使用MVC、MVP、MVVM和FRP实现Android局域网群聊应用

陈俊达，李培林，张凌哲，蔡蔚霖

2019年5月29日

项目和项目展示

- https://github.com/viccrubs/android-chat-in-4-patterns

目录

- Android应用是如何连接界面和逻辑的?
- MVC介绍
  - 代码展示
  - 出现的问题
- MVP介绍
  - 代码展示
  - 出现的问题
- MVVM介绍
  - 代码展示
  - 出现的问题
- FRP介绍
  - 代码展示
- 作业介绍

Android应用是如何连接界面和逻辑的?

- XML定义界面，Activity类定义逻辑
- 通过ID号，在类中操作界面元素

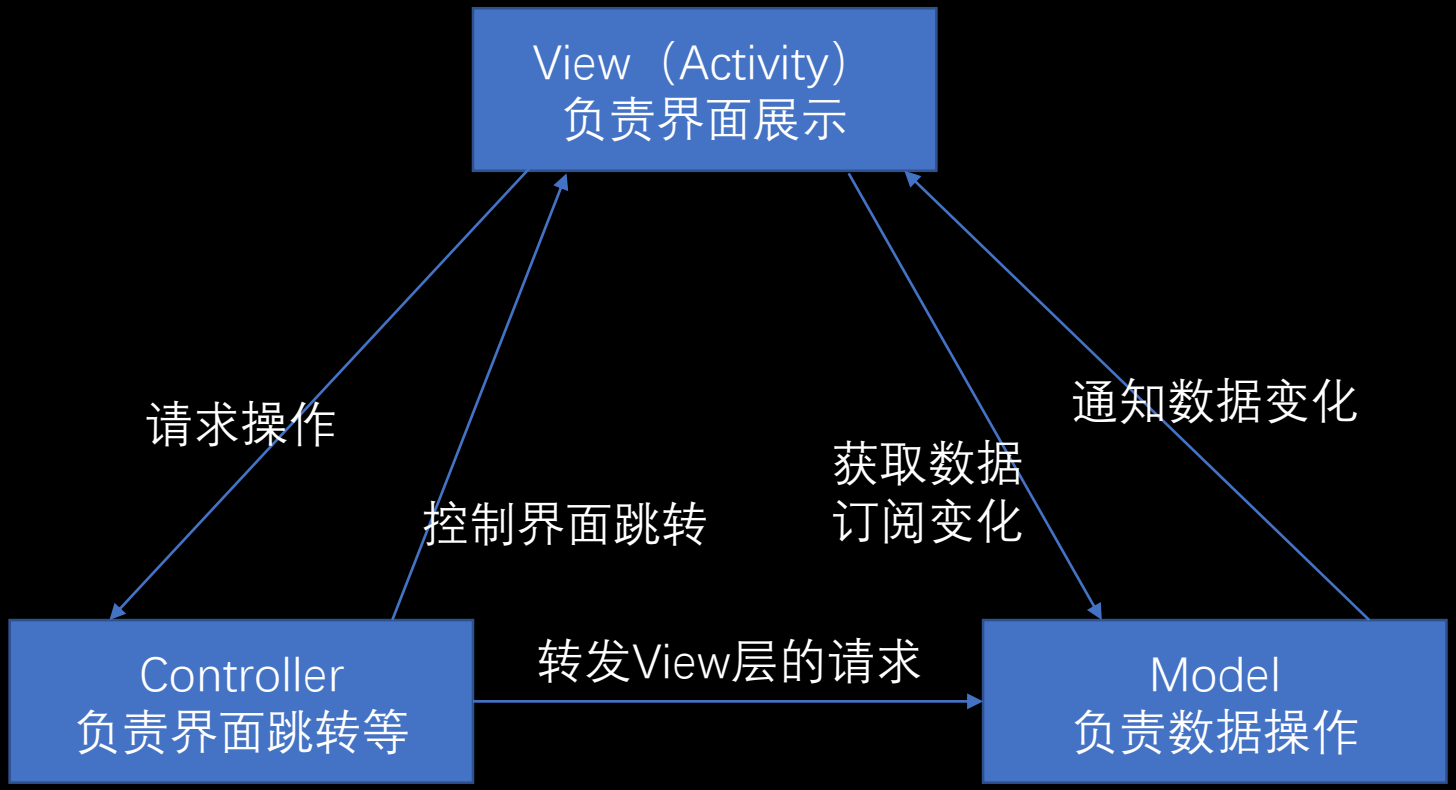
```
<EditText
    android:id="@+id/et_content"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/message_shap_chat_bg"
    android:imeOptions="actionSend"
    android:maxLines="1"
    android:minHeight="36dp"
    android:paddingStart="13dp"
    android:textSize="13sp"

    />
```

给元素设置ID

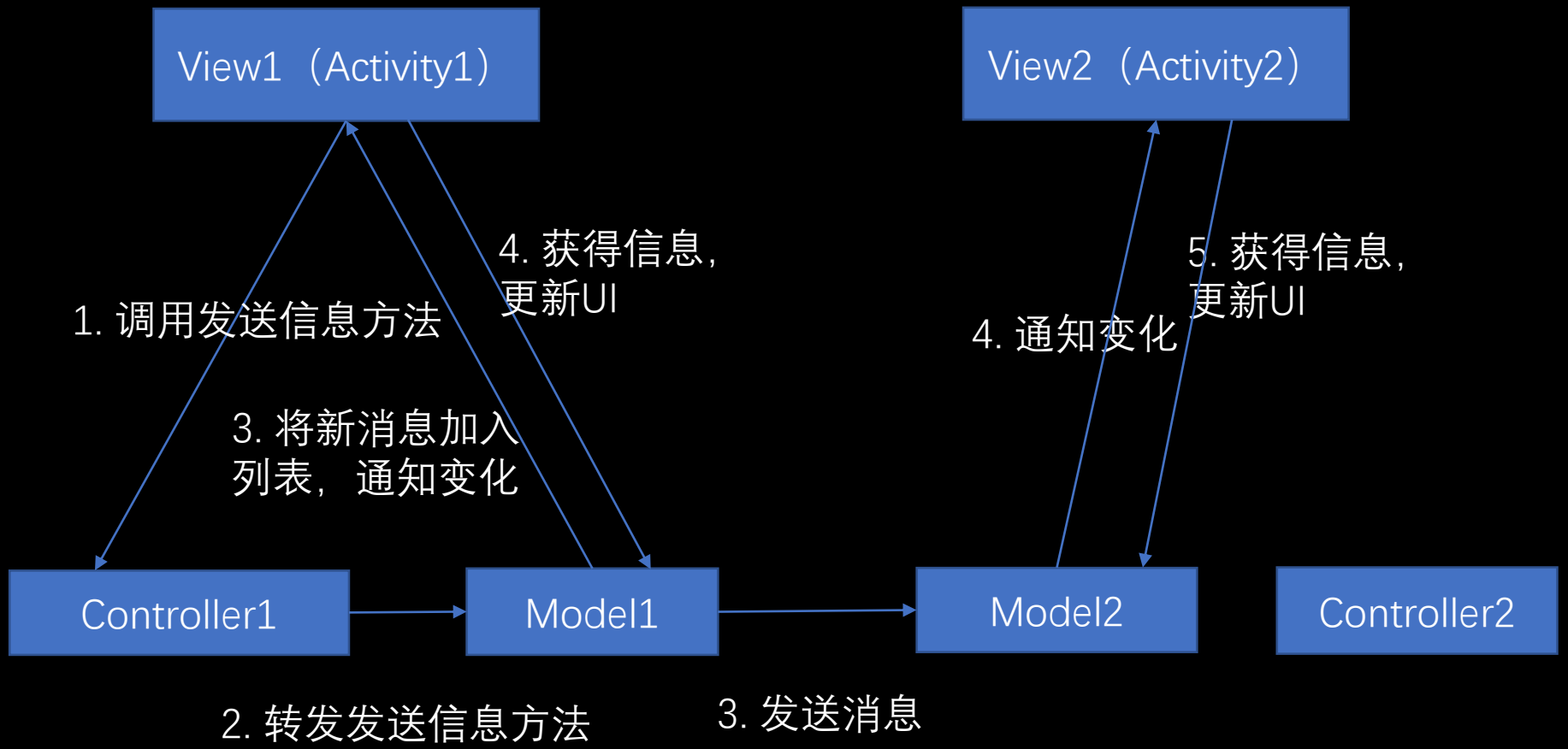
```
// Input事件处理
EditText editText = findViewById(R.id.et_content);
editText.setOnEditorActionListener(this);
```

通过ID查找元素对象并进行操作

MVC


View（Activity）
负责界面展示
请求操作
控制界面跳转
获取数据
订阅变化
通知数据变化
Controller
负责界面跳转等
转发View层的请求
Model
负责数据操作

MVC在发送信息时的数据流



MVC优点

- 分离逻辑和界面细节操作
- 避免Activity代码爆炸

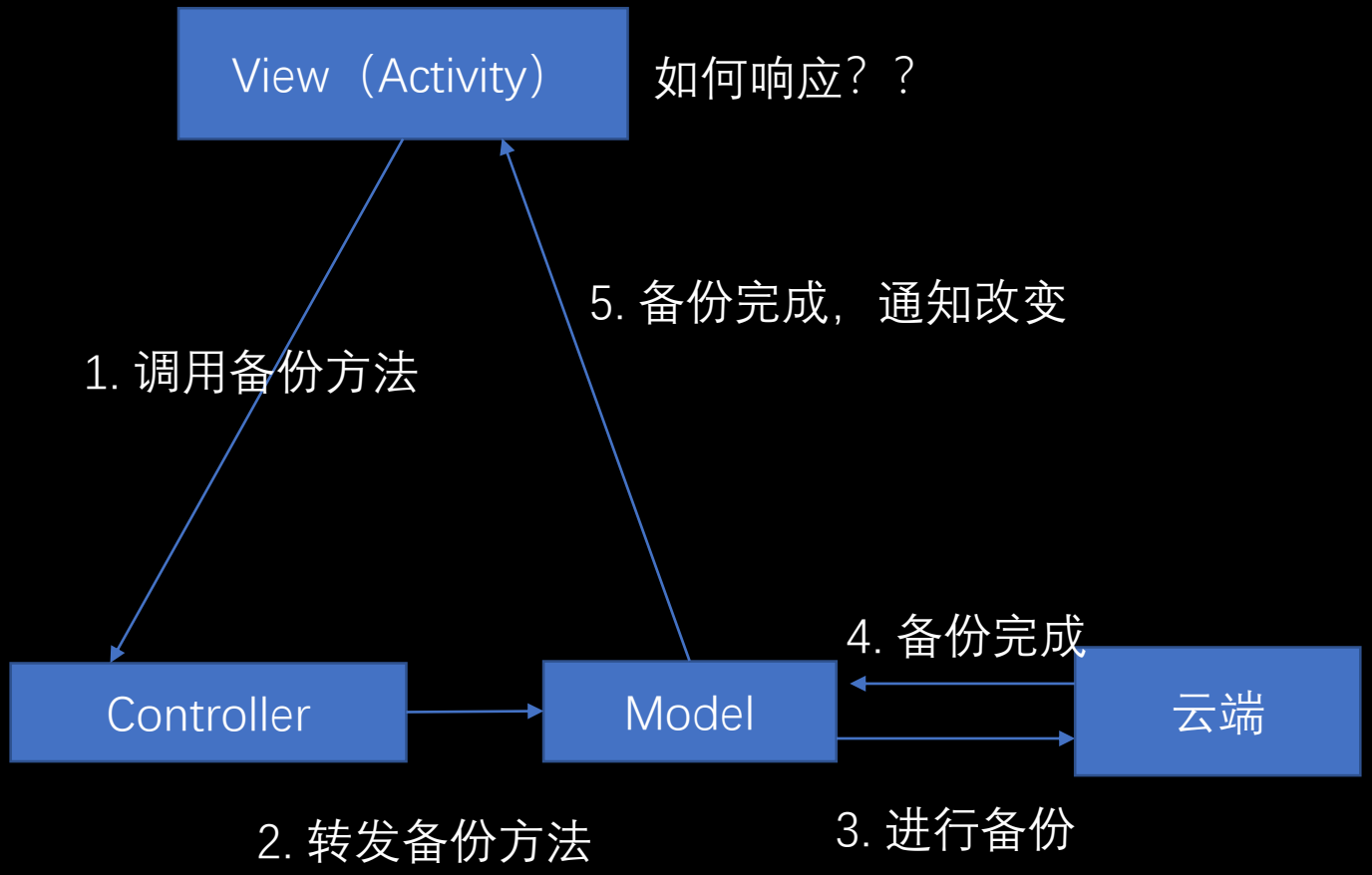
MVC缺点

- 界面控制不灵活
  - 只能由M变化通知V变化
  - 异步操作，耗时操作?

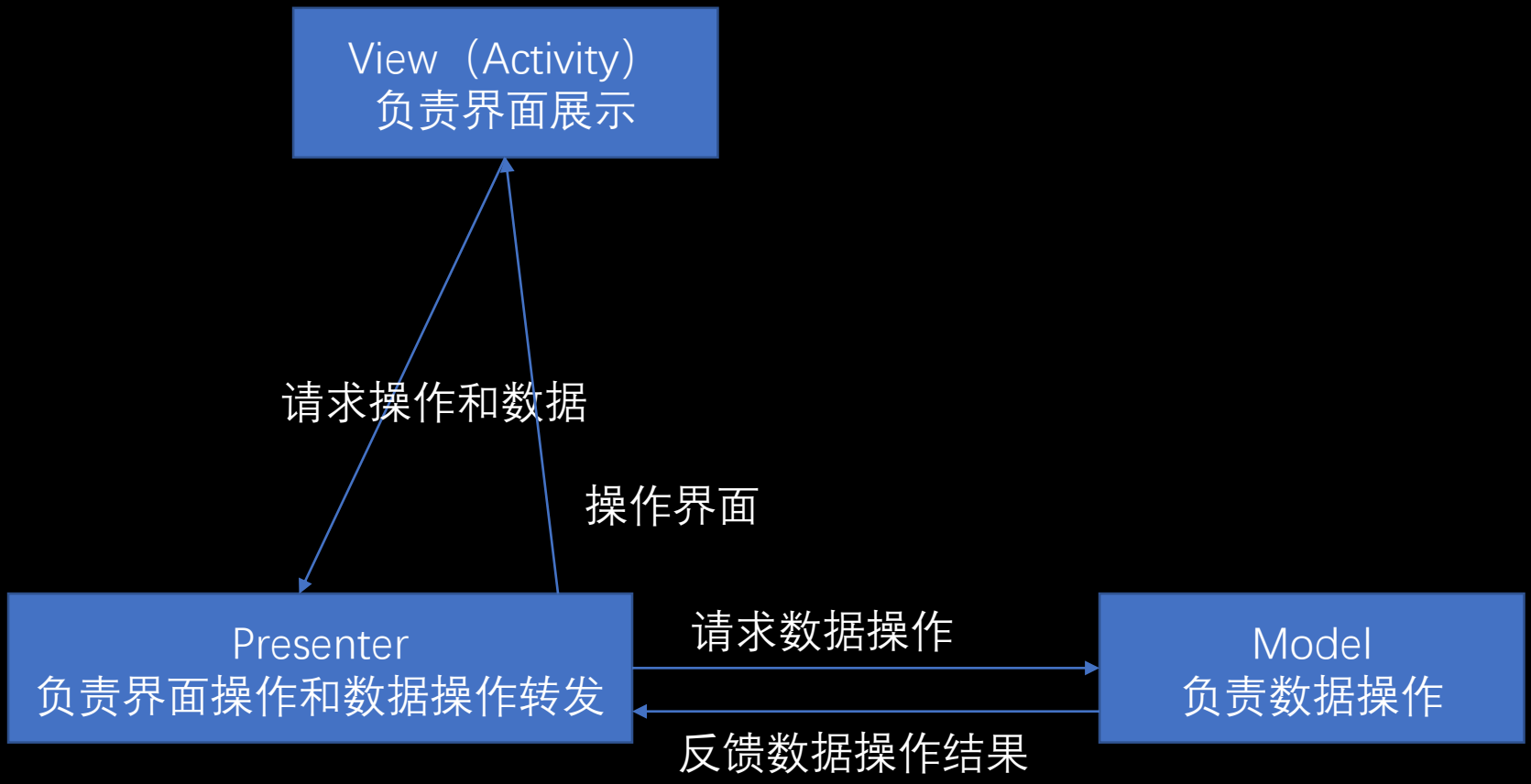
消息备份功能说明

- 一个大按钮，按下后备份消息
- 界面显示最后一次备份消息

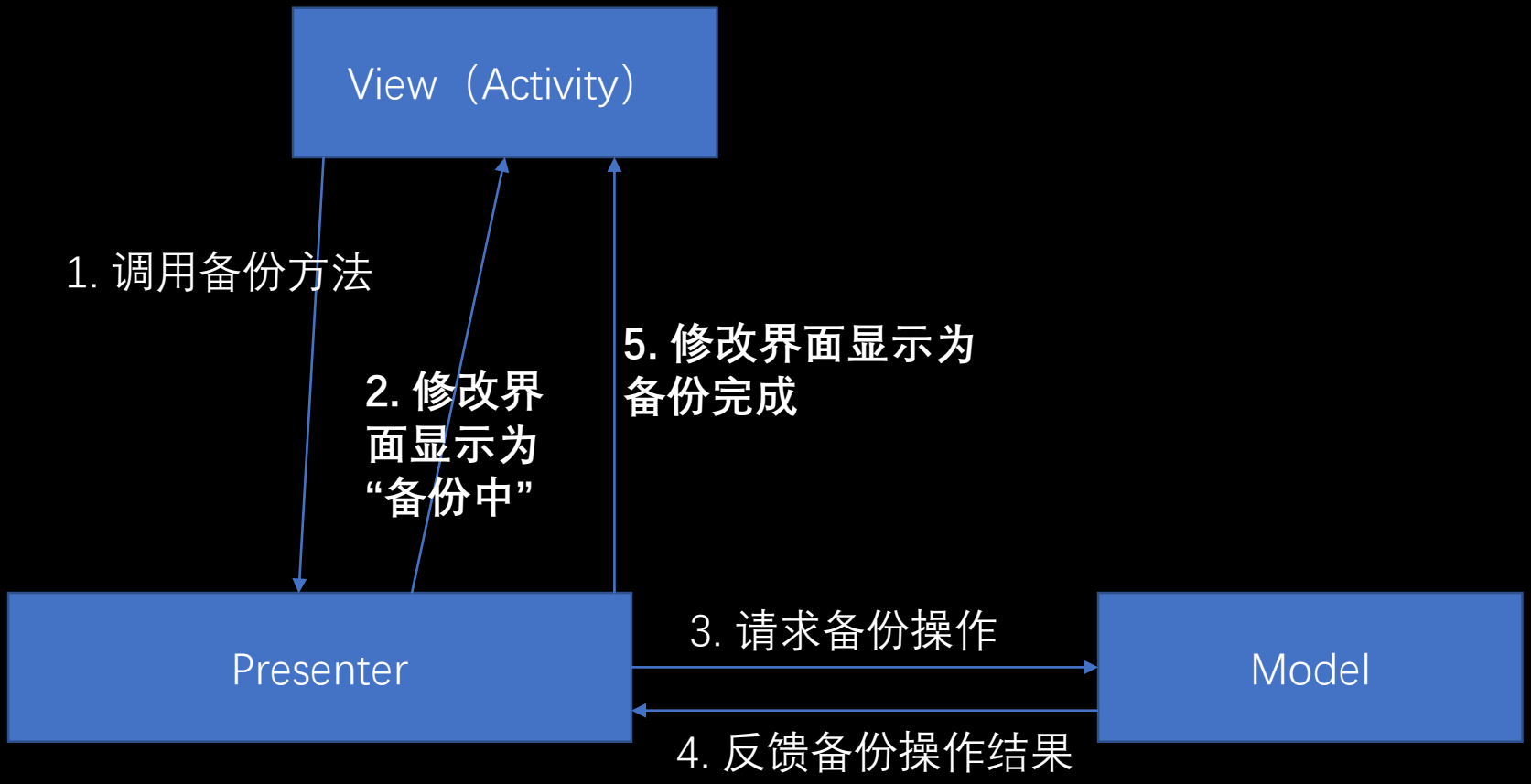
使用MVC实现备份功能界面



MVP


View（Activity）
负责界面展示
请求操作和数据
操作界面
Presenter
负责界面操作和数据操作转发
请求数据操作
反馈数据操作结果
Model
负责数据操作

MVP在备份时的数据流

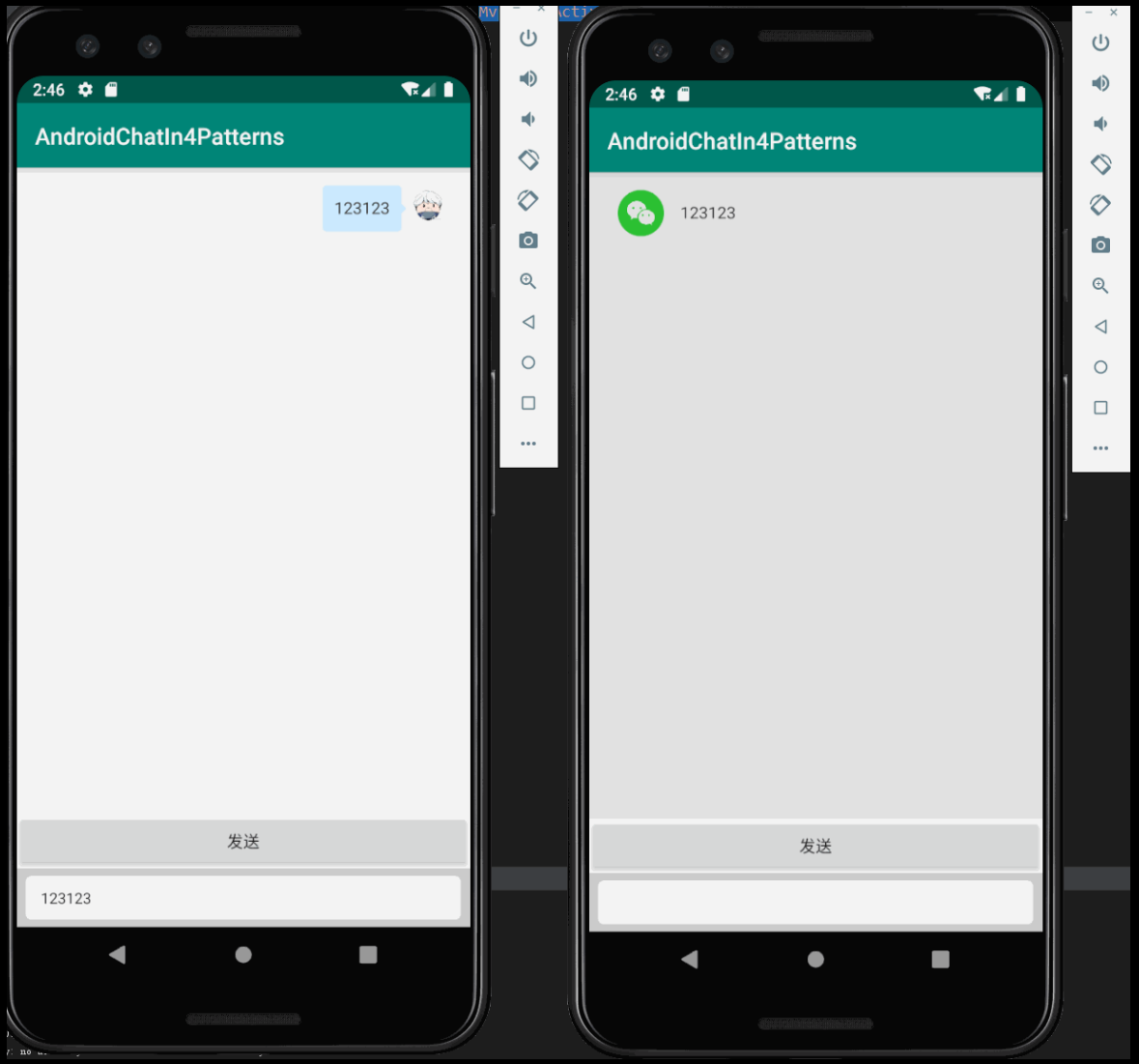


MVP特点

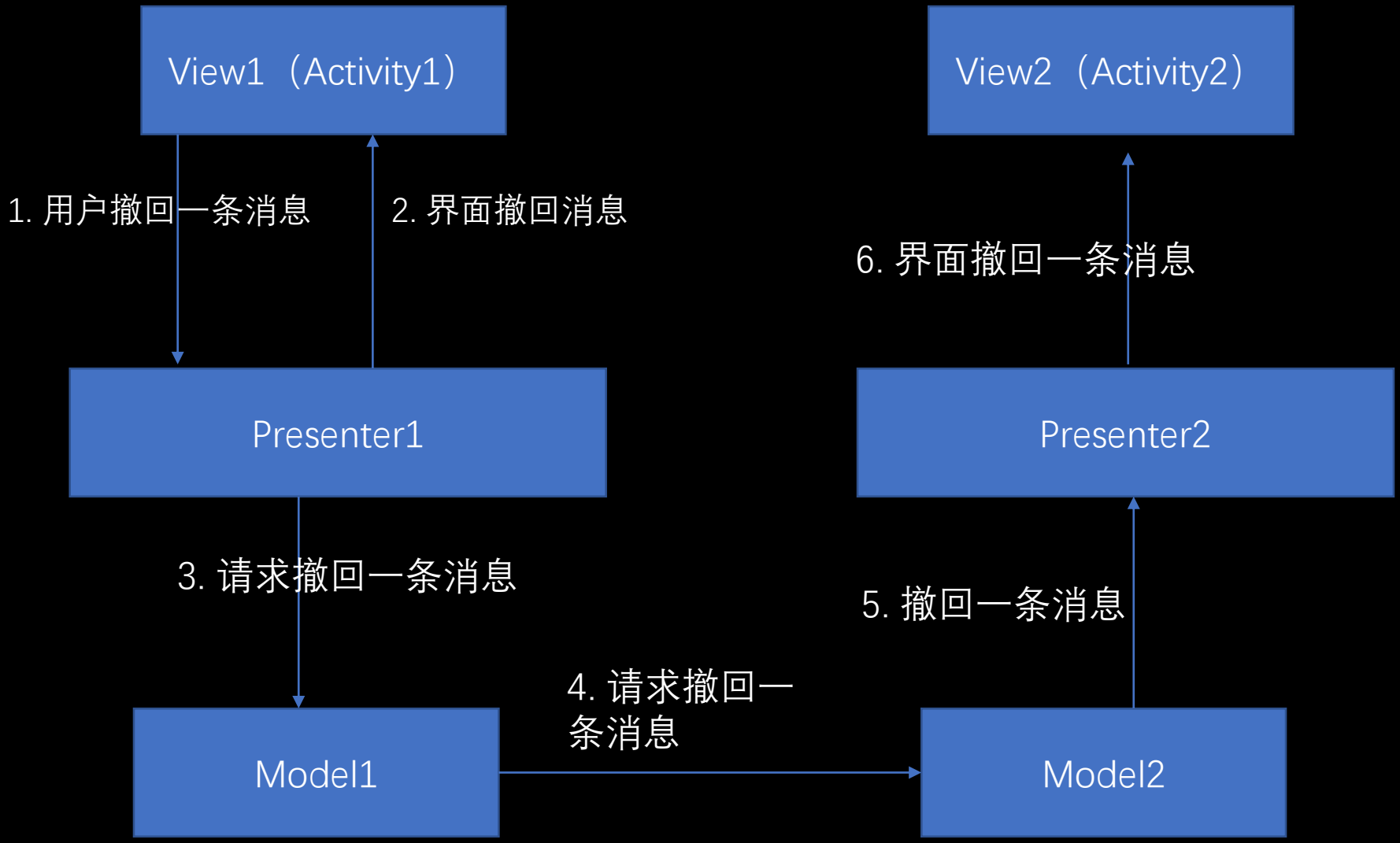
- 解耦**数据层**和**显示层**
- （通过Presenter）增加界面的控制粒度
- （进一步抽象接口）便于重用和测试

撤回消息功能说明

- 长按一个按钮，弹出确认框
- 撤回消息后消息内容被替换为“已撤回”



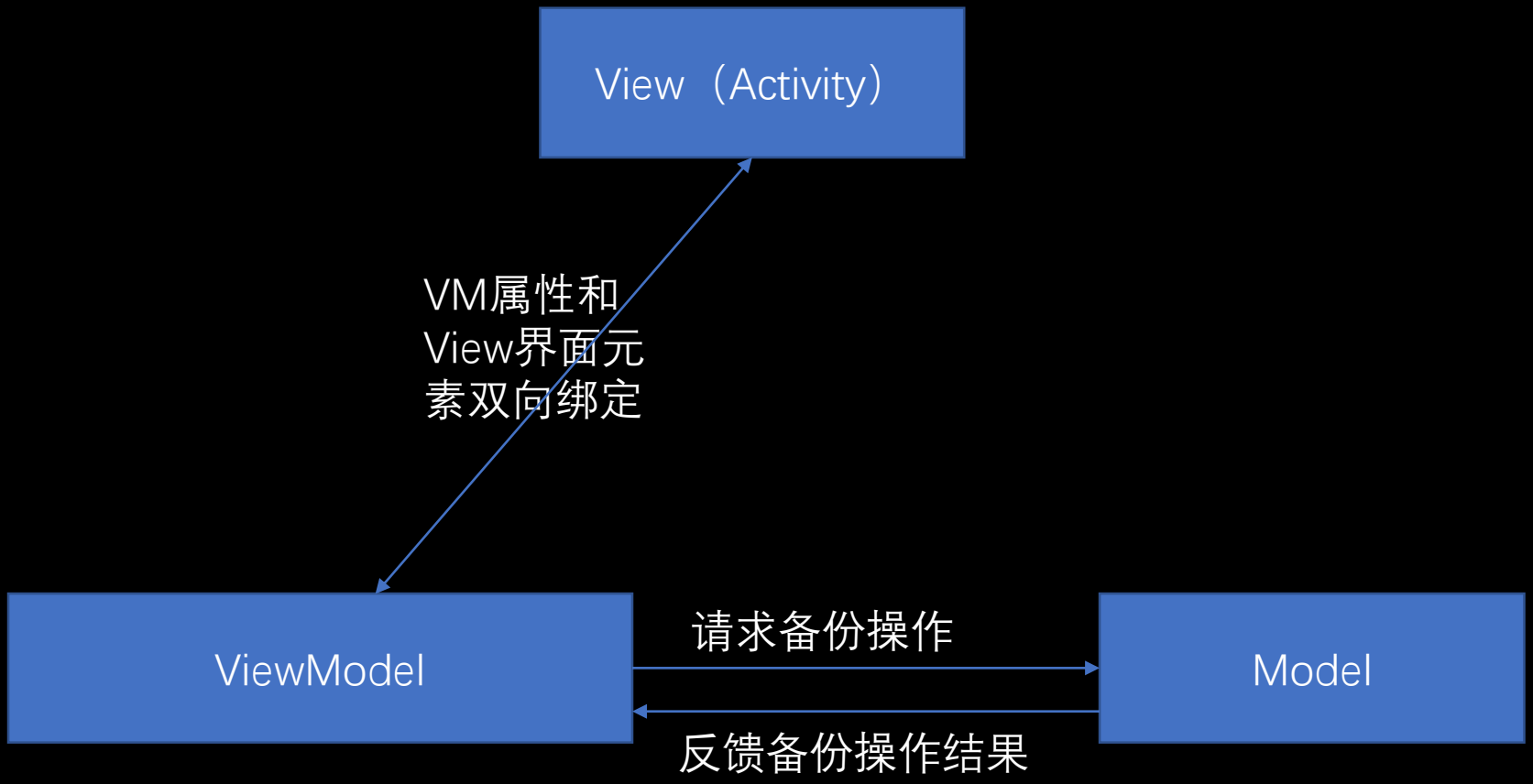
使用MVP实现消息撤回



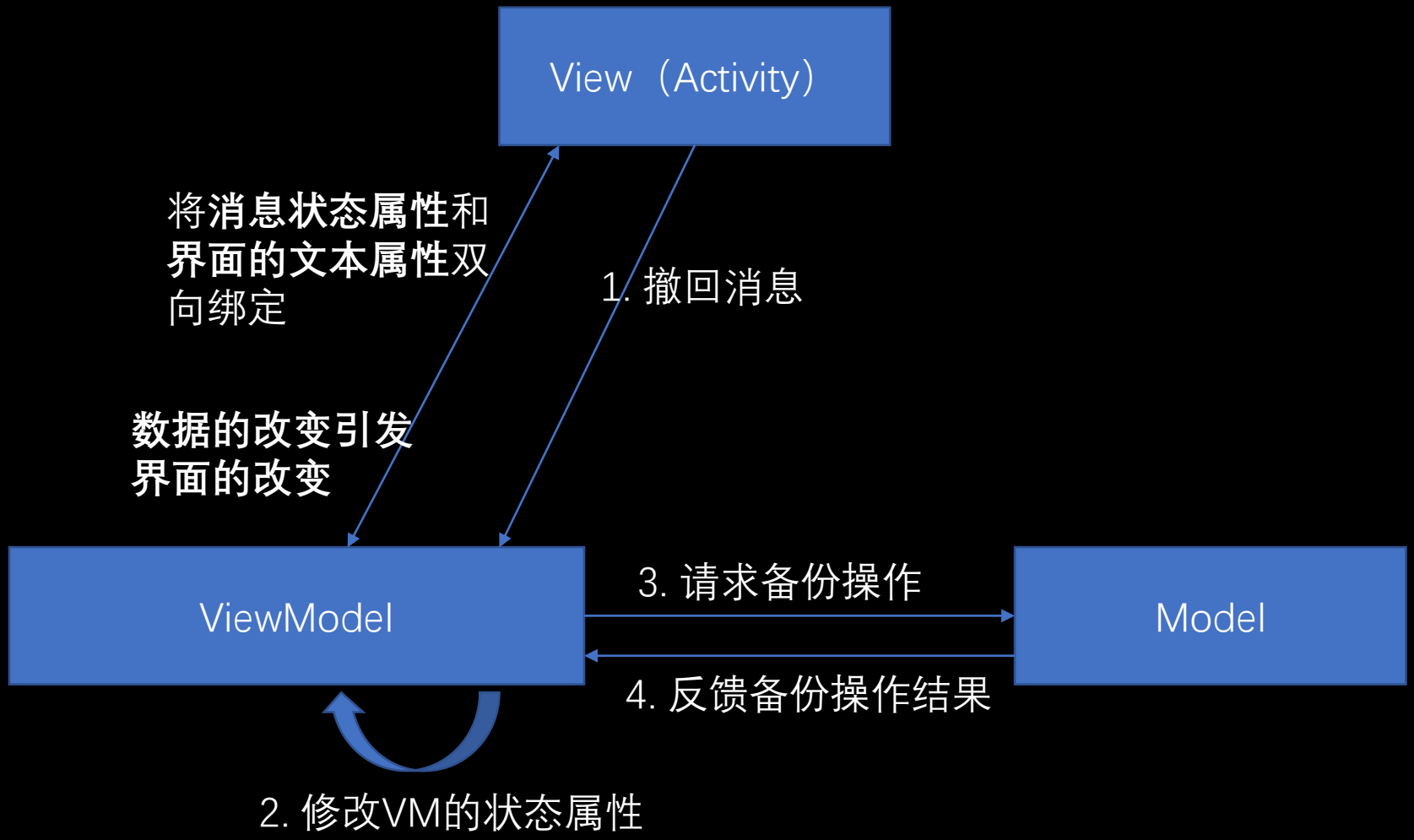
MVP问题

- Presenter职责繁重，冗余逻辑
  - 既需要负责处理数据操作，也需要操作UI
  - 将“撤回”考虑为一种状态，增加一种状态就需要增加Presenter的工作量
  - 举例：增加一个“标记消息为重要”的功能
- Presenter耦合Model和View
  - 当操作需要的UI操作比较复杂/发生改变时，Presenter容易出错/忘记修改
  - 举例：撤回失败？

MVVM


View（Activity）
VM属性和
View界面元
素双向绑定
ViewModel
请求备份操作
Model
反馈备份操作结果

使用MVVM实现消息撤回



MVVM优点

- 更加解耦**数据操作**和**界面操作**
  - 数据只操作数据，由双向绑定修改界面
  - 界面改变时，由双向绑定更新数据
    - 输入框内容改变时，修改VM中的输入框数据
- **扩展性++，可维护性++，清晰度++**

MVVM缺点

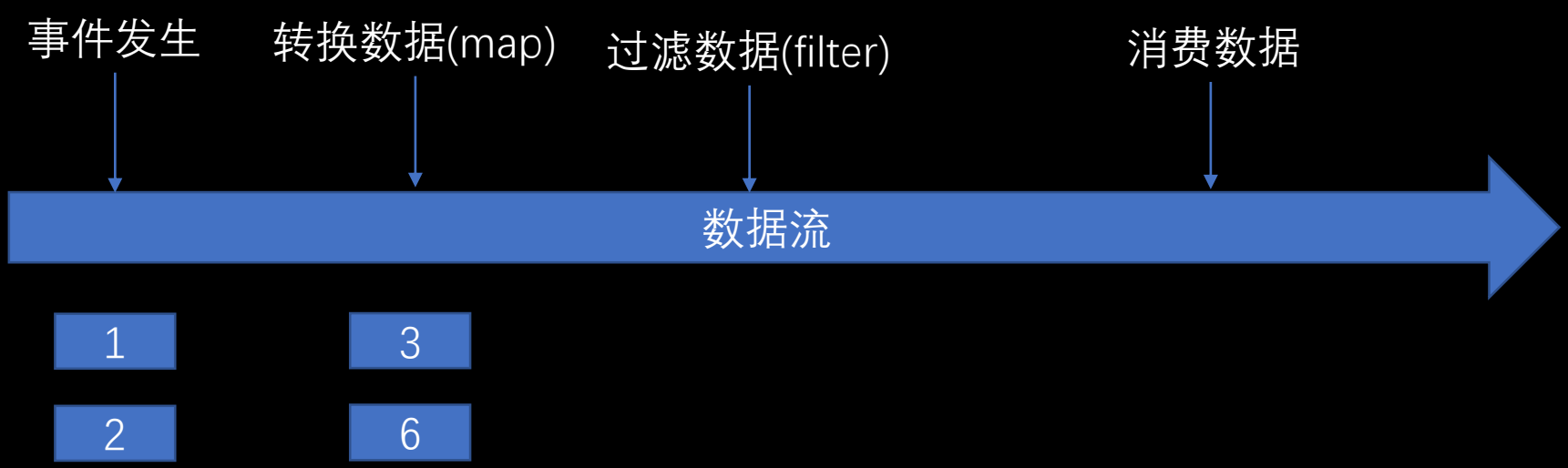
- （通知和绑定机制）代码复杂，增加工作量
- 代码和数据非一一对应关系
  - 需要**修改**已有逻辑

过滤脏话

- 当发送内容包含脏话时，不显示在界面上

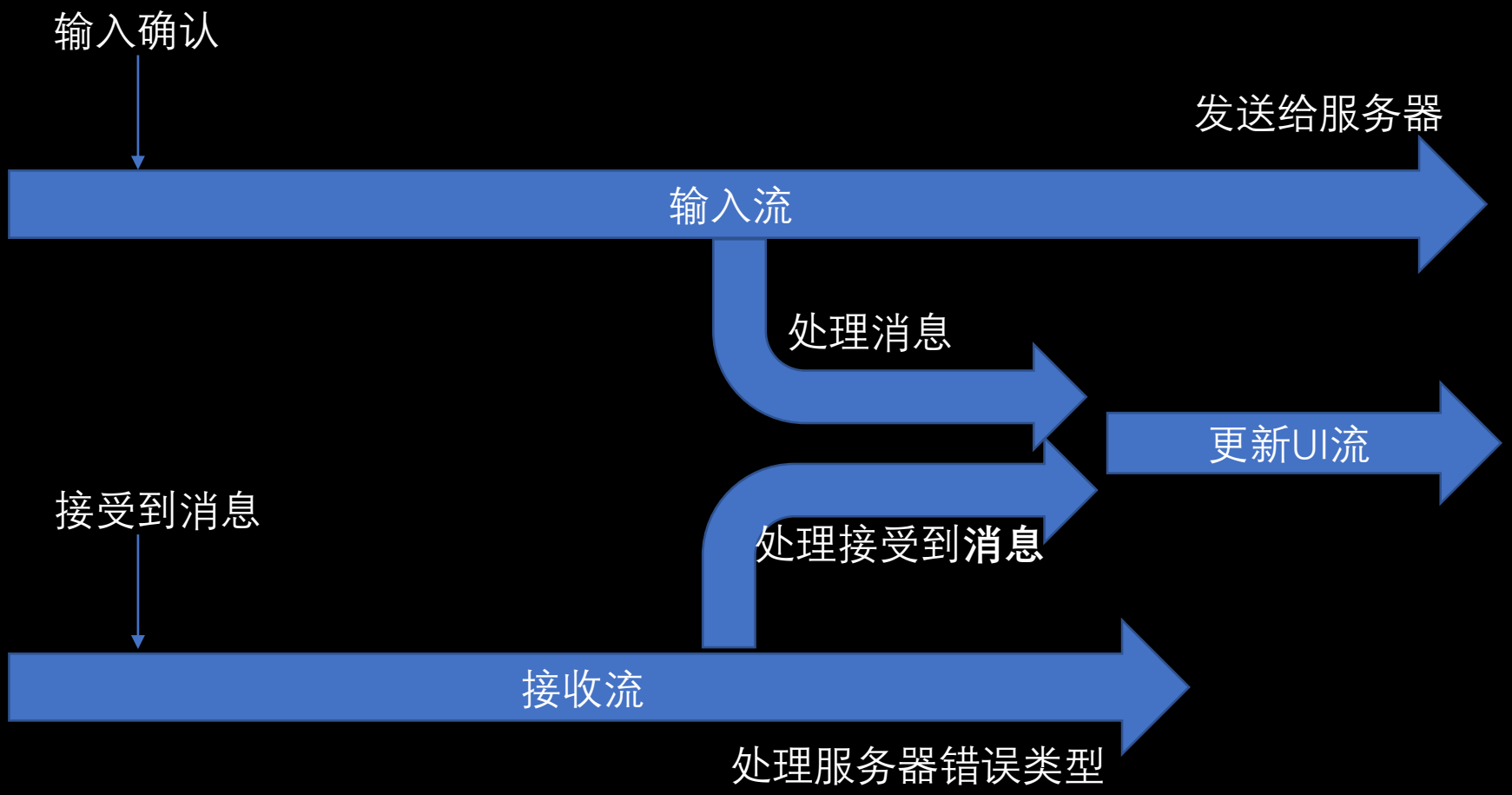
Functional Reactive Programming

- FRP
- 将事件抽象为流，在流上增加**处理函数（Functional）**
- 当事件发生时，**自动按顺序执行**处理函数（Reactive）

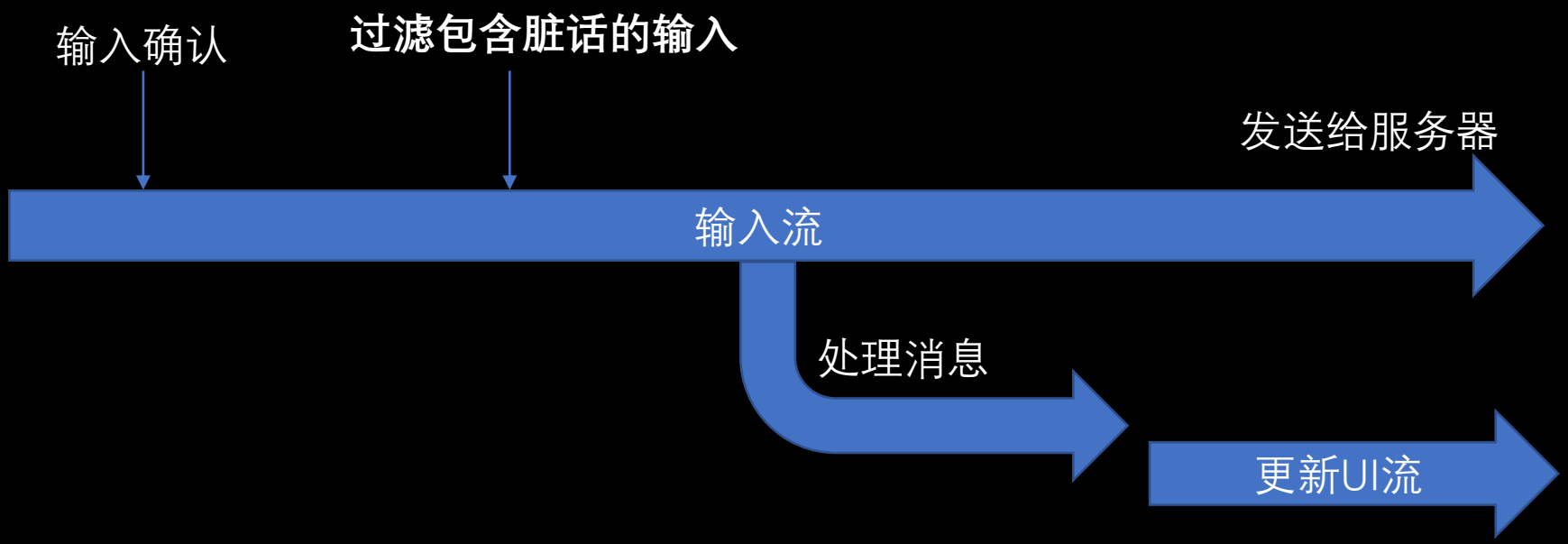


使用FRP实现消息处理

- 输入流，接收流

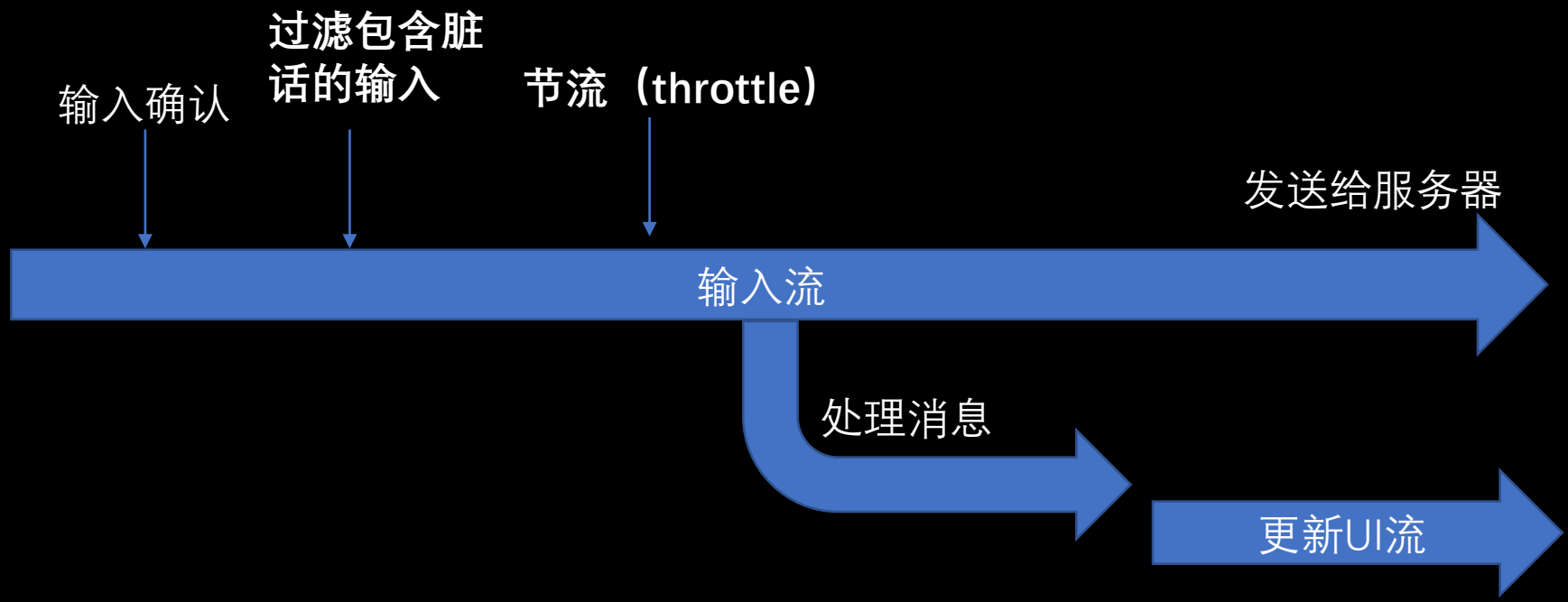


使用FRP实现过滤脏话



使用FRP节流函数

- 节流（throttle）：一段时间内只能调用某函数有限次数
- 1s内只能发送一条信息



FRP优势和劣势

- 优势
  - 非常方便的消息处理，可扩展性极佳
    - Map, filter, flatMap, throttle, reduce, first, merge…
    - 适合应对来源多样、数据处理复杂多变等情景
  - 简洁的代码
- 劣势
  - 原理较为复杂
  - 函数过多且使用易出错
  - 使用场景受限

作业（三选一）

- MVP：在MVP中，除了之前提到的便于测试之外，由于将业务逻辑和View分离，增加功能也只需要**增加一个Presenter和少量修改View**即可实现。尝试在MVP-0的基础上实现以下功能：**当所发送的信息为这样的格式时（即markdown的图片），将信息显示为对应的图片。**
- MVVM：当消息过多时，有的消息可能没有在当前屏幕上显示出来。尝试在MVVM-0的基础上实现如下功能：**显示不在当前屏幕中显示出来的消息数量，并随着用户滚动消息列表、发送和接受消息实时更新。（Tip：考虑绑定消息列表的当前位置属性）**
- FRP：使用现有的XML界面文件，使用流从头实现基本需求，**并实现功能3（撤回消息）**

谢谢