QCon
全球软件开发大会

网易云音乐网络库跨平台化实践

网易云音乐 音视频实验室 陈松茂

让我们一起回忆下…

可能是你无奈的

多端适配

- PD：XXX优化，Android上了，iOS也上下?
- 你：再加两周吧，赶赶...

一致性问题

- SDK：XXX只有Android能采集，iOS采集不了
- 你：我在后端“兼容”下吧...

被遗忘的PC

- PC：XXX有PC端的SDK吗?
- 你：没有，自己实现吧...

“一个顶十个”

- 老板：隔壁某厂，自己实现了XXX协议，效果不错，了解下?
- 你：听说他们投了一个团队呢，我们...，哎...

个人简介

陈松茂 网易云音乐

2020年底加入云音乐，一站式网络解决方案技术负责人

目前从事跨平台网络解决方案相关研究，旨在降低多端网络工作的研发成本，以及多应用间的接入成本，用较低的成本获得持续可观的性能及能效提升

曾就职于阿里，长期从事于 Chromium 相关技术研究和应用，拥有丰富的浏览器开发和内核升级经验

目录

01 背景介绍

02 方案设计

03 落地实践

04 后续规划

背景介绍

现状 → 诉求 → 挑战

现状

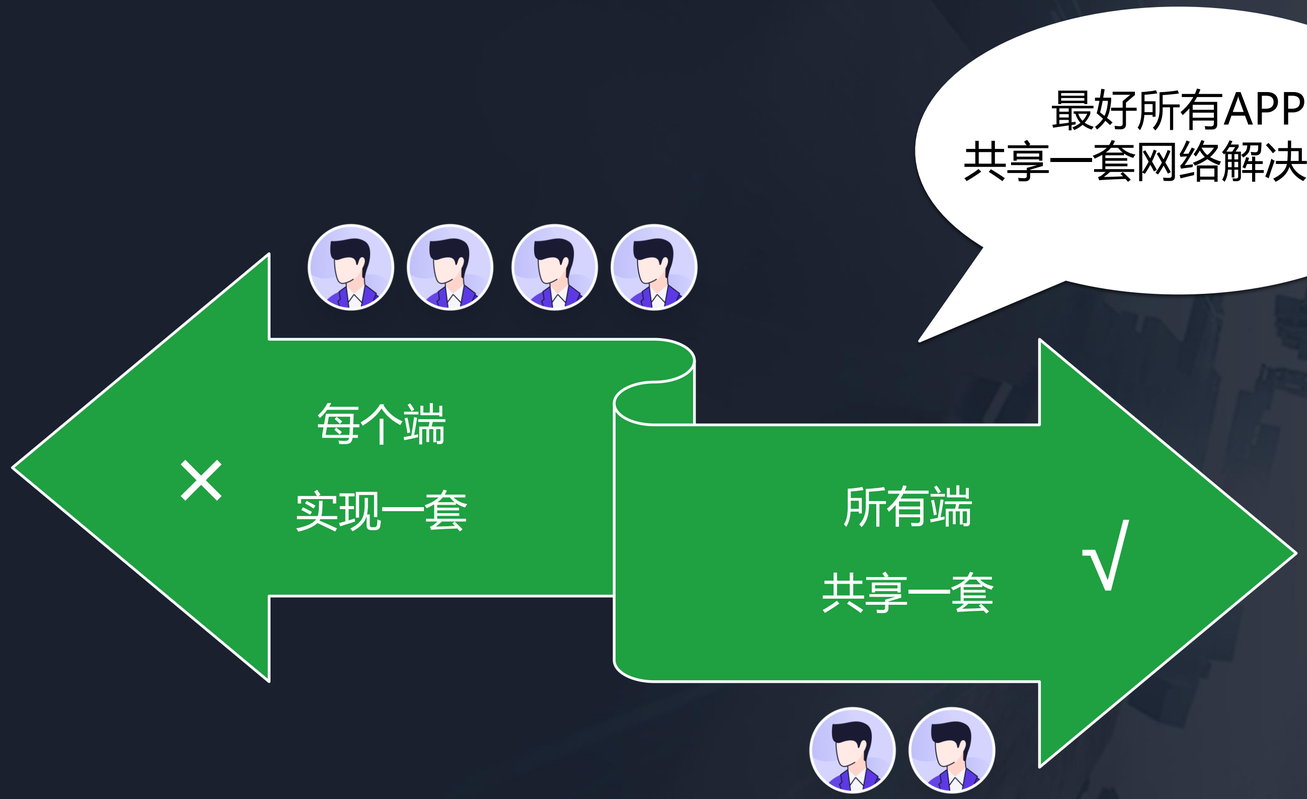

网络策略1 v1
网络策略2 v1
网络策略3 v1
网络策略4 v1
APM v1
HTTP DNS v1
OkHttp
Android


网络策略1 v2
网络策略2 v2
网络策略3 v2
网络策略4 v2
APM v2
HTTP DNS v2
CFNetwork/AFNetwork
iOS


网络策略1 v3
网络策略2 v3
网络策略3 v3
网络策略4 v3
APM v3
HTTP DNS v3
Chromium Net
Windows


网络策略1 v4
网络策略2 v4
网络策略3 v4
网络策略4 v4
APM v4
HTTP DNS v4
NSURLConnection
Mac

诉求



挑战

目录

01 背景介绍

02 方案设计

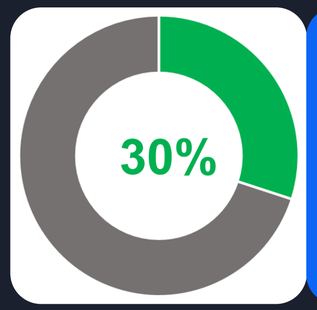
03 落地实践

04 后续规划

方案设计

设计思路 → 网络库设计 → 升级铺垫 → 避坑指南

跨平台设计思路



核心逻辑
(Utils)
跨平台化

网络策略1

网络策略2

网络策略3

网络策略4

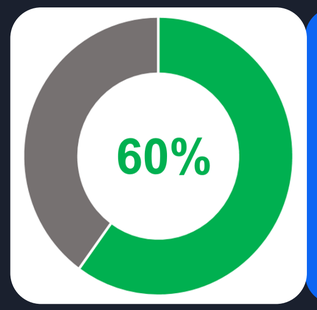
APM

HTTP
DNS

OkHttp/CFNetwork/AFNetwork/...

Android/iOS/Win/Mac

跨平台设计思路



完整功能
（组件）
跨平台化

网络策略1
网络策略2
网络策略3
网络策略4
APM
HTTP
DNS
网络代理
OkHttp/CFNetwork/AFNetwork/...
Android/iOS/Win/Mac

跨平台设计思路

整体链路
(方案)
跨平台化

网络策略1

网络策略2

网络策略3

网络策略4

APM

HTTP
DNS

跨平台网络库

Android/iOS/Win/Mac

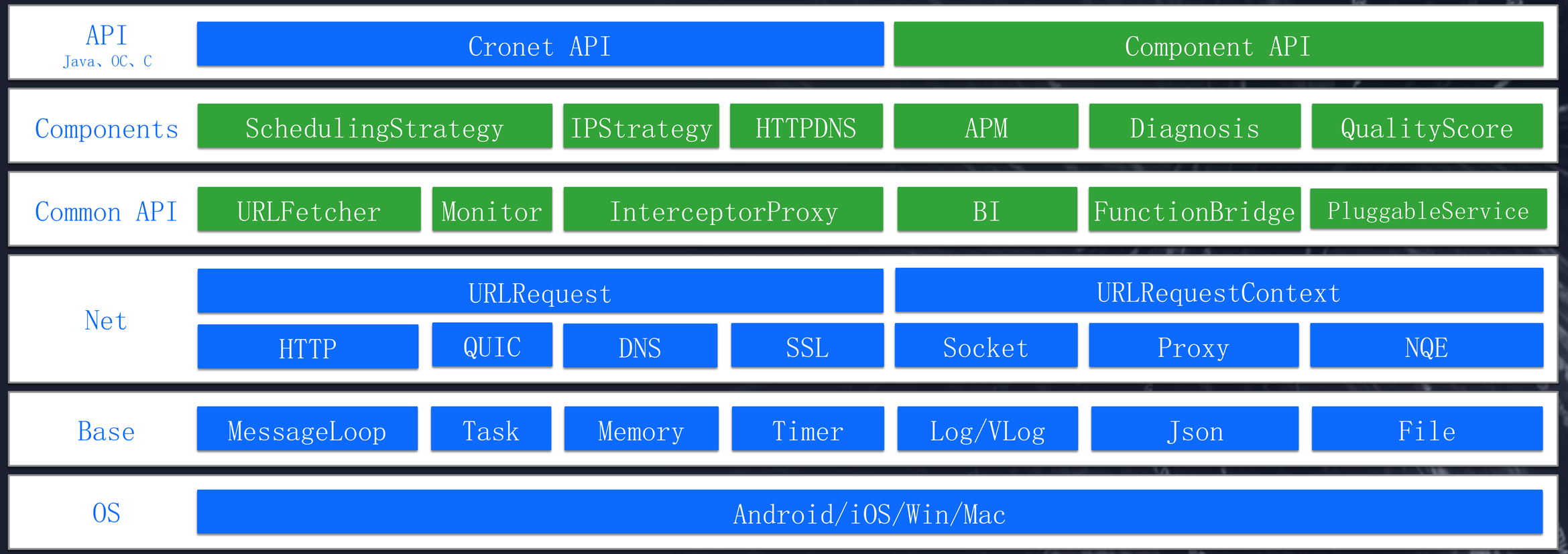
跨平台设计思路（小结）

跨平台的方法有很多，
我们选择了最彻底的一种

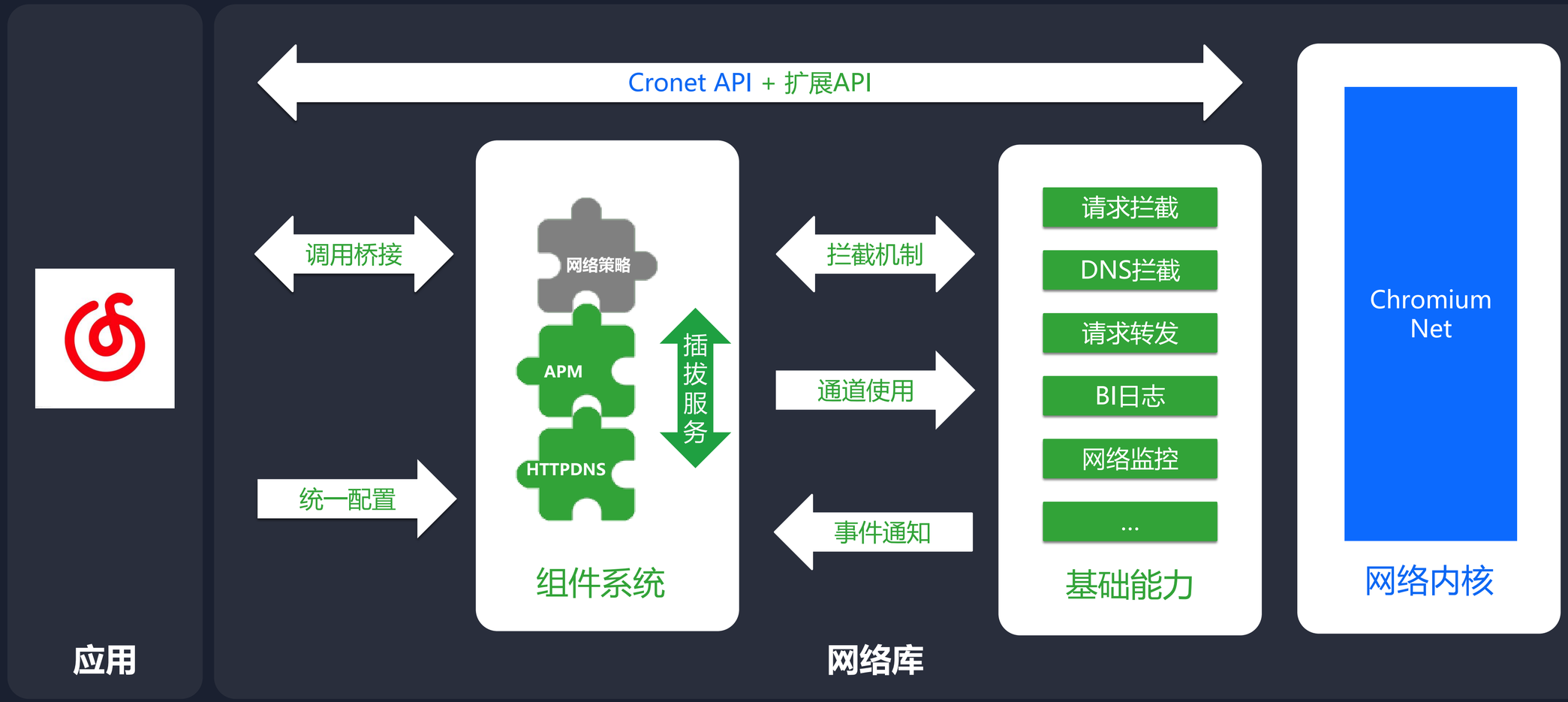
跨平台网络库选型

	Cronet（base on chromium net）
跨平台性	Android、iOS、Windows、MacOS、Linux
协议支持	HTTP 协议、HTTP/2 协议、QUIC 协议
网络优化	预连接、DNS缓存、Session复用、TCP Fast Open
普及程度（移动端）	Google系、百度、微博、网易传媒、头条系、蘑菇街
活跃程度	极其活跃，随Chromium开源项目持续演进中
开源协议	BSD许可协议

跨平台网络库架构


API
Java、OC、C
Cronet API
Component API
Components
SchedulingStrategy
IPStrategy
HTTPDNS
APM
Diagnosis
QualityScore
Common API
URLFetcher
Monitor
InterceptorProxy
BI
FunctionBridge
PluggableService
Net
URLRequest
URLRequestContext
HTTP
QUIC
DNS
SSL
Socket
Proxy
NQE
Base
MessageLoop
Task
Memory
Timer
Log/VLog
Json
File
OS
Android/iOS/Win/Mac

跨平台网络库交互


Cronet API + 扩展API
调用桥接
统一配置
网络策略
APM
HTTPDNS
插拔服务
拦截机制
通道使用
事件通知
请求拦截
DNS拦截
请求转发
BI日志
网络监控
...
Chromium Net
组件系统
基础能力
网络内核
应用
网络库

方案背后的思考

网络框架（可复用）

- 主库，框架级复用
- 更新内核及安全补丁
- 提供组件管理、基础能力封装
- 旨在降低网络内核定制成本

+

能力集（可扩展 ）

- 三方库，组件级复用
- 可根据需要灵活组合
- 提供通用能力和个性化能力
- 旨在满足多变场景或定制业务

网络库设计（小结）

我们直接基于开源的Cronet方案
构建了云音乐自己的统一网络库方案
并采用“网络框架+能力集（组件）”的模式进行业务下沉

看起来不错，
但升级怎么办…

Cronet升级（无法避免的问题）

升级原因

- 修复问题
- 获得特性
- 享受成果

升级之痛

- 代码冲突
- 合并冲突
- 功能衰退

升级铺垫

- 减少侵入
- 做好隔离
- 单测覆盖

Cronet升级铺垫-减少侵入

- Interface
- Delegate
- Observer
- Interceptor

- 适配层
- 组件机制
- 通道封装
- 插拔服务

- 网络策略
- APM监控
- HTTPDNS
- 网络诊断
- 业务组件

Cronet升级铺垫-代码隔离

```
bool·ClientSocketPoolBaseHelper::AssignIdleSocketToRequest(
····const·Request&·request,·Group*·group)·{
··std::list<IdleSocket>*·idle_sockets·=·group->mutable_idle_sockets();
··auto·idle_socket_it·=·idle_sockets->end();
··for·(auto·it·=·idle_sockets->begin();·it·!=·idle_sockets->end();)·{
····if·(!it->IsUsable())·{
·····DecrementIdleCount();
·····delete·it->socket;
·····it·=·idle_sockets->erase(it);
·····continue;
····}
```

源文件：
关于Socket复用

```
bool·ClientSocketPoolBaseHelper::AssignIdleSocketToRequest(
····const·Request&·request,·Group*·group)·{
··std::list<IdleSocket>*·idle_sockets·=·group->mutable_idle_sockets();
··auto·idle_socket_it·=·idle_sockets->end();
··for·(auto·it·=·idle_sockets->begin();·it·!=·idle_sockets->end();)·{
····if·(!it->IsUsable()·||·!SocketCanBeReuse(request,·*(it->socket)))·{
·····DecrementIdleCount();
·····delete·it->socket;
·····it·=·idle_sockets->erase(it);
·····continue;
····}
```

修改版V1：无隔离，
无法察觉是否有修改

```
bool·ClientSocketPoolBaseHelper::AssignIdleSocketToRequest(
····const·Request&·request,·Group*·group)·{
··std::list<IdleSocket>*·idle_sockets·=·group->mutable_idle_sockets();
··auto·idle_socket_it·=·idle_sockets->end();
··for·(auto·it·=·idle_sockets->begin();·it·!=·idle_sockets->end();)·{
····//·NetEase-{
····//·if·(!it->IsUsable())·{
····//·NetEase-}
····//·NetEase+{
····if·(!it->IsUsable()·||·!SocketCanBeReuse(request,·*(it->socket)))·
····//·NetEase+}
·····DecrementIdleCount();
·····delete·it->socket;
·····it·=·idle_sockets->erase(it);
```

修改版V2：使用注释“隔离” 无法编译出原始代码

为什么“源”码放前面？

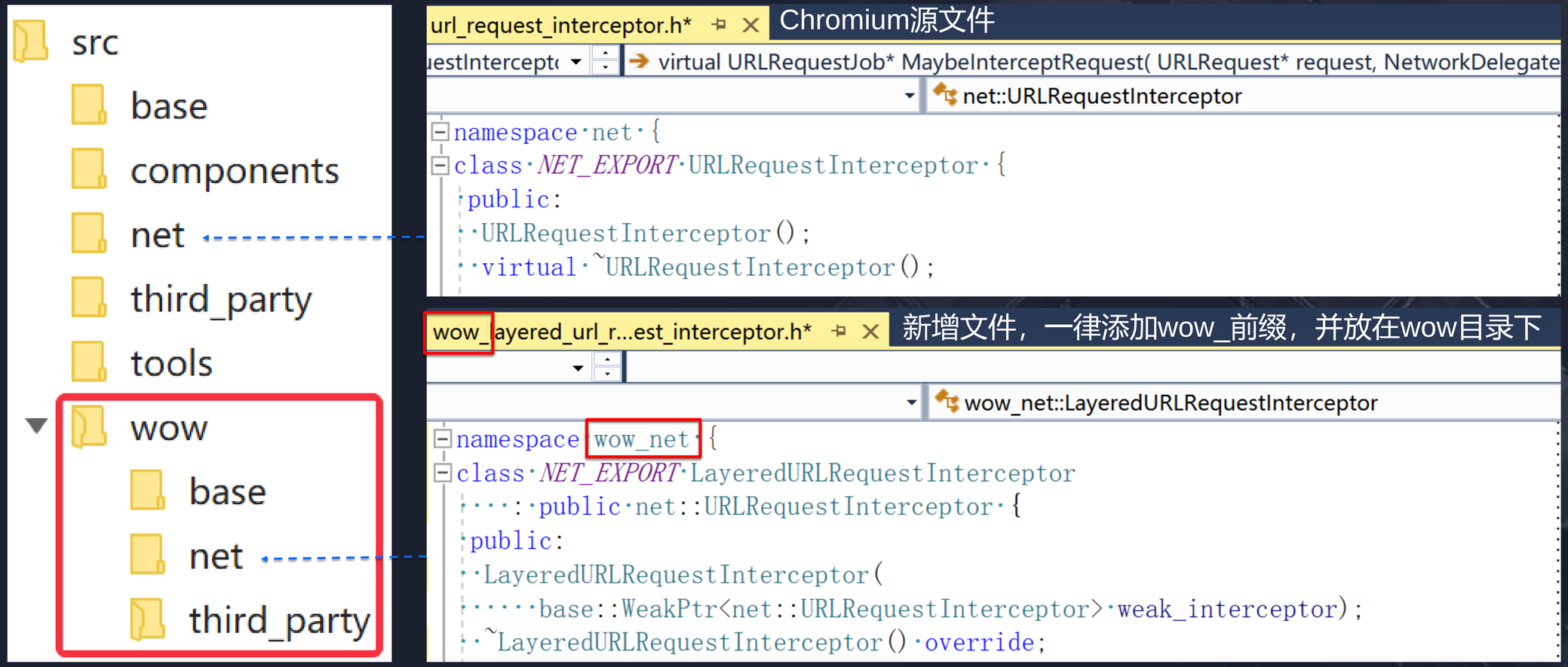
```
bool·ClientSocketPoolBaseHelper::AssignIdleSocketToRequest(
····const·Request&·request,·Group*·group)·{
··std::list<IdleSocket>*·idle_sockets·=·group->mutable_idle_sockets();
··auto·idle_socket_it·=·idle_sockets->end();
··for·(auto·it·=·idle_sockets->begin();·it·!=·idle_sockets->end();)·{
#if·!defined(WOW_BUILD)
····if·(!it->IsUsable())·{
#else
····if·(!it->IsUsable()·||·!SocketCanBeReuse(request,·*(it->socket)))·{
#endif
·····DecrementIdleCount();
·····delete·it->socket;
·····it·=·idle_sockets->erase(it);
·····continue;
```

修改版V3 :使用宏开关隔离,改动一目了然， “源” 快速切换

Cronet升级铺垫-代码隔离

1. C/C++、OC文件：宏开关WOW_BUILD、WOW_TRIM
2. Java文件：static final变量WOW_BUILD、WOW_TRIM
   （定义在BuildConfig.template文件，可通过宏开关控制）
3. Python文件：环境变量WOW_BUILD、WOW_TRIM
4. GN、GNI文件：build flag变量wow_build、wow_trim

Cronet升级铺垫-文件隔离



Cronet升级铺垫（小结）

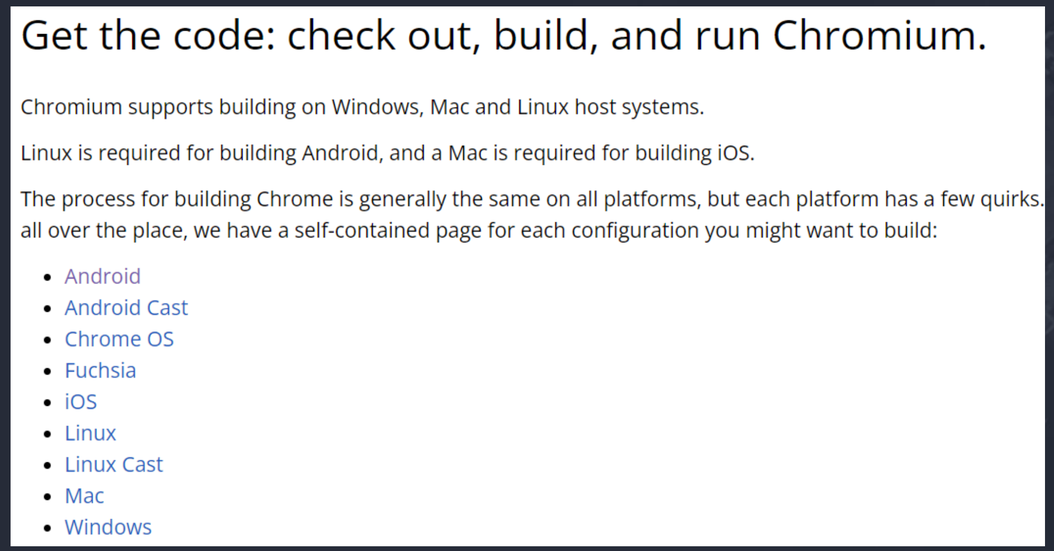
Cronet升级无法避免，也没有银弹
我们采用了低成本、可推行的“技巧”，让升级变得相对容易


看起来不错，
我也想试试了…

避坑指南（文档）

https://chromium.googlesource.com/chromium/src/+/refs/tags/72.0.****.***/docs/get_the_code.md

https://chromium.googlesource.com/chromium/src/+/refs/heads/main/docs/get_the_code.md



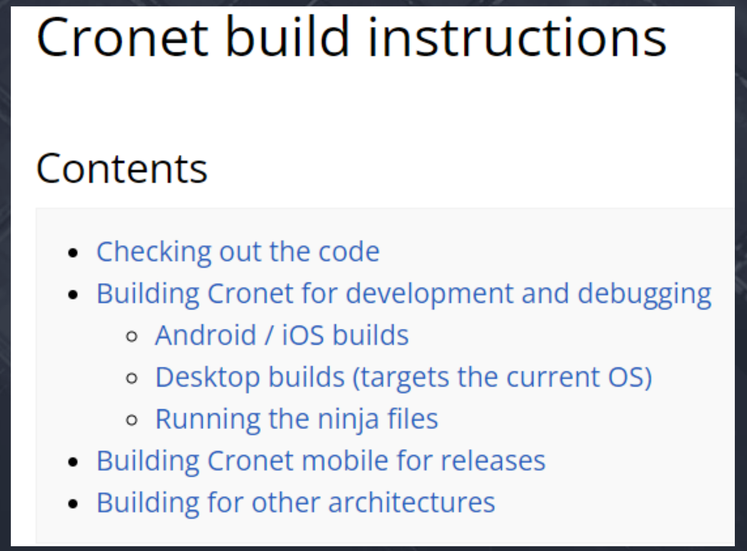
Get the code: check out, build, and run Chromium.

Chromium supports building on Windows, Mac and Linux host systems.

Linux is required for building Android, and a Mac is required for building iOS.

The process for building Chrome is generally the same on all platforms, but each platform has a few quirks. all over the place, we have a self-contained page for each configuration you might want to build:

- Android
- Android Cast
- Chrome OS
- Fuchsia
- iOS
- Linux
- Linux Cast
- Mac
- Windows



Cronet build instructions

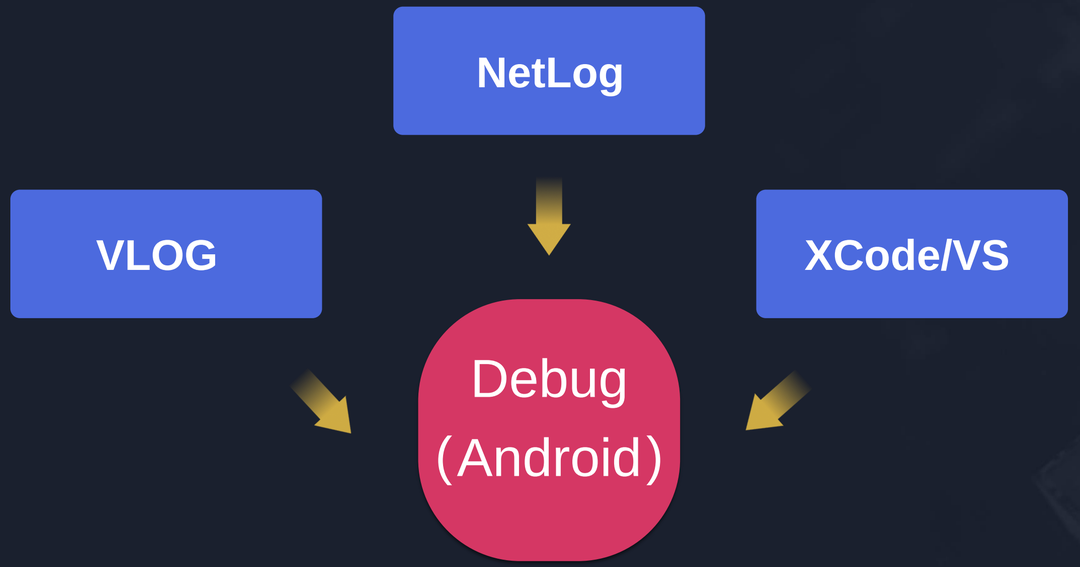
Contents

- Checking out the code
- Building Cronet for development and debugging
  - Android / iOS builds
  - Desktop builds (targets the current OS)
  - Running the ninja files
- Building Cronet mobile for releases
- Building for other architectures

避坑指南（编译）

	Android	iOS/Mac	Windows
开发工具	VSCode	XCode Command Line Tools->Xcode10.03	Visual Studio 2017
编译环境	Ubuntu16.04	Mac	Win10 (With SDK 10.0.17134)
构建系统	GN+Ninja	GN+Ninja	GN+Ninja

避坑指南（调试）

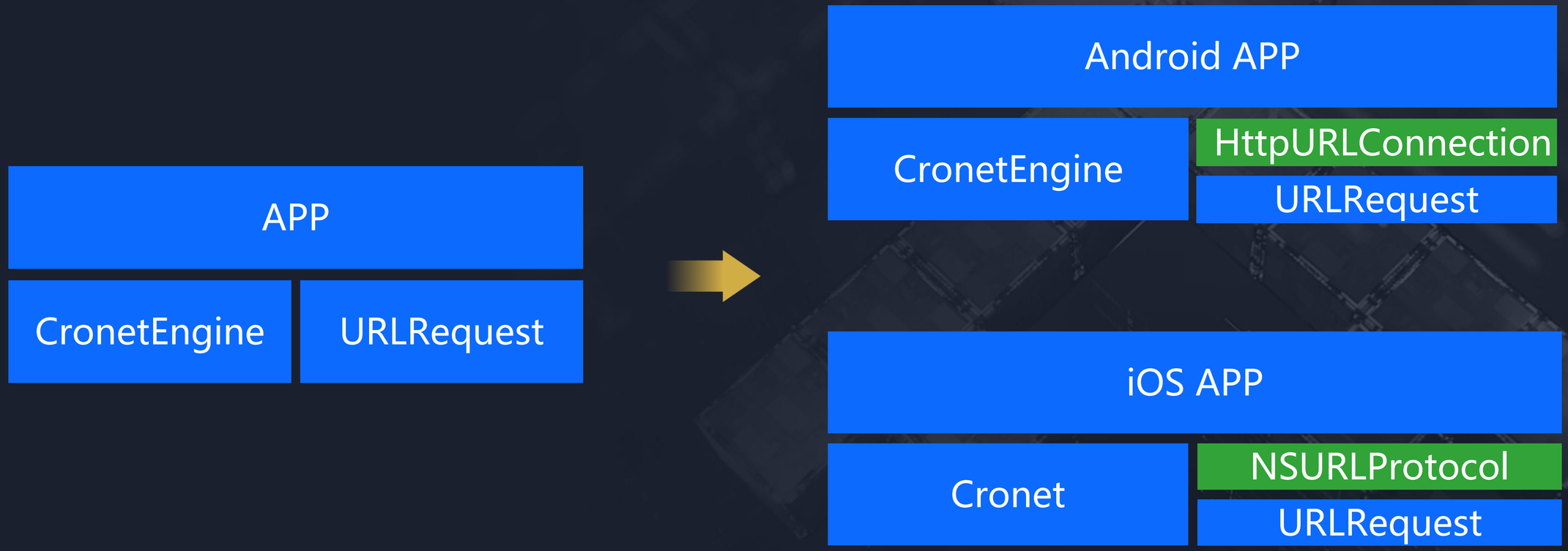


Testing and debugging Cronet for Android

Contents

- Checkout and build
- Running tests locally
  - Running Cronet Java unit tests
  - Running net_unittests and cronet_unittests_android
  - Running Cronet performance tests
- Running tests remotely
- Debugging
  - Debug Log
  - Network Log
  - Symbolicating crash stacks

避坑指南（接入）


APP
CronetEngine
URLRequest
Android APP
CronetEngine
HttpURLConnection
URLRequest
iOS APP
Cronet
NSURLProtocol
URLRequest

避坑指南（小结）

基于Cronet进行二次开发
各端SDK的编译、调试、熟悉
是一道“拦路虎”，但只是“纸老虎”

目录

01 背景介绍

02 方案设计

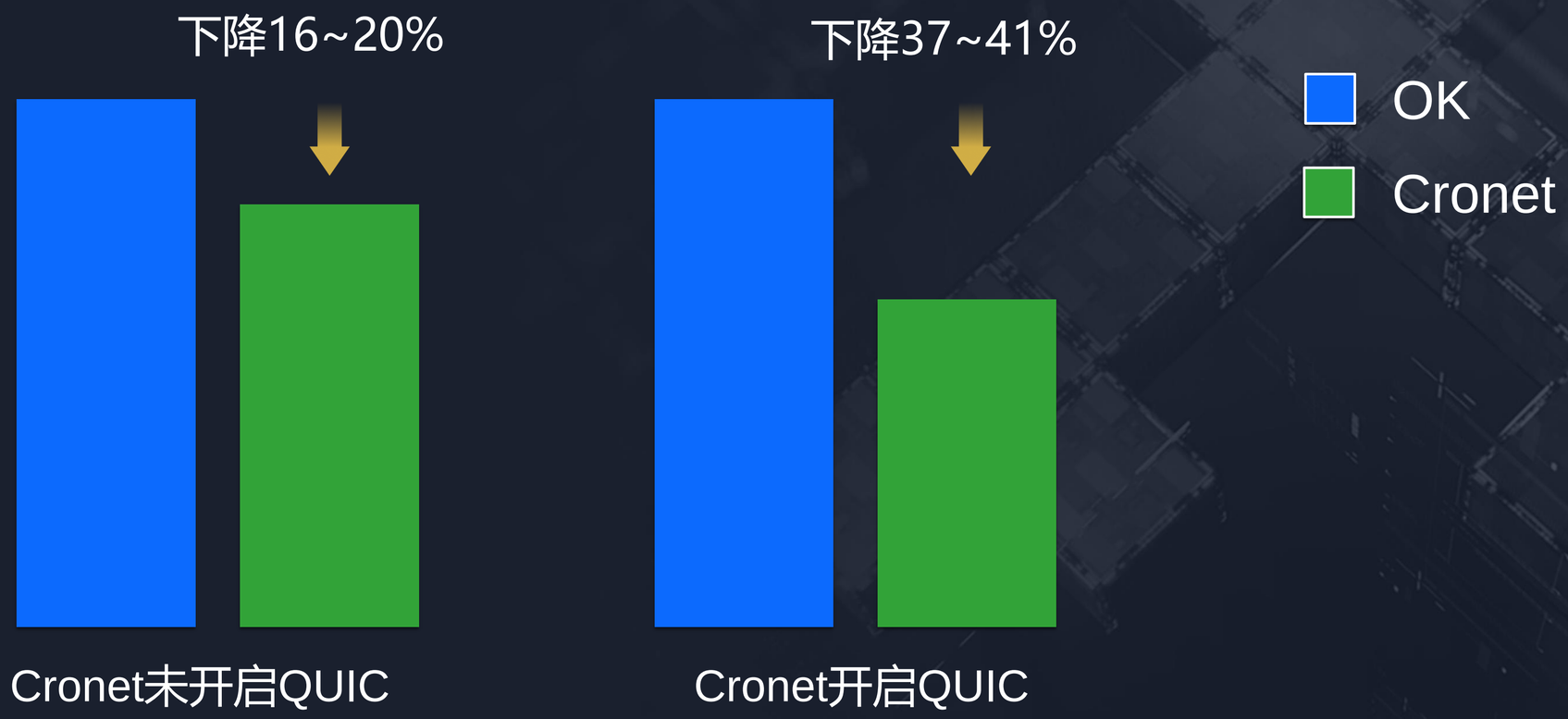
03 落地实践

04 后续规划

落地过程


官方Cronet接入
网络框架
搭建
网络组件
下沉
Android端落地
（放量中）
QUIC接入
iOS端落地
（进行中）

线上数据（Android端）



目录

01 背景介绍

02 方案设计

03 落地实践

04 后续规划

扩大落地范围

接入转向优化

Cronet接入后，“调优”之路刚刚启程：

预连接、参数调优、“竞速”优化、连接复用率、连接迁移、QUIC集群独立部署...

跟进HTTP/3

QUIC HTTP/3

- 2016 HTTP/2 over QUIC提出
- 2018 HTTP/3提出
- 2021 QUIC 标准化版本发布
- 不久的将来 HTTP/3发布

Cronet对QUIC支持

- Chromium（M72-76）默认支持gQUIC 43
- gQUIC 43以后开始支持IETF相关草案
- Chromium（M95）已默认支持 QUIC IETF RFC V1

QUIC+HTTP/3 路线

- nginx-quic合并到NGINX主分支，预计2021年底完成
- 性能优化进行中

开场白，还记得吗？

可能是你期待的

“多端适配”
- PD：XXX优化，Android上了，iOS也上下?
- 你：我更新下SDK，晚上就发布

“一致性问题”
- SDK：XXX只有Android能采集，iOS采集不了
- 你：提个bug吧，我在网络框架层修复下

“被遗忘的PC”
- PC：XXX有PC端的SDK吗?
- 你：Win、Mac、Linux都有，你要哪个?

一个顶十个
- 老板：隔壁某厂，自己实现了XXX协议，效果不错，了解下?
- 你：Cronet已经实现了，我们升级下内核就可以了

总结

1. 网络库是最应该和值得进行跨平台化改造的
2. 推荐Cronet作为跨平台网络库的首选，但有一定的门槛
3. Cronet在HTTP/2上有不错的性能优势，开启QUIC后优势进一步放大
4. 如果你还在“观望”，可以尝试先接入Cronet，再决定是否基于Cronet二次开发

InfoQ

精彩继续！
更多一线大厂前沿技术案例

北京站

AiCon
全球人工智能与机器学习技术大会

时间：2021年11月5-6日
地点：北京 · 国际会议中心

扫码查看大会详情>>

深圳站

ArchSummit
全球架构师峰会

时间：2021年11月12-13日
地点：深圳 · 大中华喜来登酒店

扫码查看大会详情>>

深圳站

GMTC
全球大前端技术大会

时间：2021年11月19-20
地点：深圳 · 大中华喜来登酒店

扫码查看大会详情>>

THANKS