

总结

本文主要描述了一种现代富客户端技术架构的实现思路，以 BND2 为实例验证了该思路的可行性。

更进一步，我们可以考虑通过一致性状态分布协议（比如区块链、IPFS）尝试数据存储的分布，从而现真正意义上的富客户端。这里只是提个引子，之后有新会更。