

零基础构建自己的服务治理框架



自我介绍

我是周梦康，6年的后端开发。

常用语言 `php`、`java`，《深入 PHP 内核》作者之一。

目前就职于阿里云。

博客	https://mengkang.net
github	https://github.com/zhoulengkang
微博	http://weibo.com/zmkang
微信	zhoulengkang
邮箱	zhoulengkang@php.net



本次分享的视频讲解地址（扫描二维码即可查看）



为什么需要服务治理

远程调用的实现

跨语言通信的实现

注册中心的实现

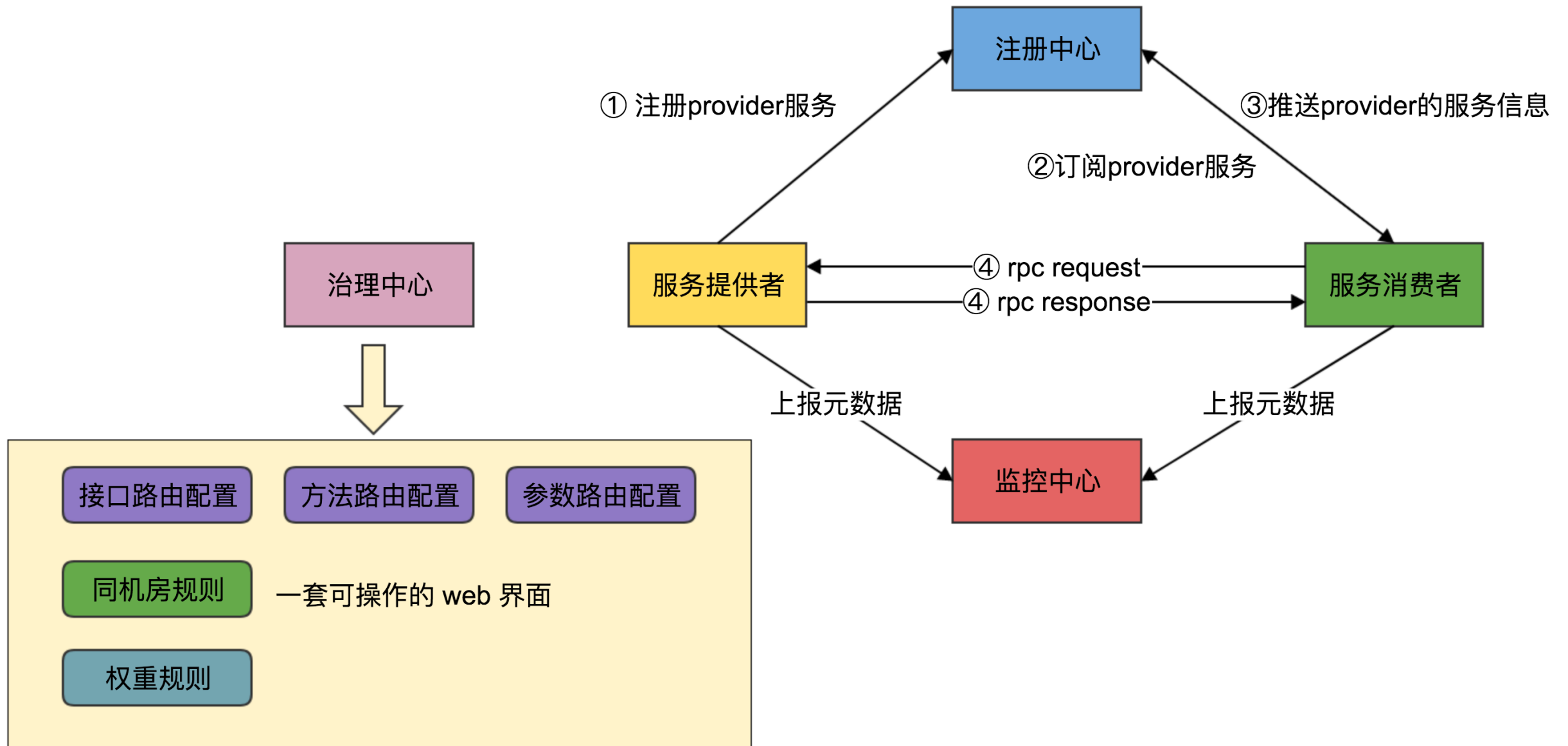
监控中心的实现

多维度的服务治理

RPC 协议的优化



服务治理框架图





为什么要使用服务治理?

原本很简单的事情为什么要弄这么复杂?

服务治理能提高性能吗?

问题追踪更加复杂了, 怎么解决?

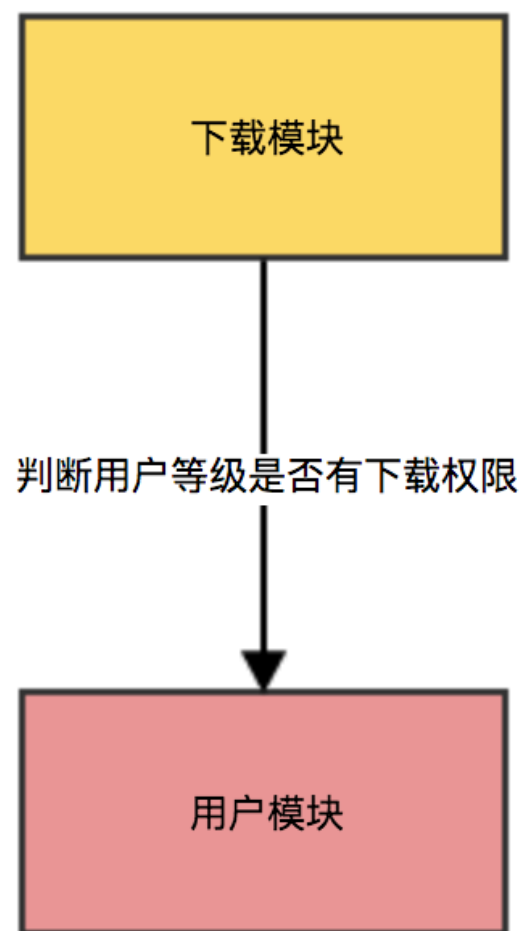
我们学习这堂课之后都要切换到服务治理的架构方式上来吗?



前期：编写模块化代码

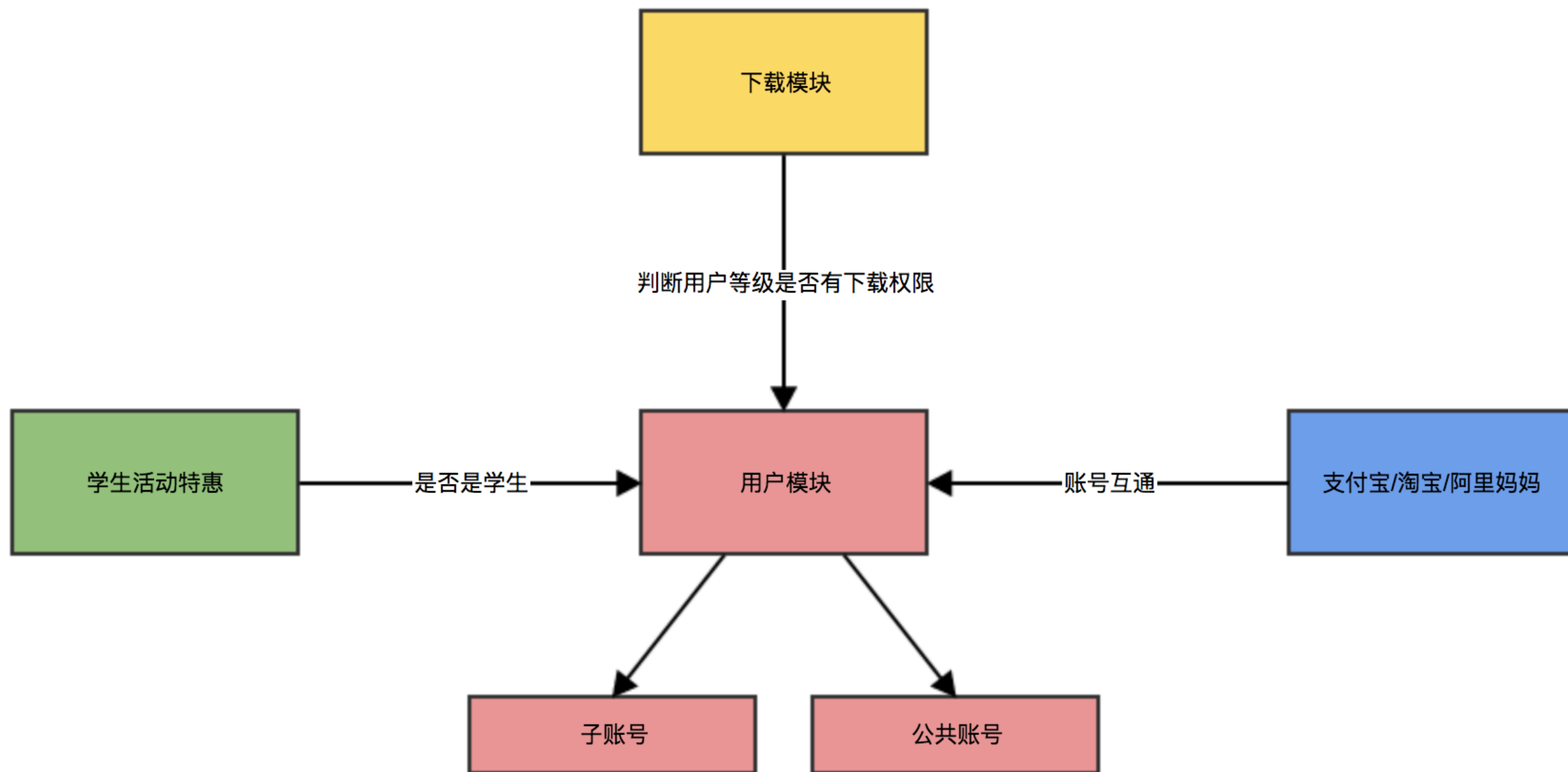


业务举例：根据用户等级下载附件





前期：编写模块化代码



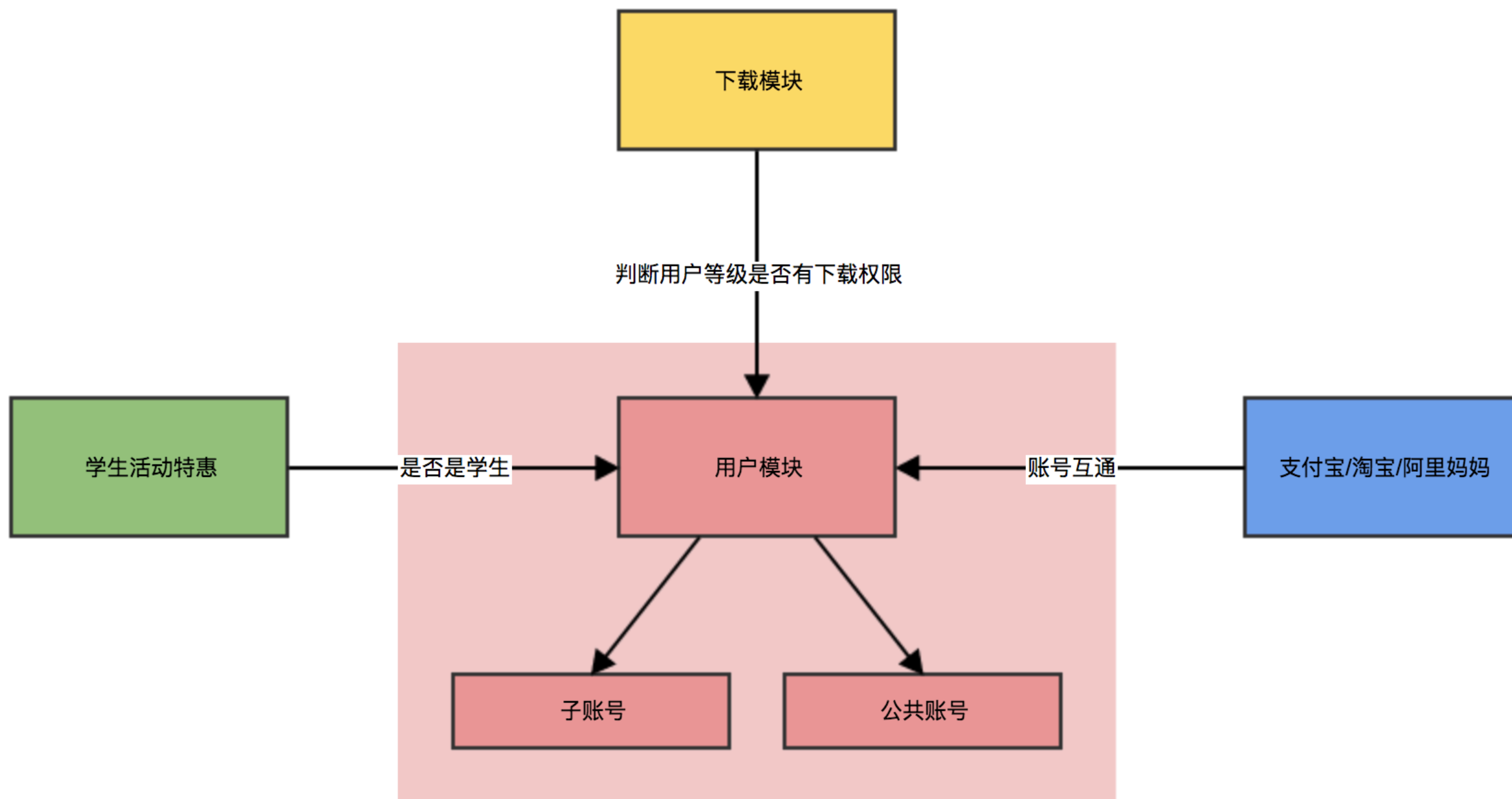


指导思想

高内聚低耦合

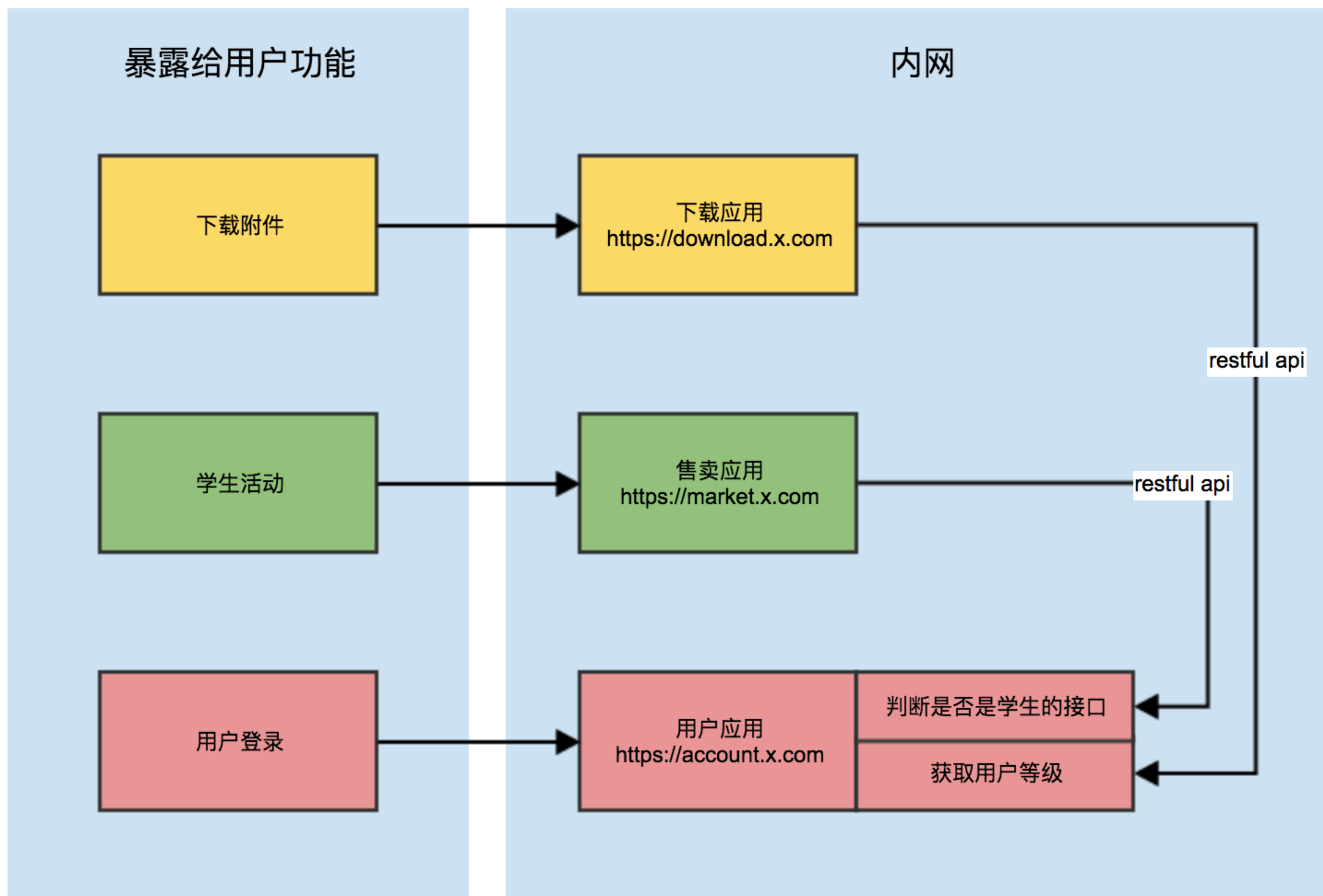


中期：服务拆分





中期：各个独立运用成形

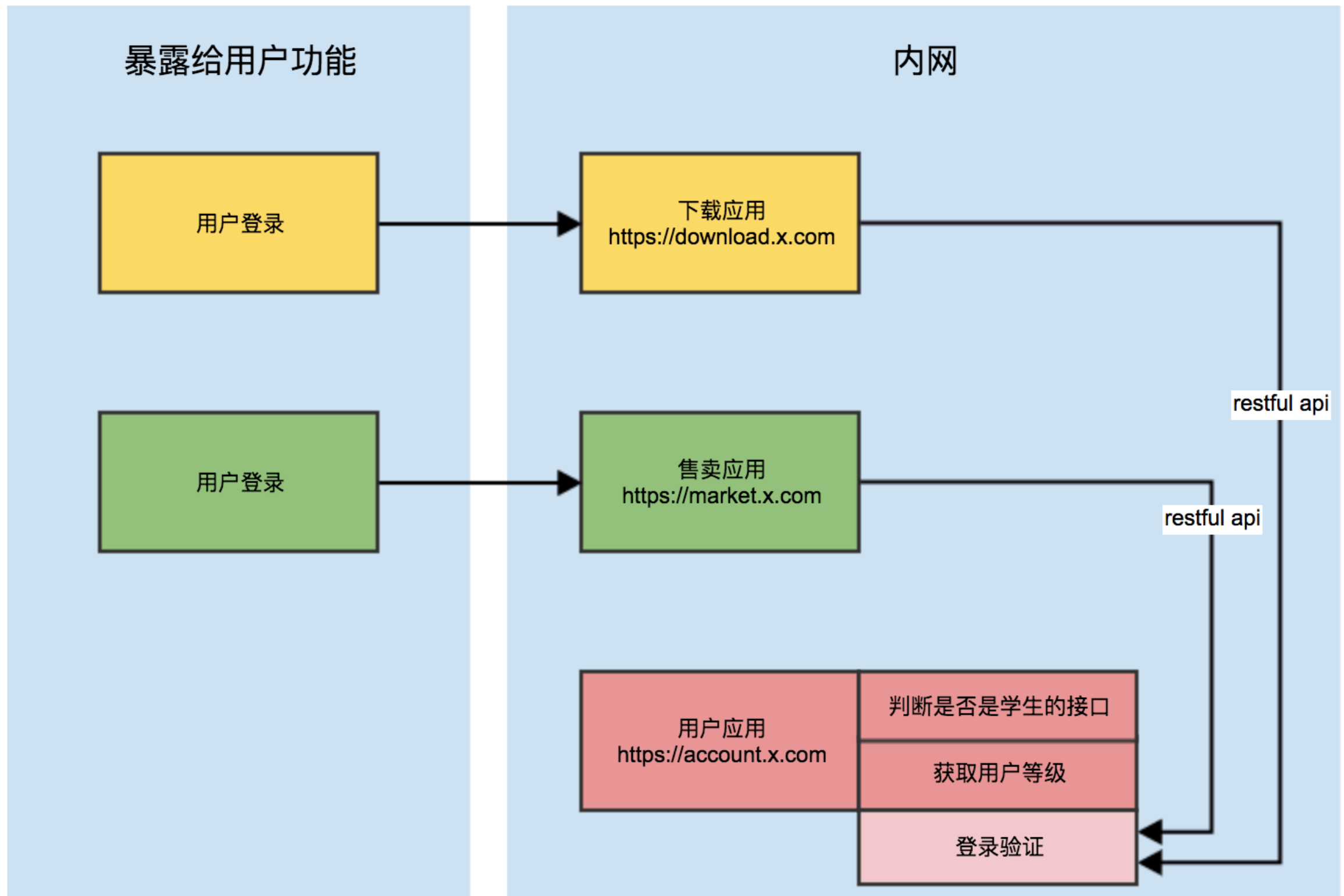




为什么登录独立处理？

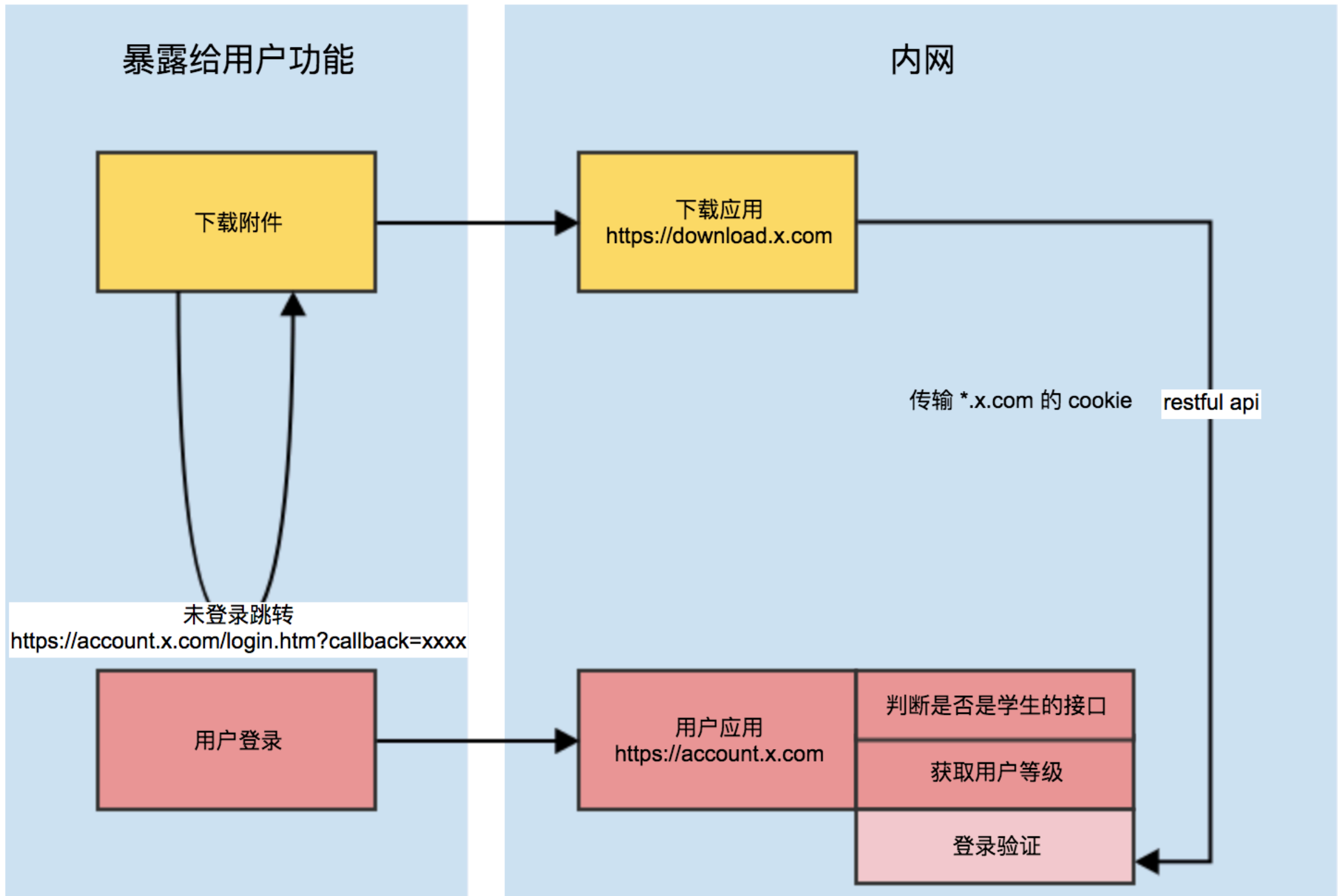


登录方案1





登录方案2





梳理用户应用对外的服务

UserService.getUserLevel

UserService.getStudentInfo

UserService.loginAuthCheck



服务治理出现的原因

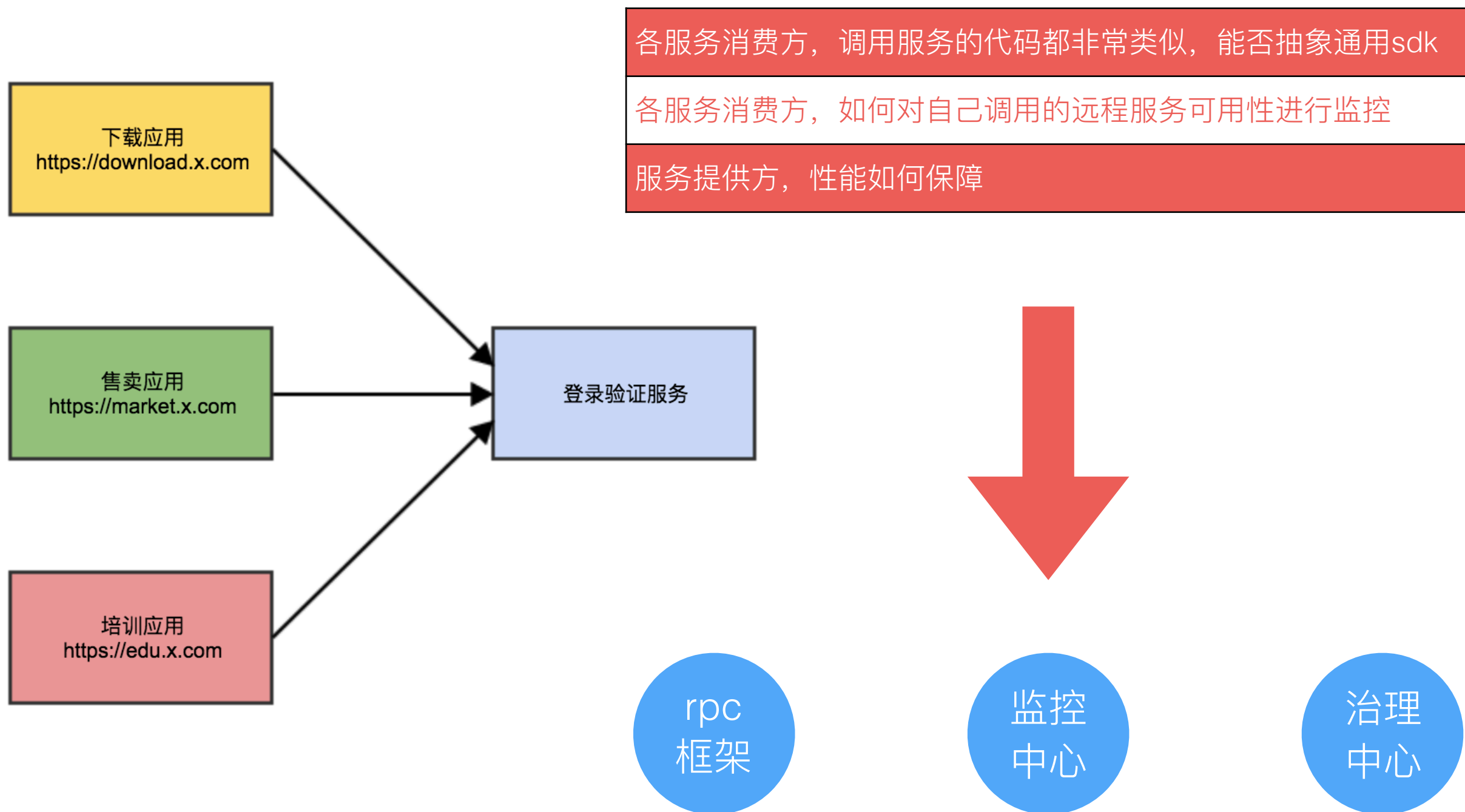
业务不断拓展，降低服务之间的耦合，从功能上拆分

接口协议杂乱无章，沟通成本高

整个分布式系统稳定性保障



以用户登录认证为例





A 服务器上

```
$userService = new UserService();  
$userService->getUserInfo($uid);
```

B 服务器想调用

```
$client = new \SDK\Client();  
$request = new \SDK\UserService\Request\GetStudentInfoRequest();  
$request->setUid($uid);  
$request->setMethod("GET");  
$response = $client->doAction($request);
```



从理论到实践 - 变型

```
class Client
{
    private $url;
    private $service;

    private $rpcConfig = [
        "UserService" => "http://127.0.0.1:8081",
    ];

    public function __construct($service)
    {
        if (array_key_exists($service, $this->rpcConfig)) {
            $this->url = $this->rpcConfig[$service];
            $this->service = $service;
        }
    }

    public function __call($action, $arguments)
    {
        $content = json_encode($arguments);
        $options['http'] = [
            'timeout' => 5,
            'method' => 'POST',
            'header' => 'Content-type:application/x-www-form-urlencoded',
            'content' => $content,
        ];

        $context = stream_context_create($options);

        $get = [
            'service' => $this->service,
            'action' => $action,
        ];

        $url = $this->url . "?" . http_build_query($get);

        $res = file_get_contents($url, false, $context);

        return json_decode($res, true);
    }
}

$userService = new Client('UserService');
var_export($userService->getUserInfo(103));
```

```
class UserService
{
    public static function getUserInfo($uid)
    {
        // 假设以下内容从数据库取出
        return [
            'id' => $uid,
            'username' => 'meng kang',
        ];
    }

    $service = $_GET['service'];
    $action = $_GET['action'];
    $argv = file_get_contents("php://input");

    if (!$service || !$action) {
        die();
    }

    if ($argv) {
        $argv = json_decode($argv, true);
    }

    $res = call_user_func_array([$service, $action], $argv);

    echo json_encode($res);
}
```



从理论到实践 - 变型 - 小结

我们做了如下工作

使用 `__call` 调用了本地不存在的方法

使用了 http 协议进行传输

使用了 json 进行内容的序列化编码 (同时 http 本身对 body 体做了二进制编码)



从理论到实践 - 跨语言调用 - java 客户端

第三方给的 sdk

```
package net.mengkang.sdk;

public class User {
    private Integer id;
    private String username;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    @Override
    public String toString() {
        return "User{" +
            "id=" + id +
            ", username='" + username + '\'' +
            '}';
    }
}
```

```
package net.mengkang.sdk;

public interface UserService {
    User getUserInfo(Integer uid);
}
```

RPC 调用

```
package net.mengkang.rpc;

import java.lang.reflect.Proxy;

public class RpcClient {
    public final Object proxy(Class type) {
        RpcClientInvocationHandler handler = new RpcClientInvocationHandler(type);
        return Proxy.newProxyInstance(type.getClassLoader(), new Class[]{type}, handler);
    }
}
```

```
package net.mengkang.rpc;

import com.alibaba.fastjson.JSON;
import com.alibaba.fastjson.JSONException;
import java.lang.reflect.InvocationHandler;
import java.lang.reflect.Method;

class RpcClientInvocationHandler implements InvocationHandler{

    private Class service;

    public RpcClientInvocationHandler(Class clazz) {
        service = clazz;
    }

    public Object invoke(Object proxy, Method method, Object[] args) throws Throwable {
        Class returnType = method.getReturnType();
        String[] serviceName = service.getName().split("\\.");

        String url = "http://127.0.0.1:8081?service=" + serviceName[serviceName.length - 1] + "&action=" + m

        String httpResponse = RpcRequest.doPost(url, args);

        Object res = null;

        try{
            res = JSON.parseObject(httpResponse, returnType);
        }catch (JSONException e){
            e.printStackTrace();
        }

        return res;
    }
}
```



从理论到实践 - 跨语言调用 - java 客户端

```
package net.mengkang;

import net.mengkang.rpc.RpcClient;
import net.mengkang.sdk.User;
import net.mengkang.sdk.UserService;

public class Demo {
    public static void demo(){
        RpcClient rpcClient = new RpcClient();
        UserService userService = (UserService) rpcClient.proxy(UserService.class);
        User user = userService.getUserInfo(10);
        System.out.println(user.getId());
        System.out.println(user.toString());
    }

    public static void main(String args[]){
        Demo.demo();
    }
}
```

```
public class Demo {
    public static void demo(){
        RpcClient rpcClient = new RpcClient();
        UserService userService = (UserService) rpcClient.proxy(UserService.class);
        User user = userService.getUserInfo(10);
        System.out.println(user.getId());
        System.out.println(user.toString());
    }

    public static void main(String args[]){
        Demo.demo();
    }
}
```

Expression: `user`

Result:

- result = {User@1515} "User{id=10, username='mengkang'}"
- id = {Integer@1518} "10"
- username = {String@1522} "mengkang"



Java 的优势

- ★ 更加本地化、更加透明
- ★ 比前面 PHP 更进一步，直接返回本地对象，远程无感知



从理论到实践 - 源码地址

PHP 客户端	https://gitee.com/zhoulengkang/soa-01-php-client
Java 客户端	https://gitee.com/zhoulengkang/soa-01-java-client
PHP 服务端	https://gitee.com/zhoulengkang/soa-01-php-service



从概念到实践 - 不足之处

```
class Client
{
    private $url;
    private $service;



    private $rpcConfig = [
        "UserService" => "http://127.0.0.1:8081",
    ];

    public function __construct($service)
    {
        if (array_key_exists($service, $this->rpcConfig)) {
            $this->url = $this->rpcConfig[$service];
            $this->service = $service;
        }
    }

    public function __call($action, $arguments)
    {
        $content = json_encode($arguments);
        $options['http'] = [
            'timeout' => 5,
            'method' => 'POST',
            'header' => 'Content-type:application/x-www-form-urlencoded',
            'content' => $content,
        ];
    }
}
```



从概念到实践 - 注册中心

TABLES		Field	Type	Length
 consumer		id	INT	11
 provider		ip	VARCHAR	15
		prot	VARCHAR	5
		service	VARCHAR	50



从概念到实践 - 简单配置中心

服务消费者	https://gitee.com/zhoulmengkang/soa-02-all/tree/master/consumer
服务提供者	https://gitee.com/zhoulmengkang/soa-02-all/tree/master/provider
注册中心	https://gitee.com/zhoulmengkang/soa-02-all/tree/master/config

为什么需要服务治理

远程调用的实现

跨语言通信的实现

注册中心的实现

监控中心的实现

多维度的服务治理

RPC 协议的优化



谢谢



原文：<https://segmentfault.com/l/1500000011300619>