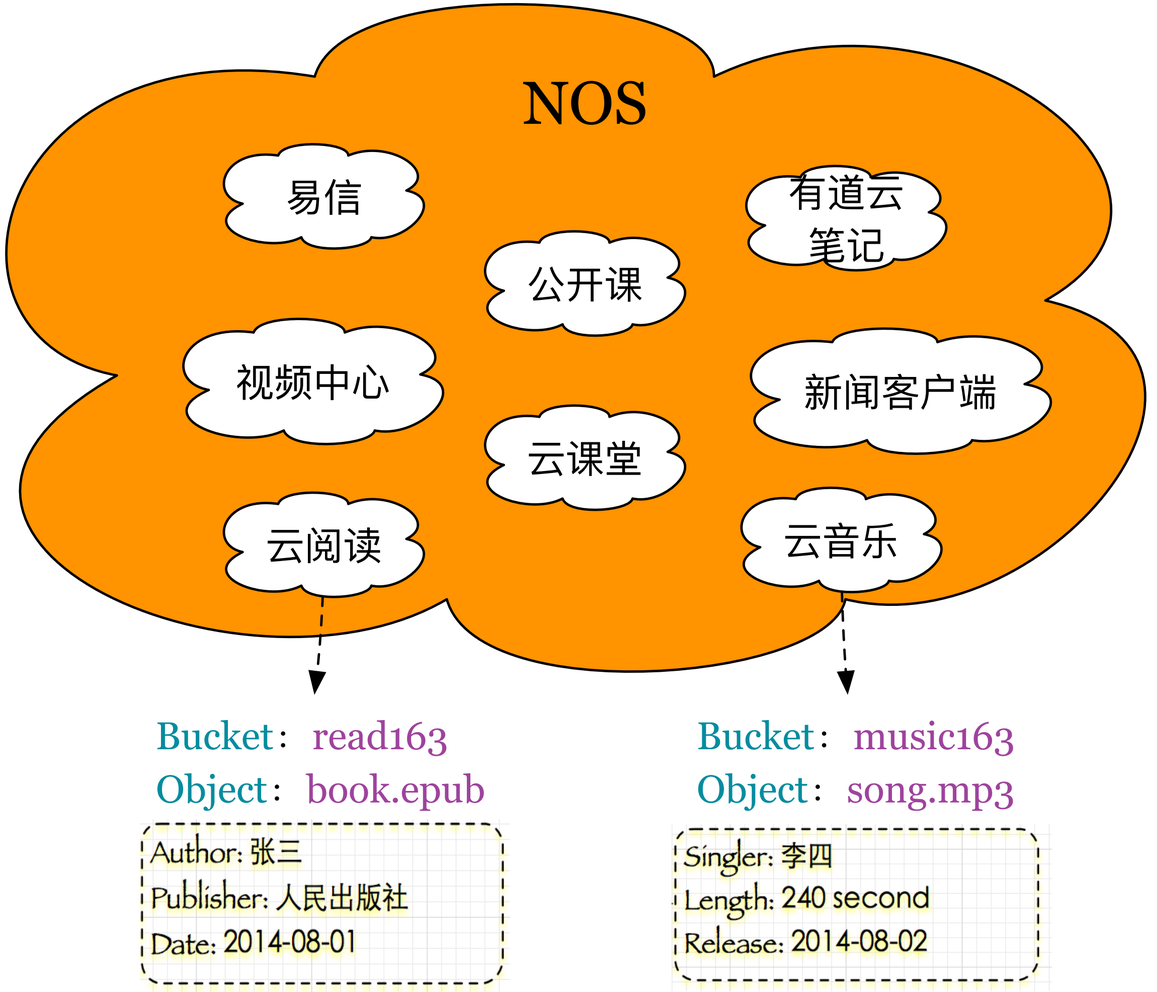
Go&网易云对象存储服务

网易－孙建良

网易云对象存储服务-NOS

- NetEase Object Store 一个海量的Key-Value Blob 存储服务
  - Key：最大支持1K字节
  - Value：最大支持1TB，图片、音视频、log、数据库备份等静态资源
  - Attribute：对象属性，最多10个键值对
  - Bucket：命名空间
- NOS（Netease Object Storage）致力于提供最优质的对象存储以及基于存储的富媒体和上下行加速服务，一站式解决移动互联网时代非结构数据管理难题，助力产品方实现最佳用户体验。



网易考拉海购
kaola.com

网易云课堂
study.163.com

LOFTER
专注兴趣 分享创作

网易云音乐
听 见 · 好 时 光！

網易公开课
传播属于全人类的知识和智慧

网易青果

网易云信
NETEASE.IM
真正稳定的IM云服务

网易蜂巢
为开发者打造的DOCKER容器云

网易视频云

网易七鱼
七鱼，改变客服

有道云笔记
note.youdao.com

BoBo
bobo.com
网 易 旗 下 视 频 社 区

易信
让沟通更有趣

网易
严选

花田
我们相遇的地方！

网易云阅读

163 网易免费邮
mail.163.com
中文邮箱第一品牌

网易漫画
manhua.163.com

美聊

网易视频
V.163.COM

与go结缘

- 2012 实时负载统计隔离系统
- 2014 图片处理系统
- 2015 直传加速服务系统
- 2015 新一代存储引擎系统

技术选型

- 核心诉求－高并发
  - Thread-Based Concurrency - tomcat
  - Event-Driven Concurrency - libevent
  - Staged Event-Driven Concurrency(SEDA) - netty
- 对比
  - netty
    - 异步编程框架，核心上还是无法避免异步编程的复杂性
    - 事件处理代码段小精干，尽可能避免阻塞及拖延
  - go
    - RunTime 搞定网络层次异步，上层只需要使用同步编程逻辑
    - 轻量级的携程

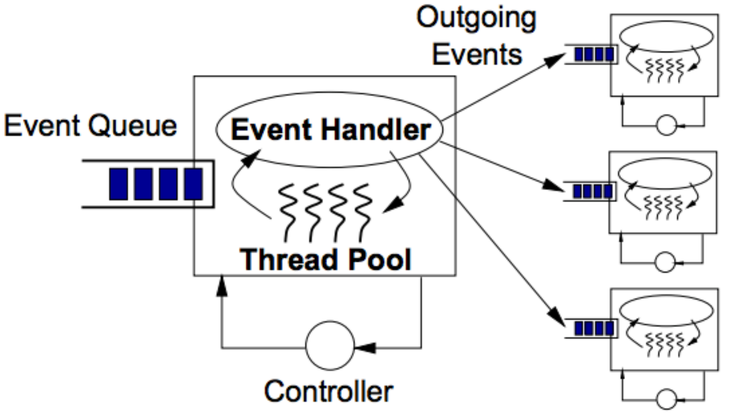
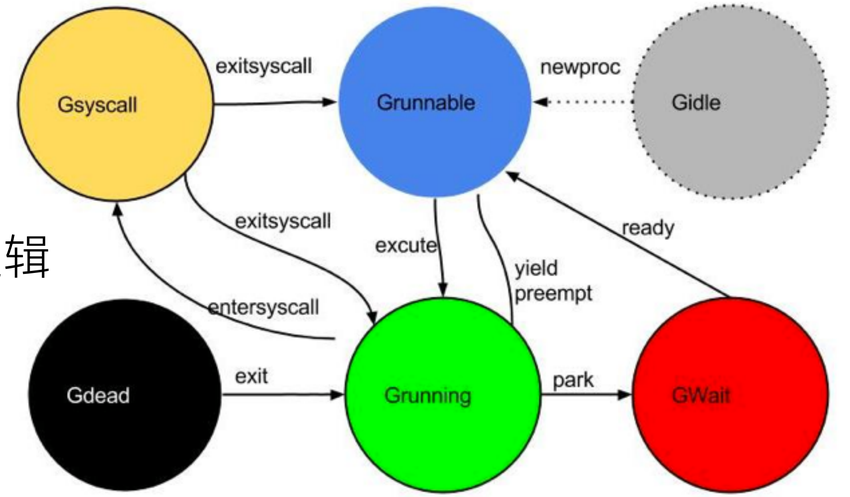
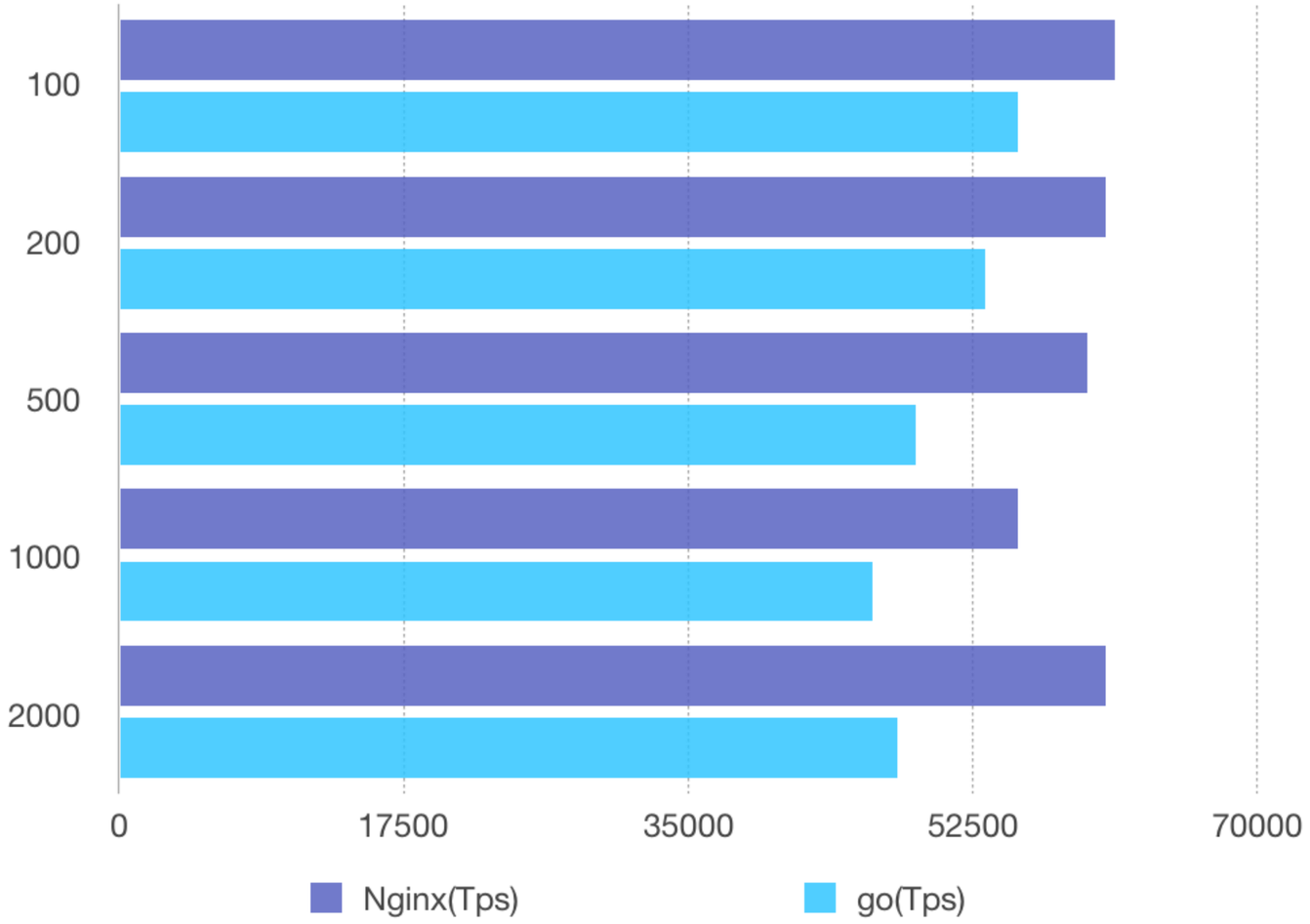


Figure 6: **A SEDA Stage:** *A stage consists of an* incoming event queue, *a* thread pool, *and an application-supplied* event handler. *The stage's operation is managed by the* controller, *which adjusts resource allocations and scheduling dynamically.*



Go vs Nignx Lua

- 测试方法
  - 使用 ab测试不同并发场景下nginx和golang http 服务的性能，测试数据大小512Byte。
  - 测试命令示例：ab -n 1000000 -c 5000 -k “http://127.0.0.1:8081/512b“
- 详细测试内容见：Nginx And Go HTTP并发测试



We Choose Go

Go在广域网直传加速服务上的应用

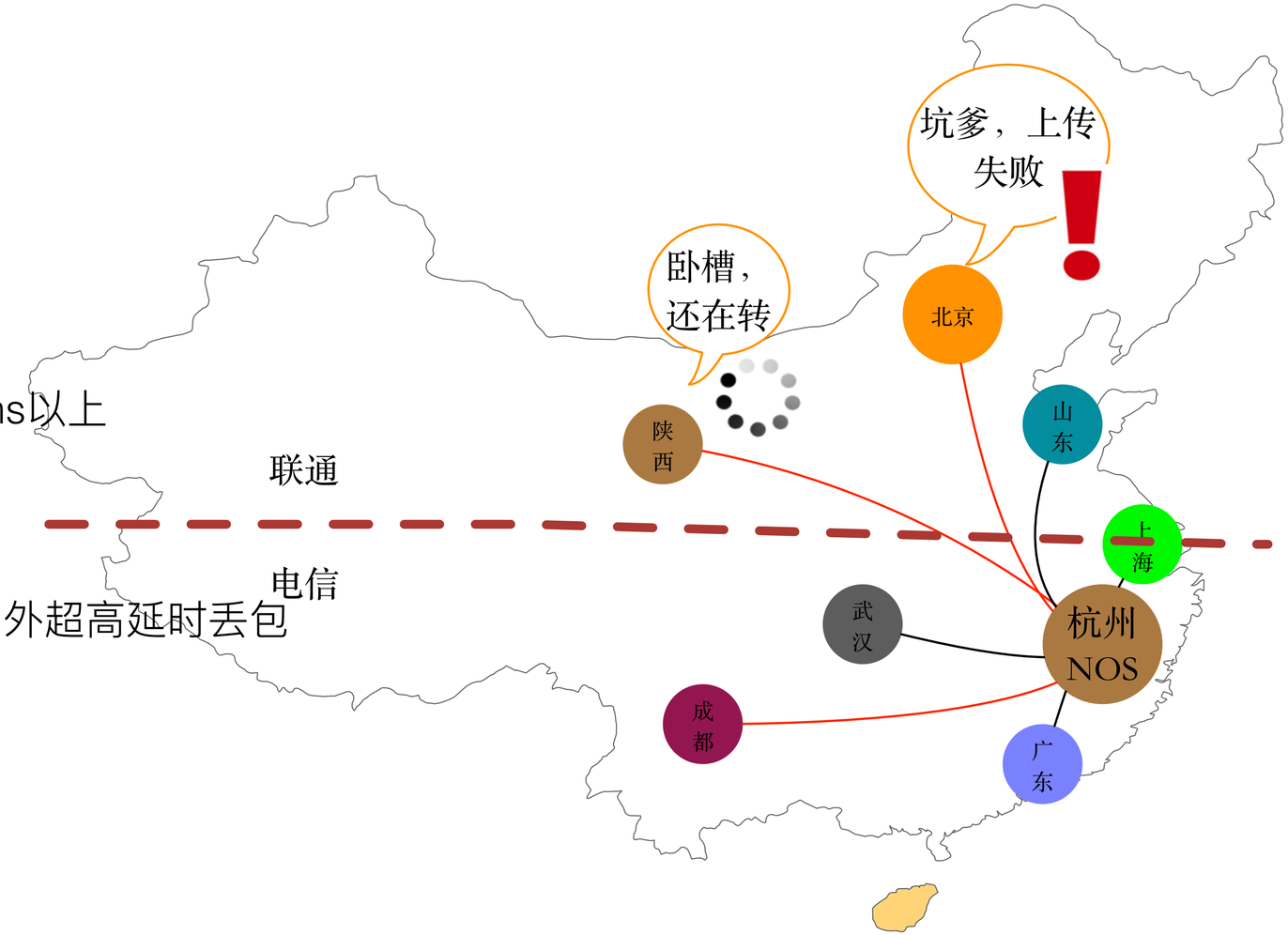
- 移动广域网网络环境
- 核心问题
- 解决思路
- 上传协议
- 移动端上传优化
- 广域网TCP、HTTP优化
- 系统架构
- 路由优化系统
- 全球网络布局及加速效果

Part1

- 移动广域网网络环境
- 核心问题
- 解决思路

移动广域网网络环境

- 丢包
  - 移动网络环境、广域网
- 延时
  - 华北、西北、西南延时30～50ms，夜间100ms以上
- 国内外络环境
  - 电信、联通南北分隔；教育网，小运营商；国外超高延时丢包
- 导致
  - 上传失败率高
  - 上传速度差



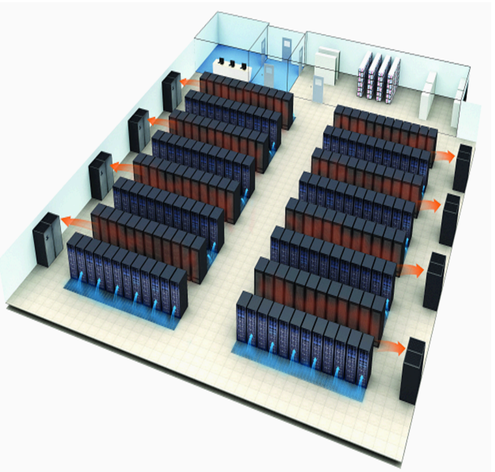
核心问题

客户端

High loss

基站

High rtt



数据中心

TCP Connection： High loss rate + High RTT

- TCP 拥塞控制算法
  - high Loss Rate => smaller throughput
  - high RTT => smaller throughput

注：附录1

$$throughput \approx \frac{1}{RTT}\sqrt{\frac{1.5}{loss}}$$

解决思路

- 将边缘节点部署到离用户最近的地方
  - 前半段
  - 后半段

$$throughput \approx \min\left\{\frac{1}{RTT_1}\sqrt{\frac{1.5}{loss_1}}, \frac{1}{RTT_2}\sqrt{\frac{1.5}{loss_2}}\right\}$$

注：附录1

Part2

- 上传协议
- 移动端上传优化
- 广域网TCP、HTTP优化

上传协议

- 传统标准OSS上传协议
  - 分块上传，最小分块5M
  - 专为服务端设计
- NOS直传协议
  - 支持任意大小分片
  - 分片append协议

```
POST /${bucketName}/${objectName}?offset=${Offset}&complete=${Complete}&context={context}&version=1.0 HTTP/1.1
Content-Length: ${length}
Content-Type: ${contentType}
x-nos-token: ${token}

<data of body>
```

bucketName/objectName: 上传后的文件名

offset：上传数据在整个文件中的便宜

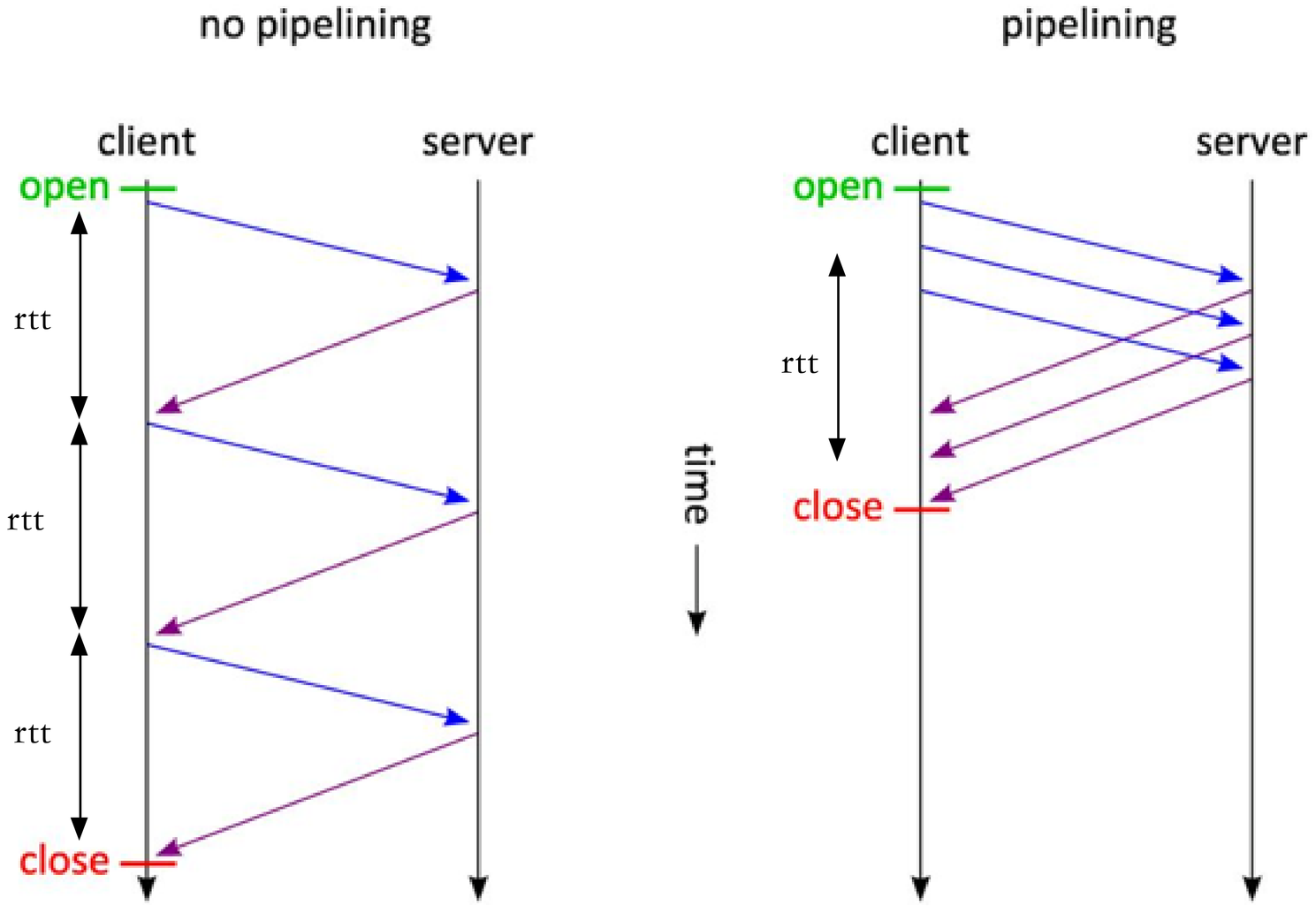
x-nos-token:上传令牌

context：服务端返回标识唯一标识一次上场

complete：标示是否为最后一个分片

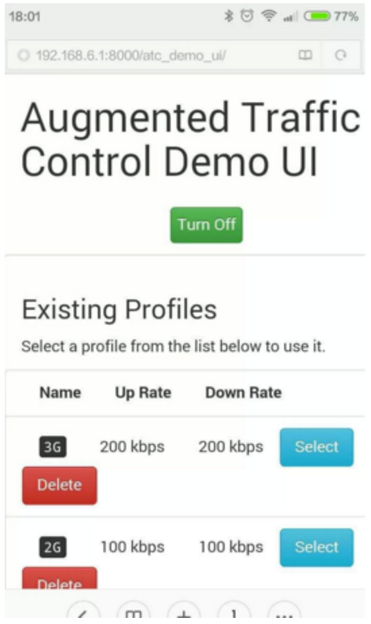
移动端上传优化

- HTTP 1.1 PipeLine
  - 分片拆分
    - 切小分片，提高上传成功率
    - 导致多次交互
  - non pipeling
    - 延迟高表现很差
    - 2G 650ms；3G 100ms；4G：25r
  - pipeling：充分利用网络广域网带宽



PipeLine测试

- 测试方式
  - 树莓派+ Facebook Augmented Traffic Control
  - 测试数据：2G/3G/4G 带宽 延迟（参见附录:2)
  - 国内外



国内

网络环境	上行带宽	下行带宽	上行延迟	下行延迟	丢包率
2G网络	32kbps	35kbps	680ms	680ms	2%
3G网络	1Mbps	5Mbps	130ms	130ms	0
4G网络	15Mbps	50Mbps	55ms	40ms	0

国外

网络环境	上行带宽	下行带宽	上行延迟	下行延迟	丢包率
2G网络	32kbps	35kbps	867ms	867ms	2%
3G网络	1Mbps	5Mbps	317ms	317ms	0
4G网络	15Mbps	50Mbps	242ms	227ms	0

PipeLine效果

国内

网络环境	上传分块大小	非pipeline上传速率	pipeline上传速率	pipeline速率提高比率
2G网络	4KB	1.84KB/s	2.82KB/s	53.26%
3G网络	64KB	38.75KB/s	85.46KB/s	120.54%
3G网络	128KB	49.90KB/s	94.03KB/s	88.44%
4G网络	512KB	398.48KB/s	496.53KB/s	24.61%

国外

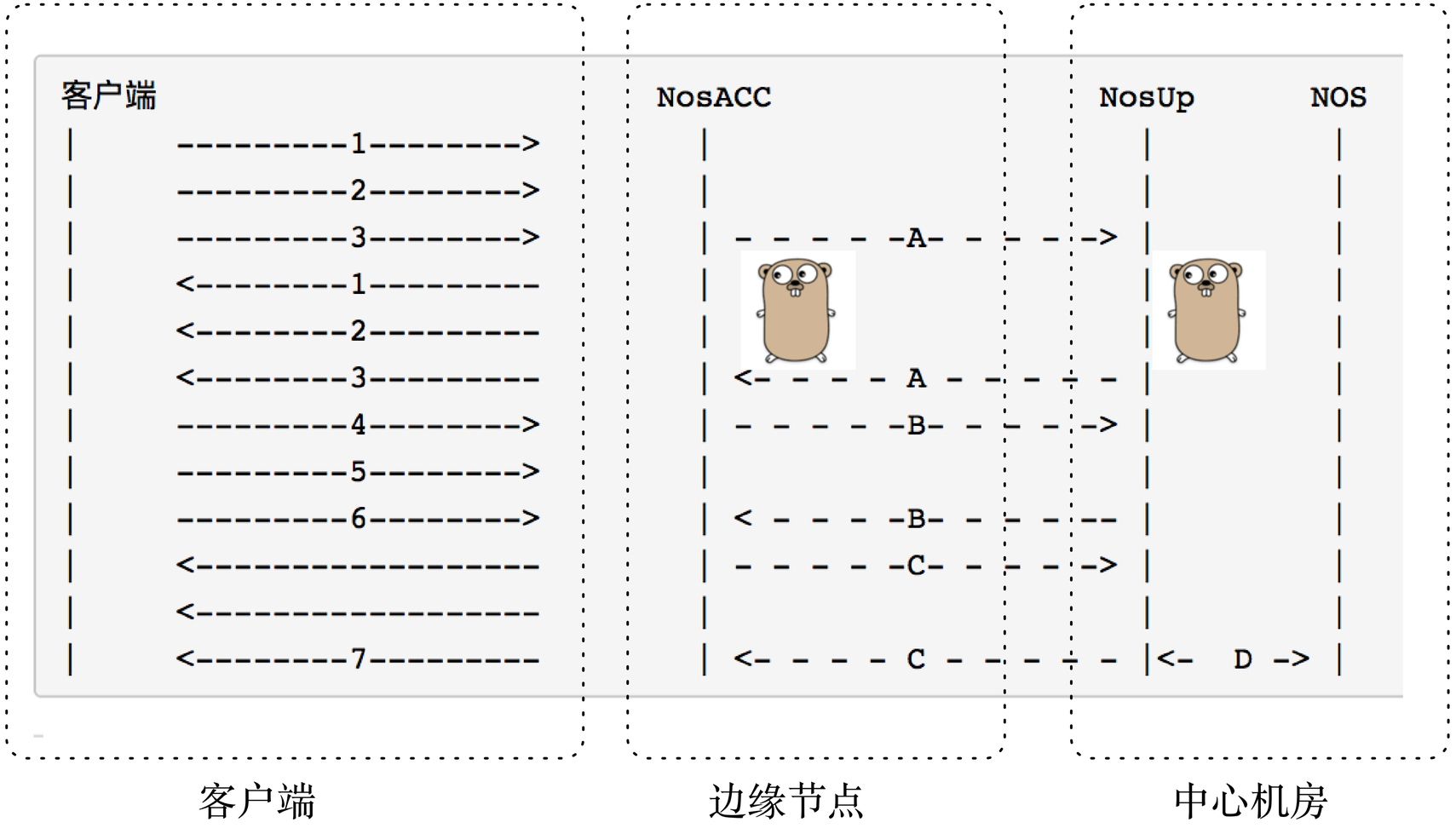
网络环境	上传分块大小	非pipeline上传速率	pipeline上传速率	pipeline速率提高比率
2G网络	4KB	1.32KB/s	2.72KB/s	106.06%
3G网络	64KB	17.27KB/s	73.14KB/s	323.51%
3G网络	128KB	25.82KB/s	73.86KB/s	186.06%
4G网络	128KB	82.51KB/s	157.82KB/s	91.27%
4G网络	512KB	123.20KB/s	158.83KB/s	28.92%

广域网优化化

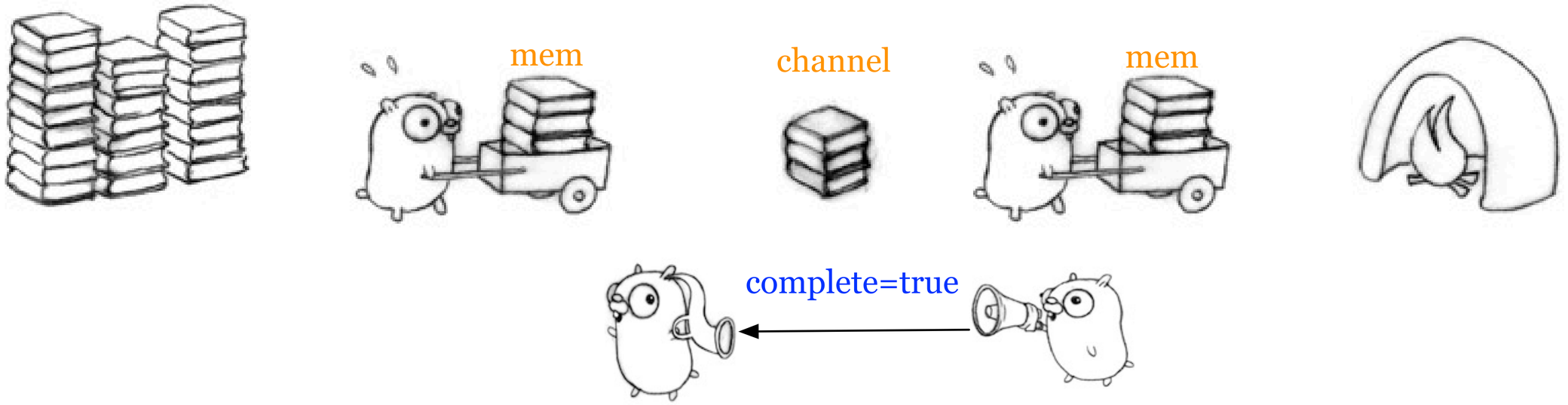
- 长连接池：避免慢启动、提升拥塞窗口
- 协议层并发
- TCP调优
- HTTP调优

协议层并发

- 客户端
- NosAcc
- NOS



并发模型


mem
channel
mem
complete=true
mem
channel
mem
Request
NosACC
NOS数据中心

内存池

- go 1.3 官方 sync.Pool包
  - 对象生命周期为2次gc之间
  - runtime支持，锁有一定的优化
- leaky_buffer bytepool.go
  - 固定最大资源池
  - 超过maxSize依赖垃圾回收

```go
type BytePool struct {
    c chan []byte
    w int
}

func NewBytePool(maxSize int, width int) (bp *BytePool) {
    return &BytePool{
        c: make(chan []byte, maxSize),
        w: width,
    }
}

func (bp *BytePool) Get() (b []byte) {
    select {
    case b = <-bp.c:
        b = make([]byte, bp.w)
    }
    return
}
func (bp *BytePool) Put(b []byte) {
    select {
    case bp.c <- b:
    default:
    }
}
```

TCP层优化

- **tcp_wmem**

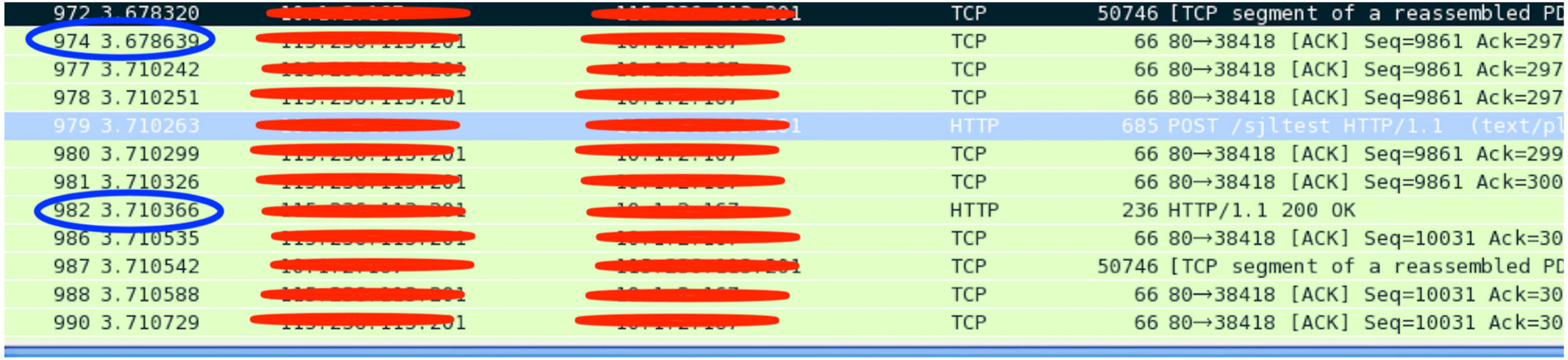
```
wanproxy@hangyan-nosupload2:~$ cat /proc/sys/net/ipv4/tcp_wmem

4096 65536 4194340
```

坑

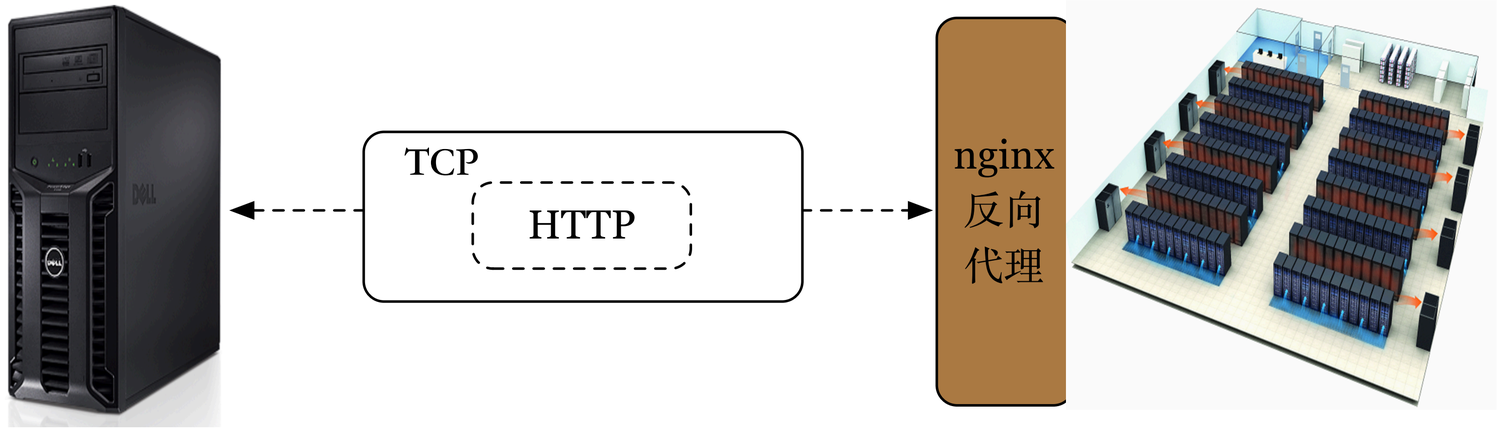
```
tcpCon.SetWriteBuffer(t.TCPWriteBufferSize);
```

- **tcp_slow_start_after_idle**
- **net.ipv4.tcp_sack**



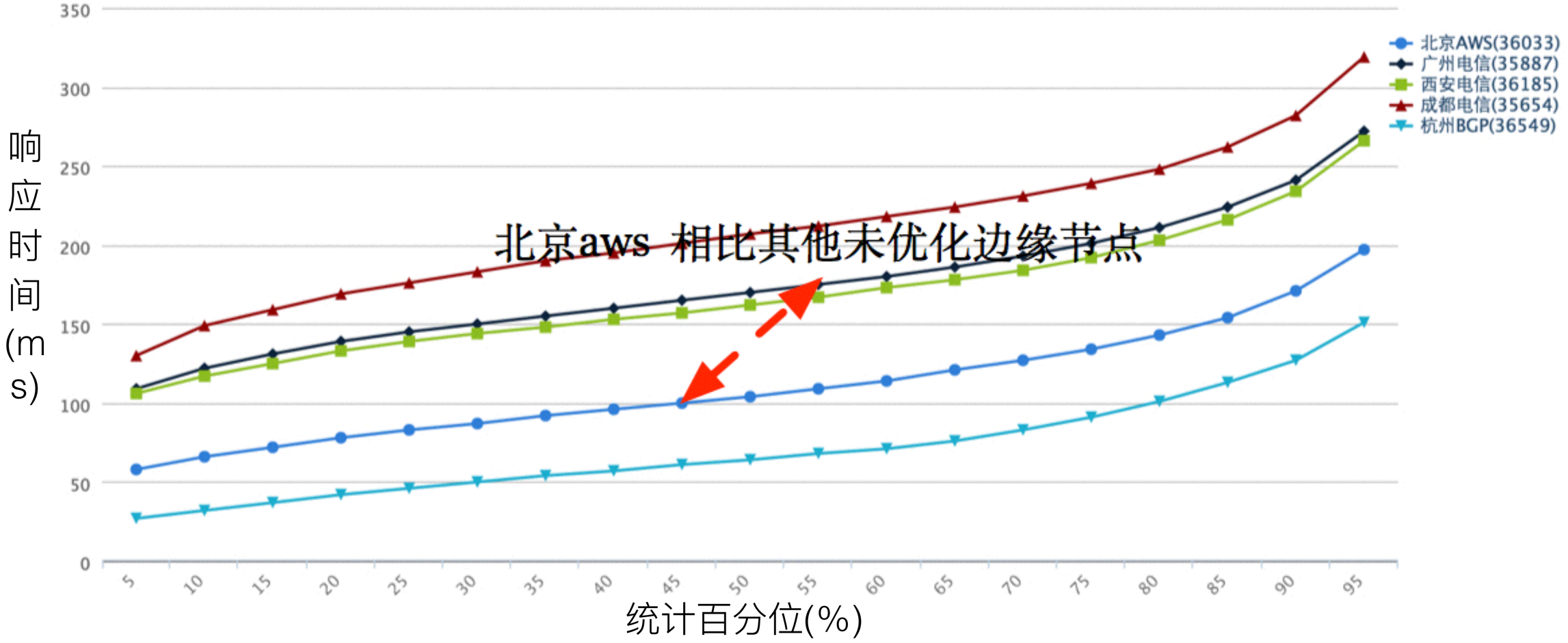
HTTP层优化

- 边缘节点与中心机房走HTTP
- keepalive_requests：默认100
- keepalive_timeout： 默认75s
  - 刚开始低负载情况下开大，避免刚刚打开的拥塞窗口的HTTP长连接流失



网络优化效果

- 边缘节点到杭州中心机房响应时间

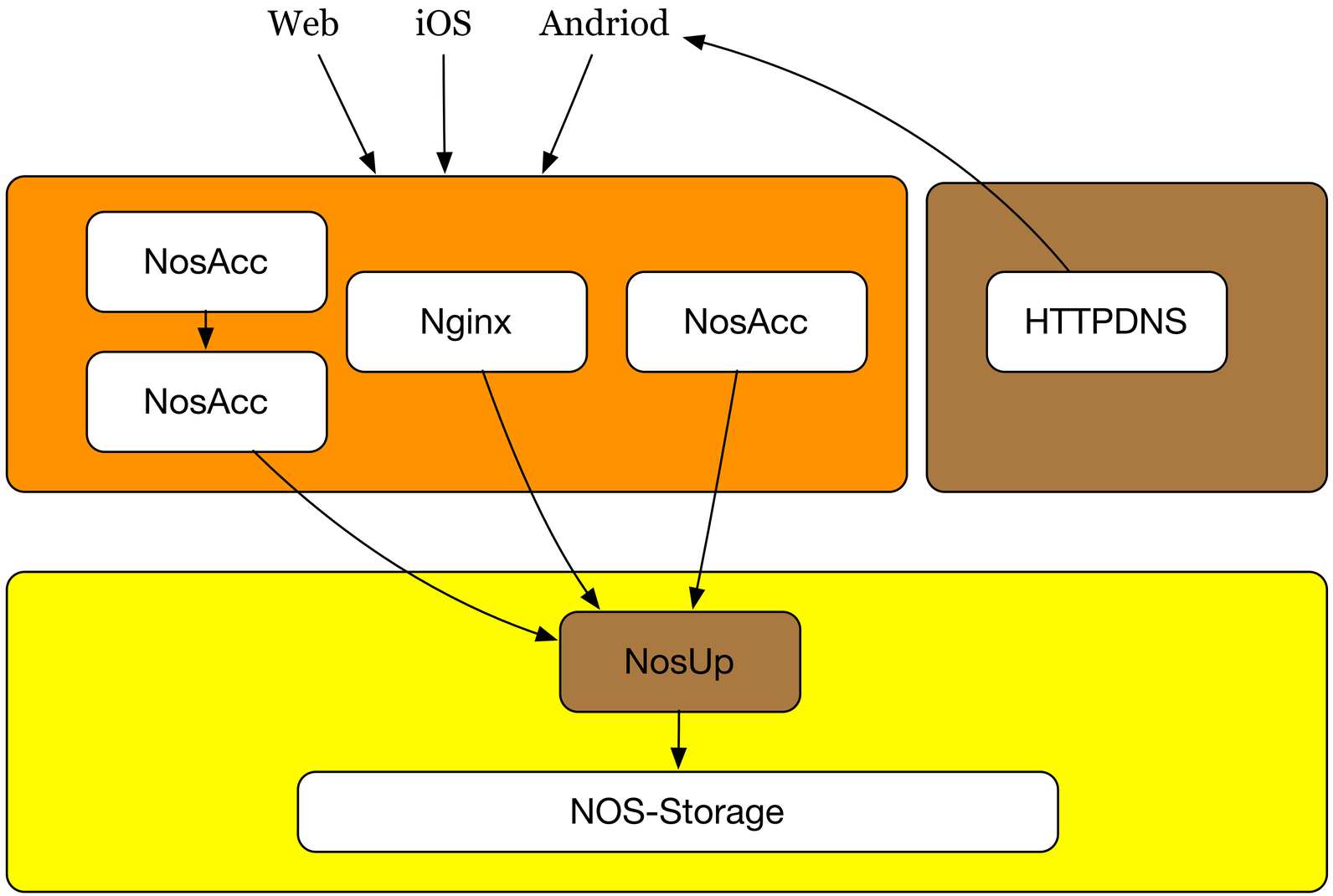


Part3

- 系统架构
- 路由优化系统
- 全球网络布局及加速效果

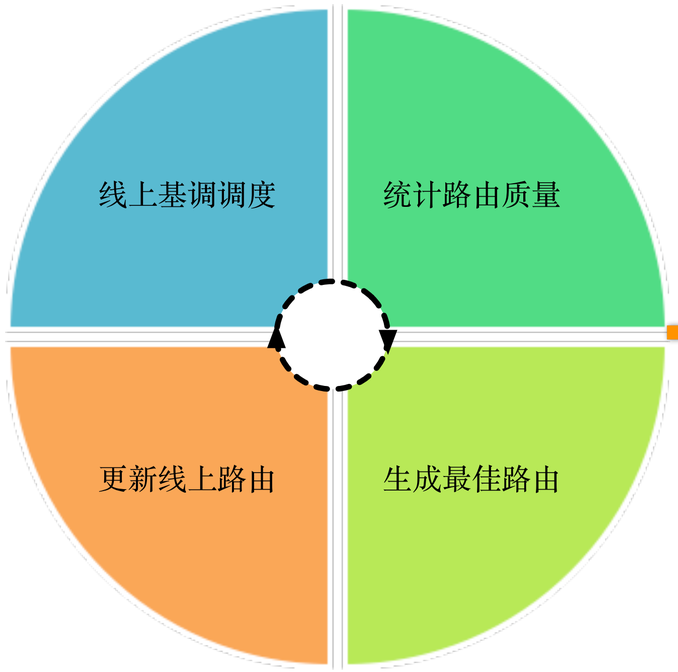
系统构架

- Client
  - Web、iOS、Android
- NosACC、Nignx
  - 加速接入
- NosUP
  - 分片数据缓存
  - 断点信息维护等
- HTTPDNS
  - 用户请求调度

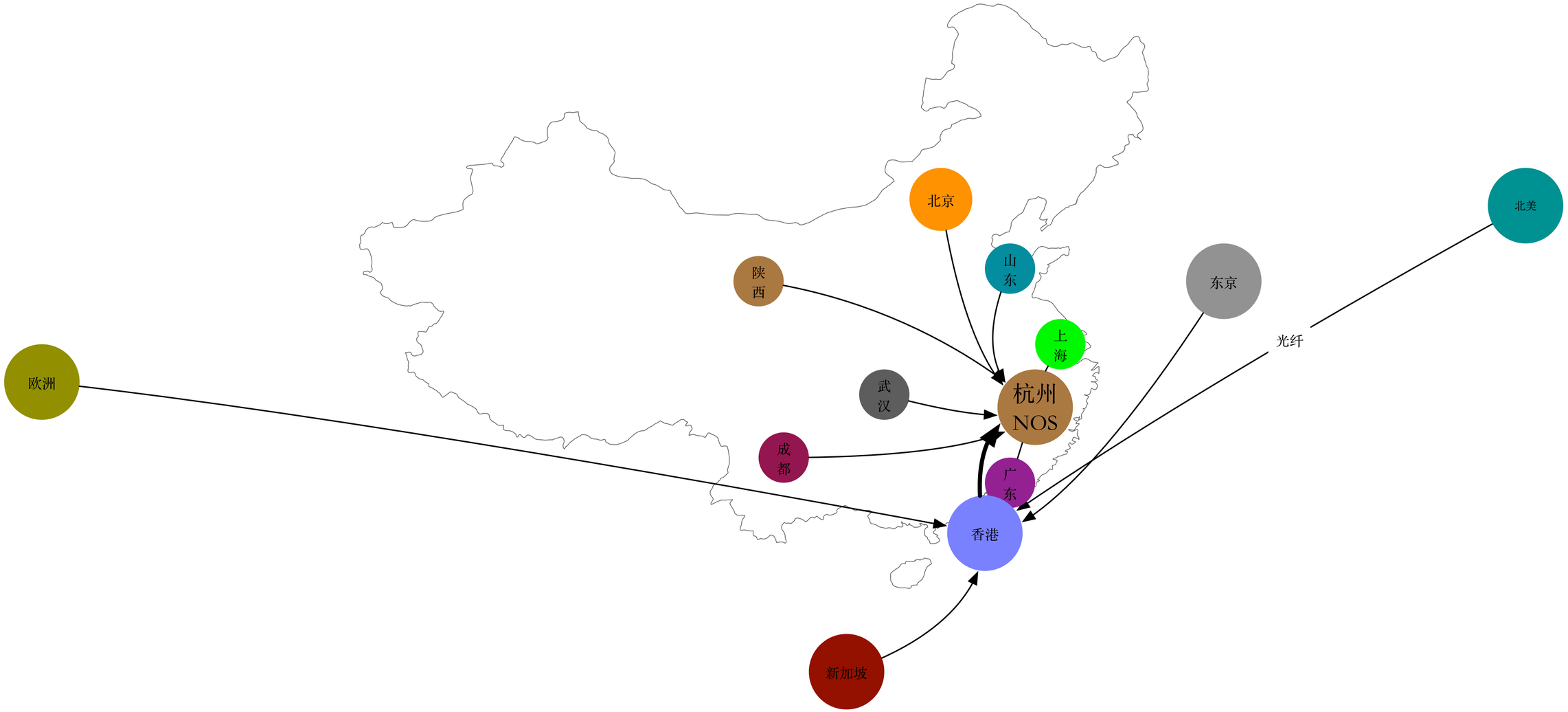


上传线路优化系统

- 实现上传路由的实时评估和定期优化
- 动态跟踪广域网路由变化，使得HTTPDNS调度规则最优
- 自动生成国内外路由并每周更新



全球上传加速网络部署



加速效果

国家	接入节点	加速前(KB/S)	加速后(KB/S)	加速比(%)
英国	爱尔兰	18	110	611
新加坡	新加坡	41	231	563
西班牙	爱尔兰	20	88	440
加拿大	美国西部	28	112	400
日本	东京	59	236	400
法国	爱尔兰	18	68	377
马来西亚	新加坡	19	70	368
泰国	新加坡	23	83	361
德国	爱尔兰	24	75	312
美国	美国西部	31	85	274
意大利	爱尔兰	26	67	257
澳大利亚	新加坡	22	39	177
韩国	上海电信	77	109	141

总体提升：40%🚀

省份	城市	运营商	加速节点	加速比%
广东	阳江	联通	广州电信	123
云南	昆明	联通	成都电信	118
北京	北京	电信/联通	北京移动	90
广东	广州	电信	广州电信	85
广东	肇庆	电信	上海移动	83
江苏	南京	联通	上海移动	77
广东	广州	教育网	广州电信	70
北京	北京	电信	北京AWS	66
北京	北京	联通	北京AWS	64
陕西	西安	联通	西安电信	62

用户接入

NOS iOS SDK

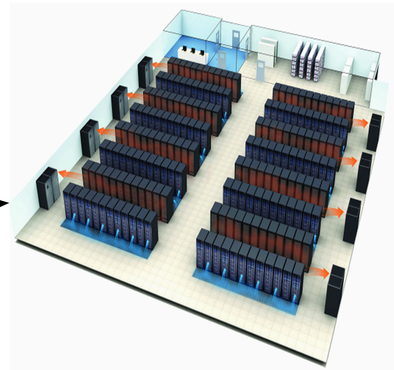
NOS Android SDK

产品方业务服务器

颁发上传token(令牌)

NOS Web SDK

NOS上传加速网络



NOS中心机房

其它

- HTTPDNS的国内外部署
- 国外线路专线高可用
- 国外多级加速的方案
- 基于NCDN以及NCAN多机房上传下载高可用

opening soon

Thank You & QA

github-page

附录

1. Velocity 2011:MobilePerformanceVelocity
2. Performance Measurements of TD-LTE, WiMAX and 3G Systems.pdf
3. http://facebook.github.io/augmented-traffic-control/
4. Rob Pike :Concurrency is not Parallelism
5. SPDY configuration: tcp_slow_start_after_idle
6. SetReadBuffer/SetWriteBuffer have no effect on TCP connection