



蘑菇街高并发多终端无线网关实践

美丽联合 - 无线应用 - 王子默



三 概述



架构演进

问题
分层
选型



技术实现

接入层
路由层
聚合层



挑战与规划

接入层改造
下行通道
去中心化



MWPP出现之前

HTTP + HTTPS

✗ 服务单一扩展性差

服务发布/升级困难
噪音太多无法专注业务开发

✗ 无法多路复用

链接使用不充分
平均RT 650ms

✗ 数据安全无法保证

请求数据篡改
下发配置不安全

✗ 建连成功率低

弱网下https建连成功率低
大部分超时出现在https建连



MWP

提供一个双工的安全可靠的云服务



服务标准化

标准的输入输出
多端适配
服务治理



高效安全通道

0-RTT
算法优化

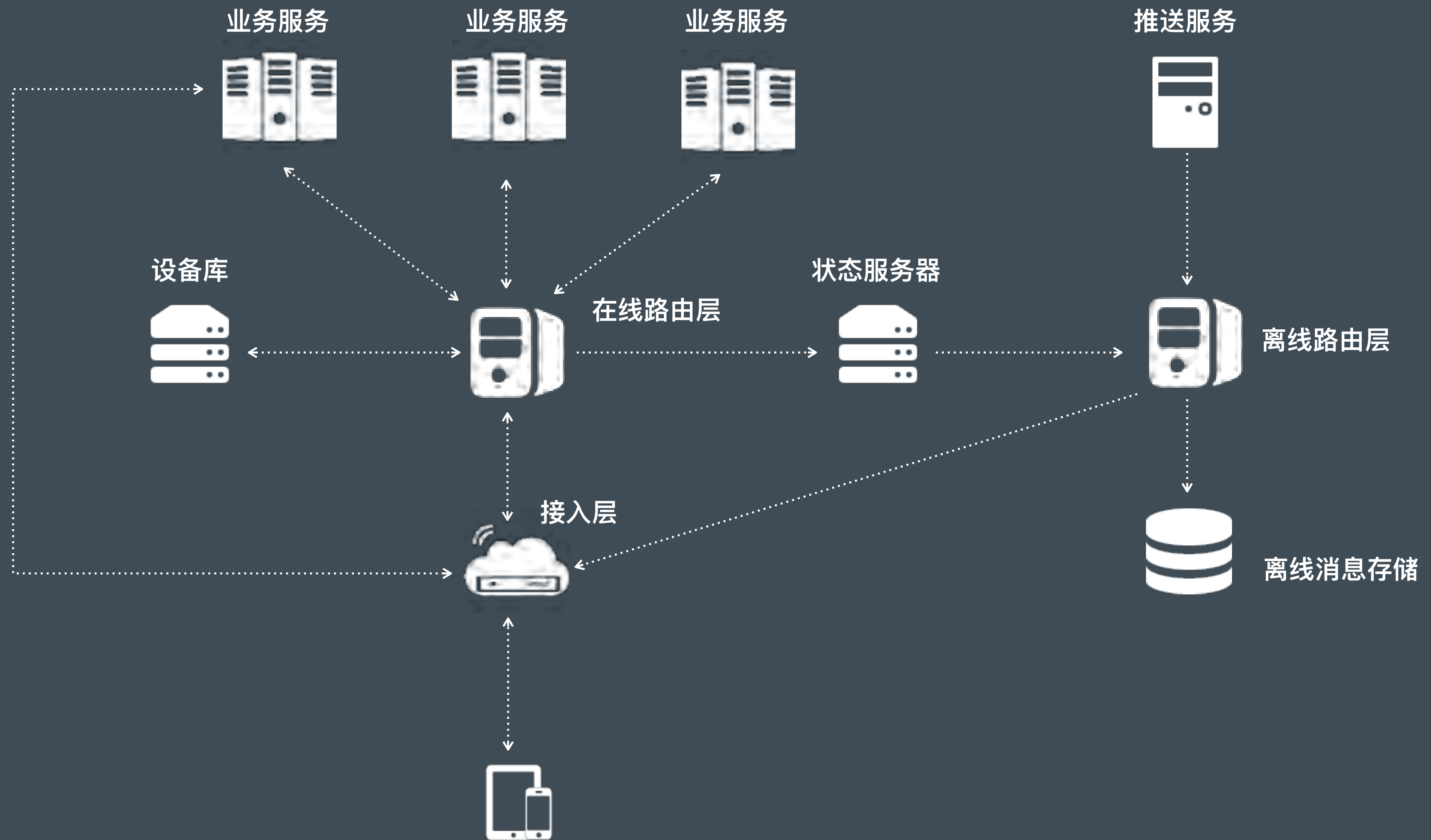


聚合服务

前后端分离
提升用户体验



三 MWP整体架构



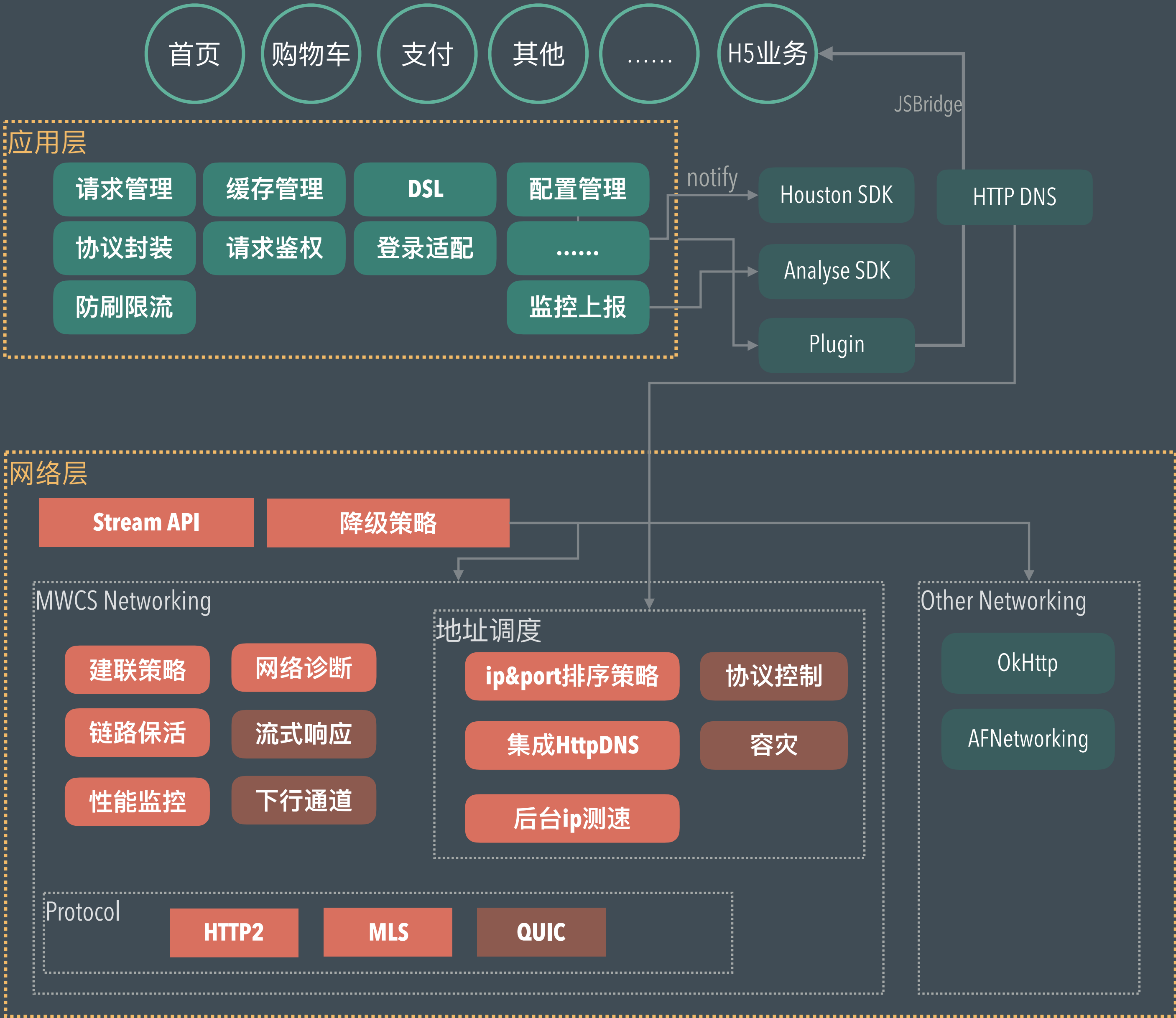
三 MWP客户端接入层

应用层

- ✓ 协议封装
- ✓ 平台化服务
- ✓ 横向拓展

网络层

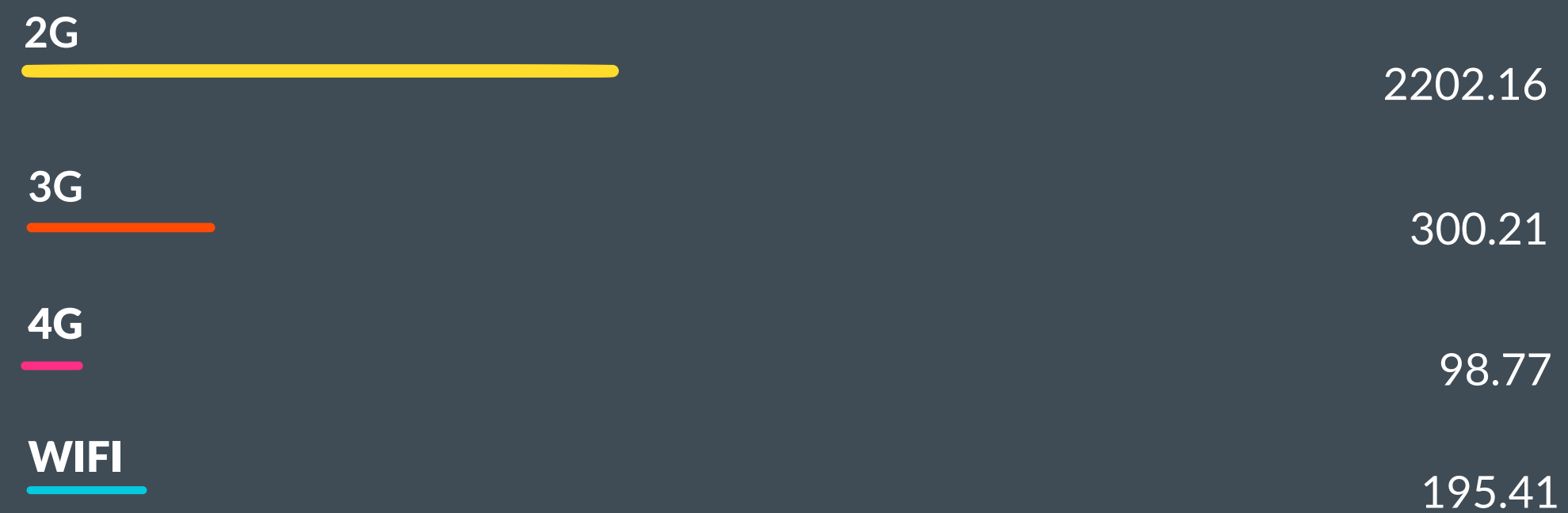
- ✓ 链接管理
- ✓ 地址调度
- ✓ 建链优化



安全链路

无线环境2-RTT是不能接受

http建连耗时



https建连耗时



三 MWP网络接入层

MLS

TLS1.3 {0-RTT}



证书预埋

ECDH



算法选择

AEAD, 向前加密



session
reuse

session加密下发

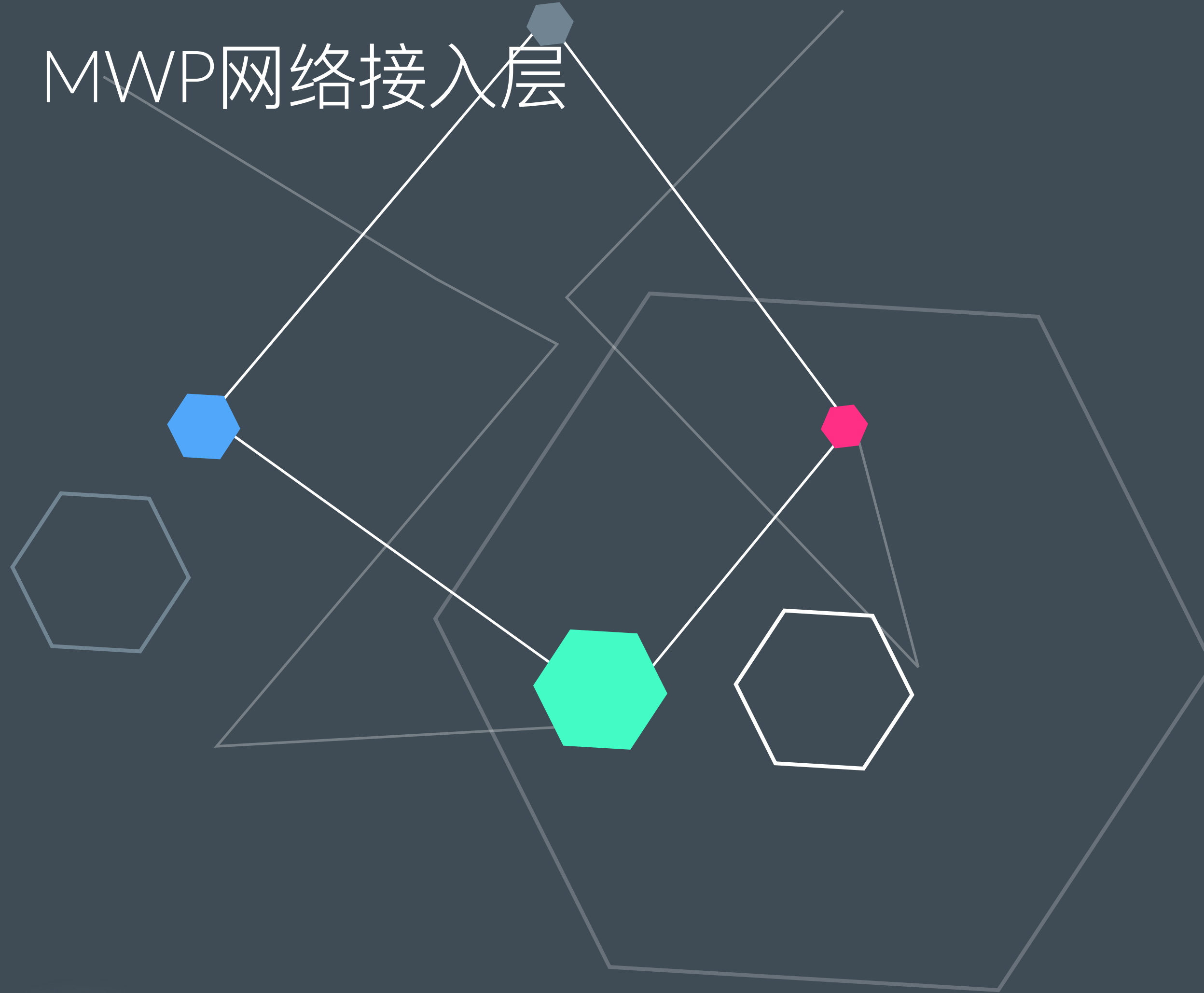


加密流程

协商非对称, 传输对称



三 MWP网络接入层



MLS vs TLS

建连耗时提升百分比

43%

2G

26%

3G

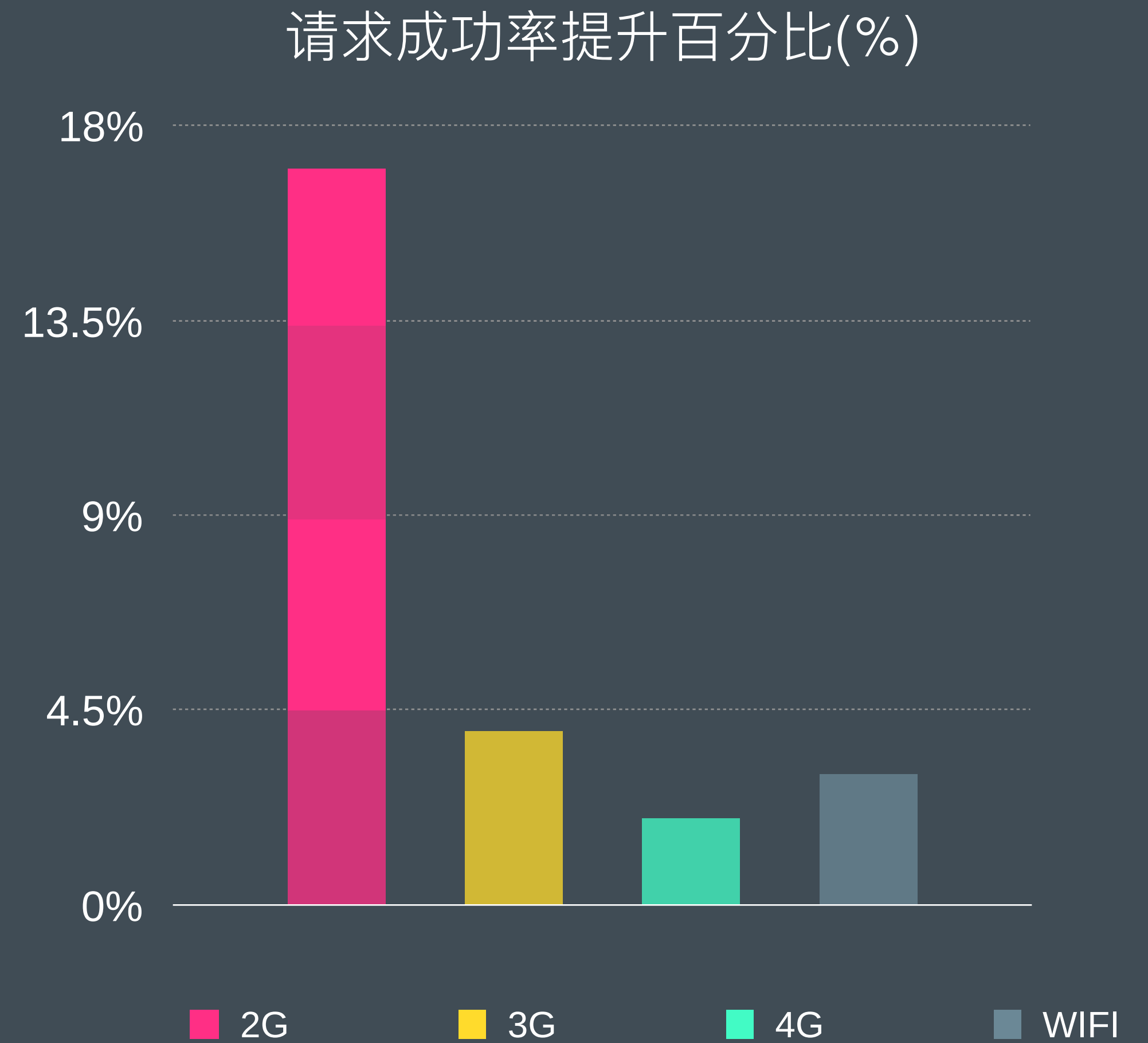
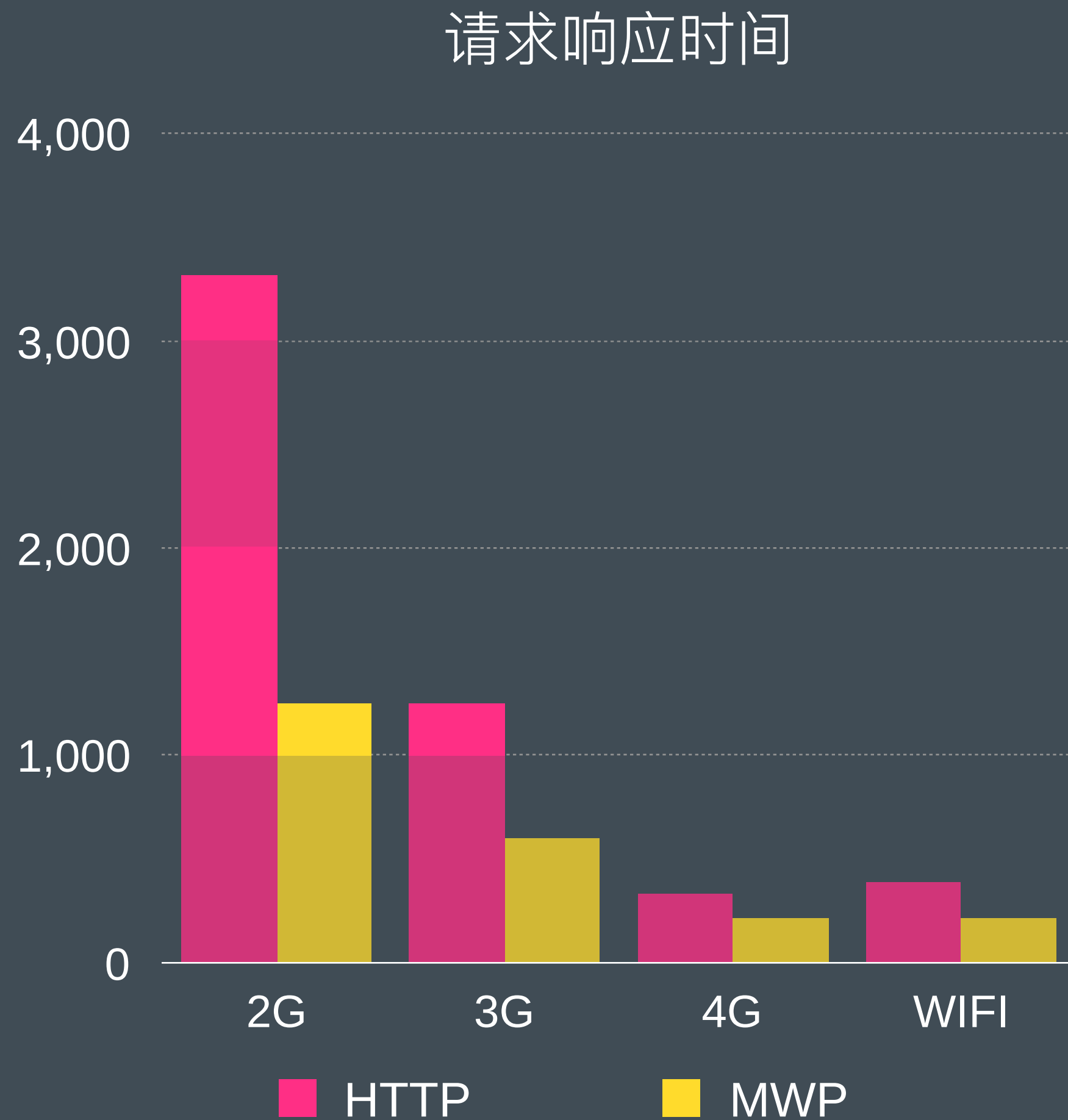
57%

4G

44%

wifi

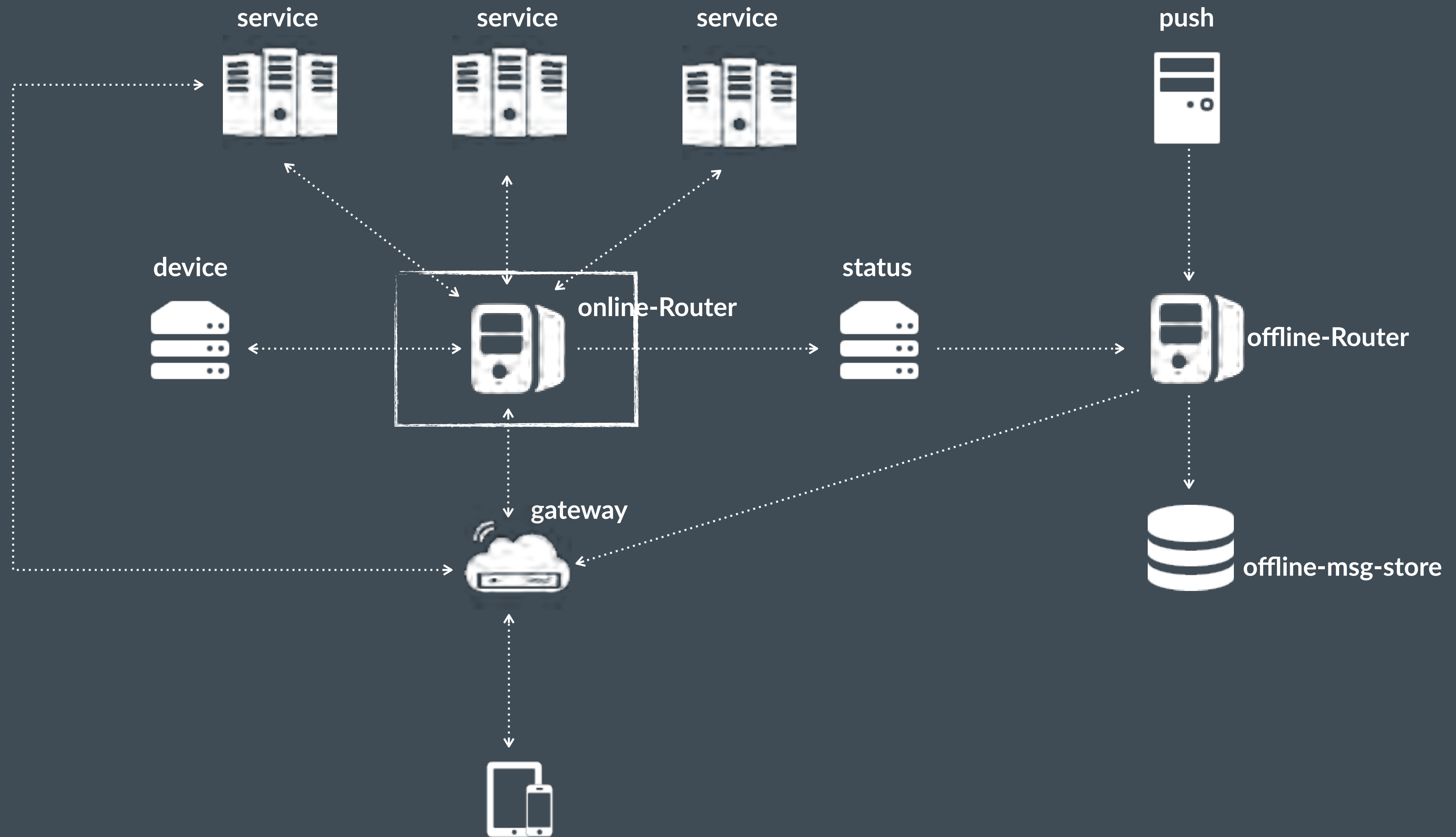
三 MWP通讯协议 vs HTTP



此外缓存包头也为客户端节省了36%的流量



三 MWP整体架构



MWP-Router

Servlet3 + async RPC client

泛化调用

服务治理

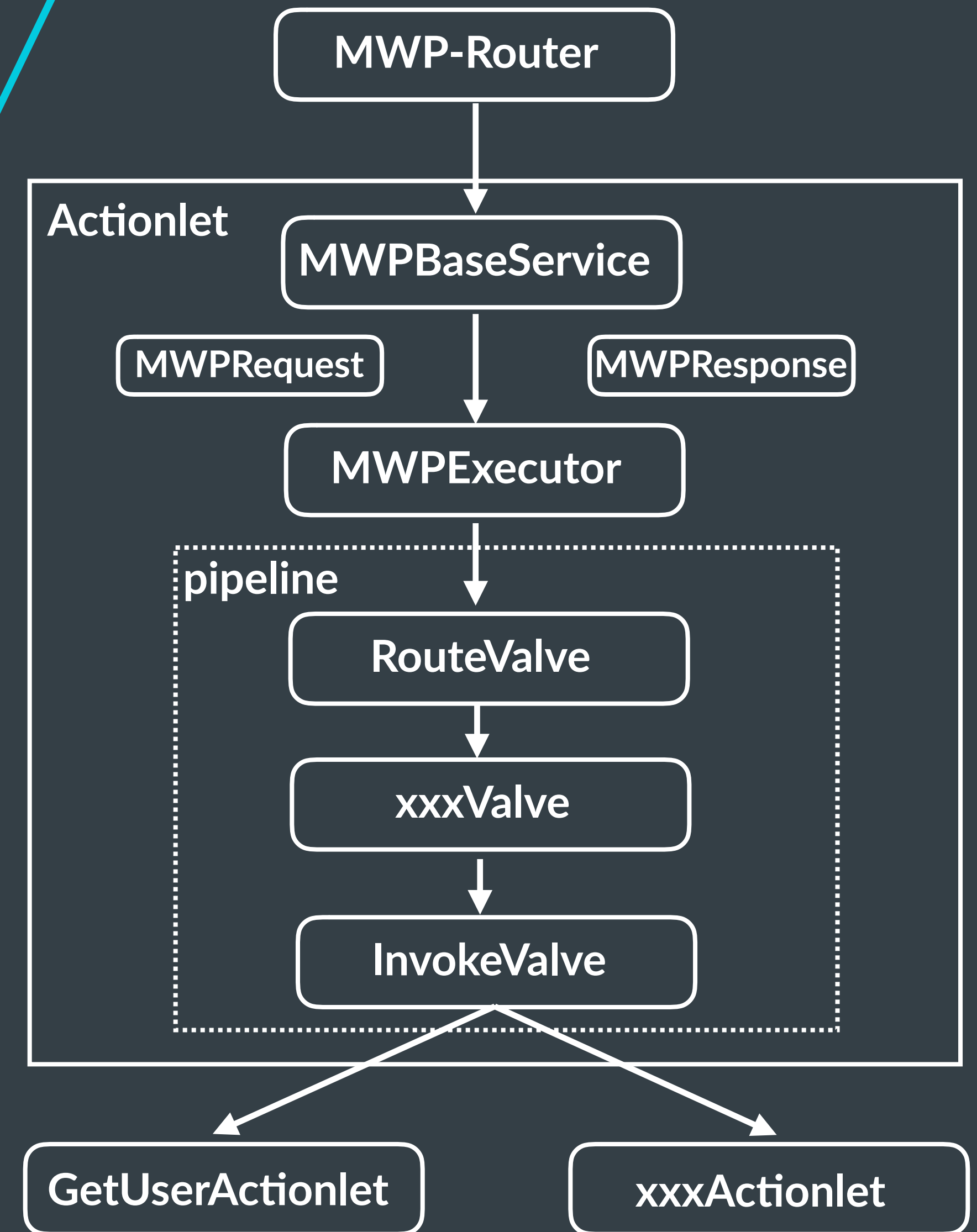
服务聚合



三 MWP路由层泛化调用

Actionlet

- ✓ 服务标准化, 专注业务代码
- ✓ 多端渲染
- ✓ @MWPApi(name=xx.xx.xx,version=xx)



三 MWP路由层服务治理

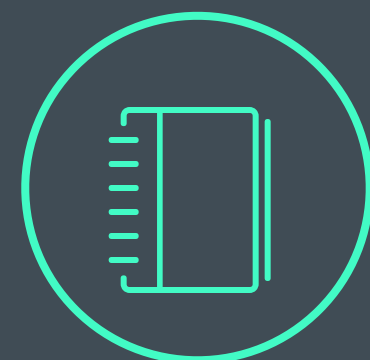
App管理

包括管理App的基础信息和版本信息，最重要的维护订阅的应用列表



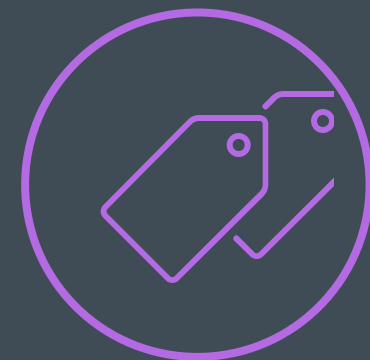
应用管理

包括管理应用的基础信息和版本信息，最重要的维护提供api列表及api级别的配置



别名管理

主要用于接口迁移，分流等场景



DSL管理

也就是服务聚合脚本的管理



监控细项

包含appkey, appinfo, version, api, result等级别的监控细项



配置下发

通过configcenter实时下发配置



三 MWP路由层服务聚合

聚合层出现之前

? 订单状态
渲染数据的转换

? 商品详情页
多接口的合并



三 MWP路由层服务聚合

配置

code

```
1 def flushMap = [:]
2 def $m1 = $payloadMap['mwp.application.Actionlet1@1']
3 def $m2 = $payloadMap['mwp.application.Actionlet2@1']
4 def $m3 = $payloadMap['mwp.application.Actionlet3@1']
5 flushMap['flushkey'] = [:]
6 flushMap['flushkey']['ret'] = $m1['ret']
7 flushMap['flushkey']['msg'] = $m1['msg']
8 flushMap['flushkey']['data'] = [:]
9 if($m2['data']){
10     if ($m1['data'] != null) {
11         def $default6 = [:]
12         $default6.putAll((Map)$m1['data'])
13         flushMap['flushkey']['data'] = $default6
14     }
15     flushMap['flushkey']['data']['moreLink'] = 'url'
16     if(_DList.valueOf($m1['data']['list']).size() > 20){
17         flushMap['flushkey']['data']['list'] = _DList.subList(_DList.valueOf($m1['data']['li
18     }
19     else{
20         flushMap['flushkey']['data']['list'] = $m1['data']['list']
21     }
22 }
23 def parameterMap = [:]
24 def header = [:]
25
```

参数列表

测试结果

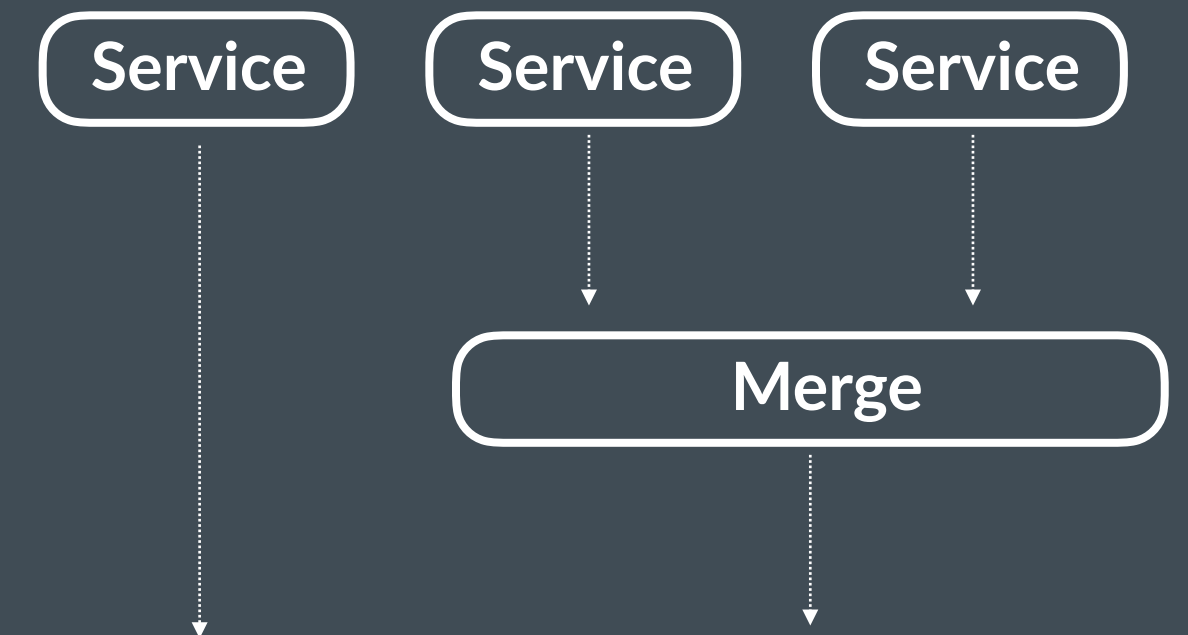
运行异常
Dson配置错误: @line 7

Groovy异常堆栈: java.lang.NullPointerException: Cannot get property 'ret' on null object
at org.codehaus.groovy.runtime.NullObject.getProperty(NullObject.java:60)
at org.codehaus.groovy.runtime.InvokerHelper.getProperty(InvokerHelper.java:172)
at org.codehaus.groovy.runtime.DefaultGroovyMethods.getAt(DefaultGroovyMethods.java:256)
at Dsl_Test_Forschema_10_1_11488247680.Biz(script1488247680945192585124.groovy:6)

N个接口任意组合

M个Flush到客户端

T个Callback



Web IDE

代码补全/提示

Debug工具

单元测试/ 接口测试

DSON!

基于JSON，所见即所得

能力受限的api及udf

自身语法和目标语言隔离



三 MWP路由层服务聚合 - DSON

```
{
  "#set($m1)": "$payloadMap['mwp.application.api@1']",
  "flushMap": {
    "flushkey": {
      "ret": "$m1['ret']",
      "data": {
        "MRArray": [{
          "#map($m1['data']['feature'],Map)": {
            "id": "$node['id']",
            "counter": {
              "#set($templist1)": "[1,2,3,4]",
              "#reduce($templist1, int, 0)": {
                "#set($my)": "_DInteger.valueOf($init)",
                "#if($init+1)": "$my+_DInteger.valueOf($node)",
                "#else": "$my"
              }
            }
          }
        ]
      }
    }
  },
  "parameterMap" : {},
  "header" : {}
}
```



```
{
  "flushMap" : {
    "flushkey" : {
      "ret" : "SUCCESS",
      "data" : {
        "MRArray" : [ {
          "id" : 1,
          "counter" : 6
        },
        {
          "id" : 2,
          "counter" : 6
        },
        {
          "id" : 3,
          "counter" : 6
        },
        {
          "id" : 4,
          "counter" : 6
        }
      ]
    }
  },
  "parameterMap" : {},
  "header" : {}
}
```



三 MWP路由层服务聚合 - 客户端使用

```
MWPRemote.getDSL()  
    .apiAndVersionId("dsl.test","1")  
    .parameterIs(dslParam).newCall()  
    .addObserver("flushkey1",new IDslObserver<List<Pet>>(){  
        @Override  
        public void call(IRemoteResponse<List<Pet>> response){  
            List<Pet> list = response.getData();  
        }  
    })  
    .addObserver("flushkey2",new IDslObserver(){  
        @Override  
        public void call(IRemoteResponse response){  
            Map data = response.getData();  
        }  
    })  
    .async(new IDslCallback(){  
        @Override  
        public void onCompleted(IRemoteContext context, IRemoteResponse response){  
        }  
    })  
    })
```



三 MWP聚合层

传统模式 =

客户端聚合

代码复杂度高

修改发版

多端维护

业务边界不清

服务端资源

聚合服务 =

客户端透明

专注业务代码

热部署

多端一致

展示逻辑后置

解放服务端



三 MWP挑战与规划



接入层改造

协议

语言

效率



下行通道

状态服务器

消息堆积

消息风暴



去中心化

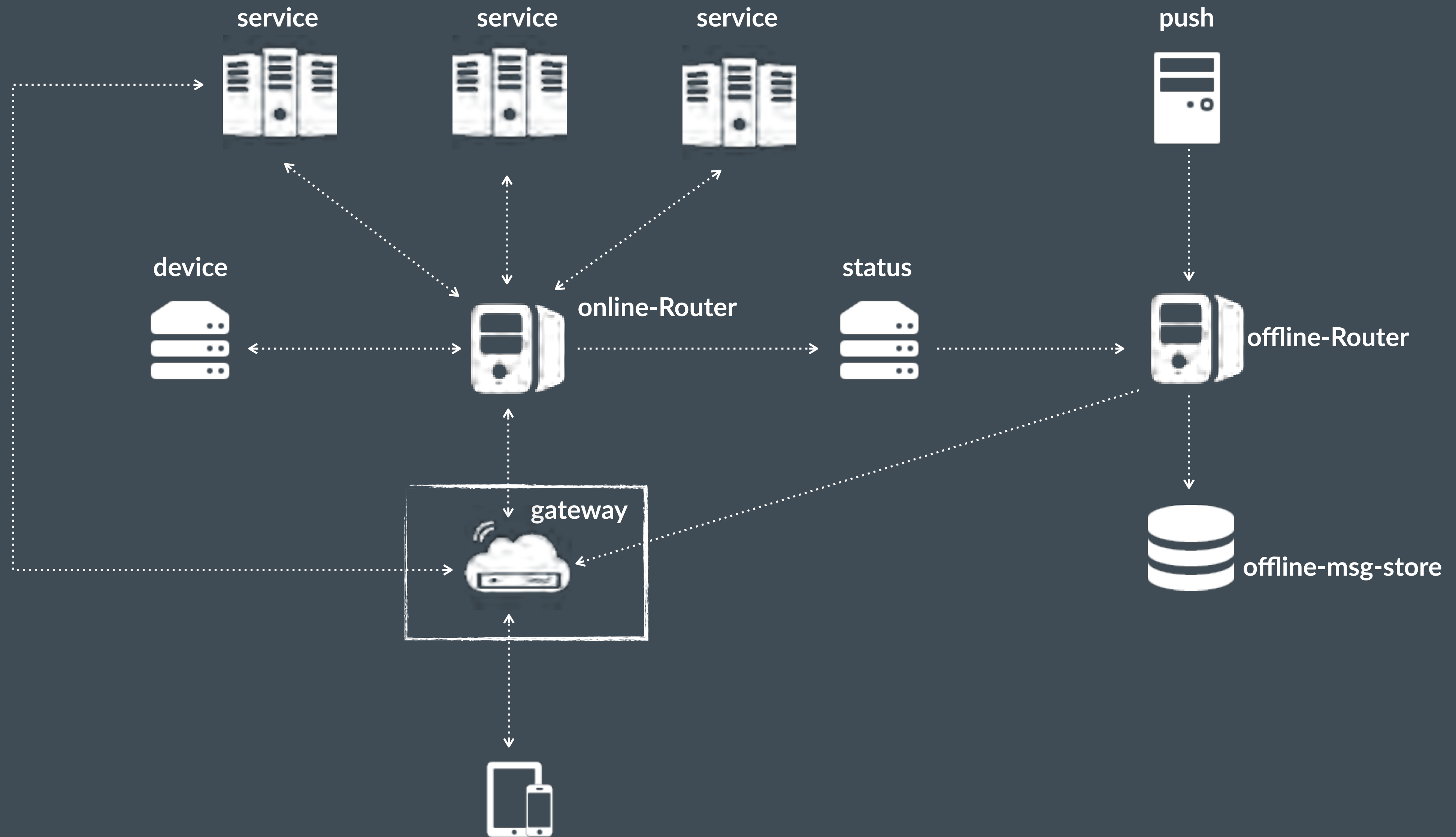
稳定性

职责下沉

容器化运维



三 MWP整体架构



三 MWP挑战与规划



接入层改造

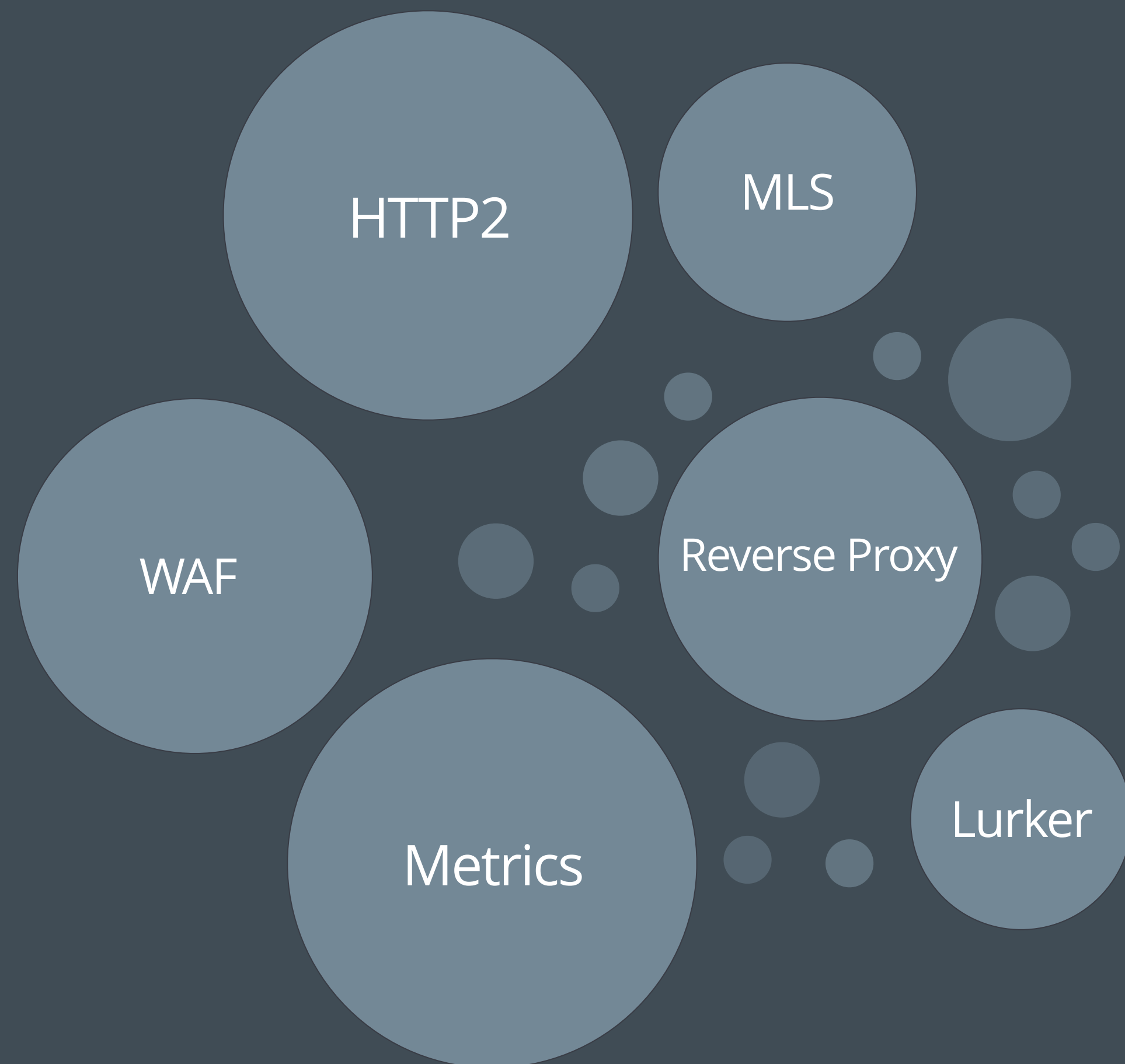
协议

语言

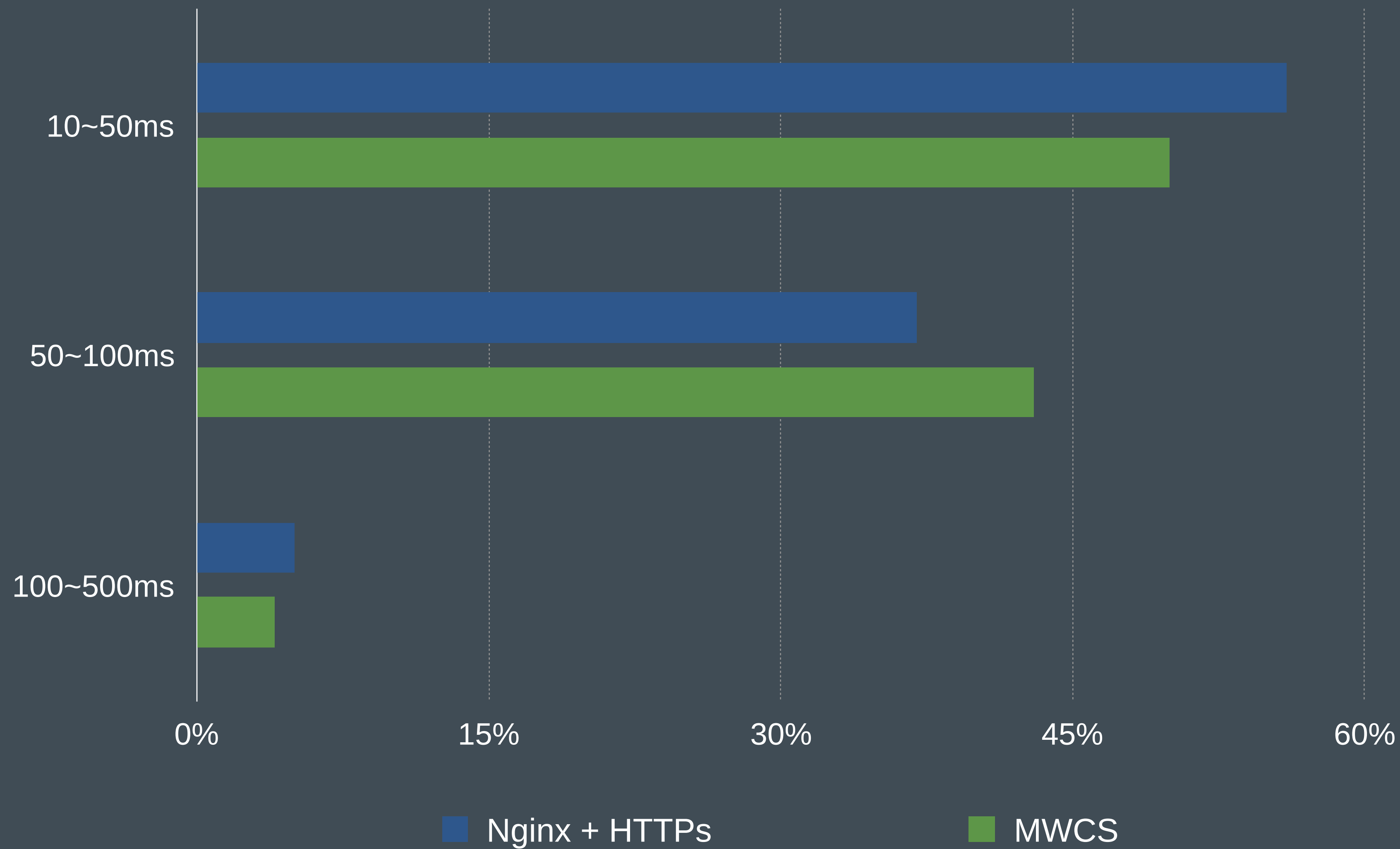
效率

- ✓ 需要更适合RPC的协议
stream RPC / bigpipe / 易扩展
- ✓ C++ & Java & Golang
野指针 / lambda / 内存 / gc / cgo
- ✓ 运维和效率
Nginx化/ 守护进程 / 插件化开发





≡ MWCS VS NGINX



压测数据

QPS : ± 12000

✓ goroutine num : ± 800

✓ gc count per second : 2

✓ pause time per gc : $\pm 1\text{ms}$



思考和规划

Stream-RPC

- 多次调用flush的性能
- 滑动窗口优化

Log

- SDK log对内存和api不友好
- flush dirty page ratio

GC & CGO

- sync.pool是好东西
- cgo 有风险
- golang 1.8



Thank you

