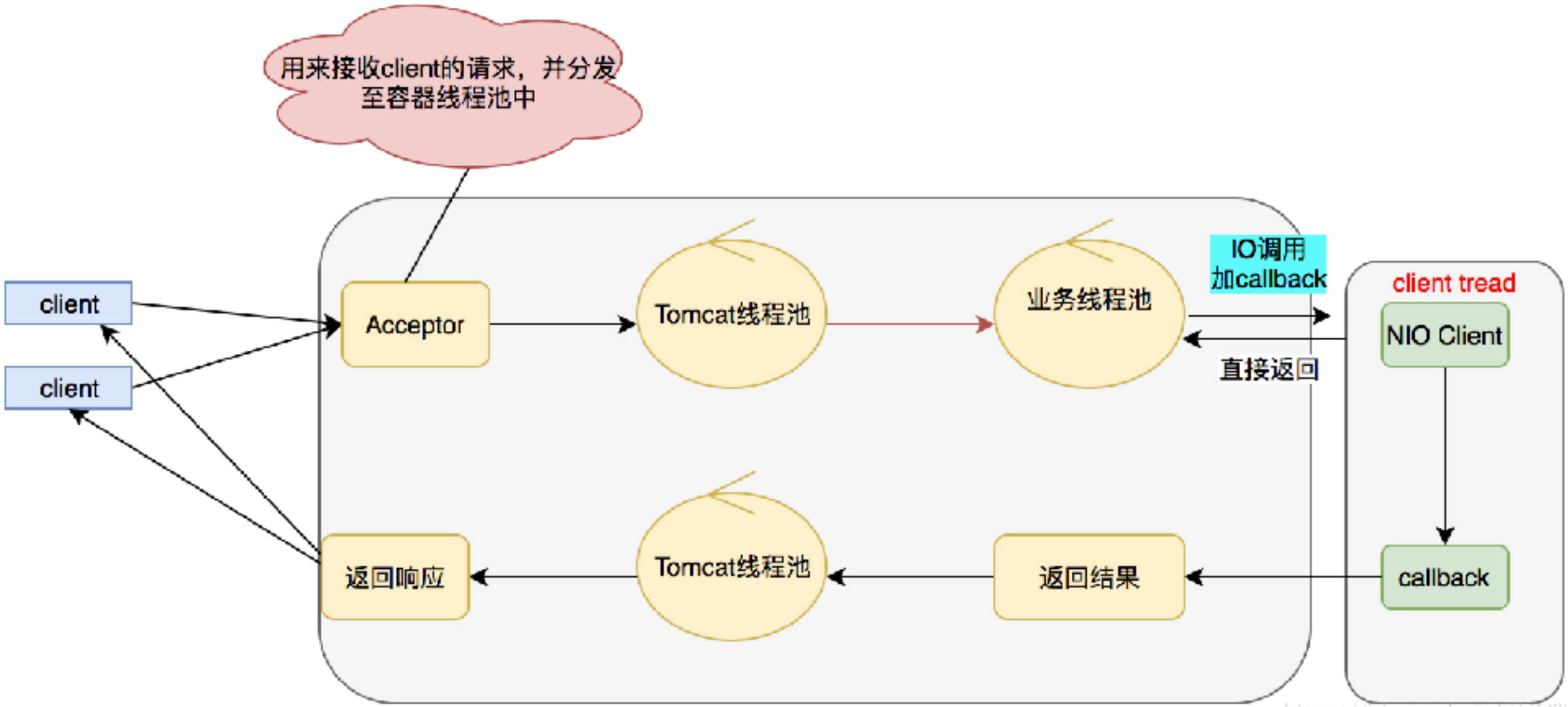
Spring WebFlux 第二弹

黄阳全

WebFlux处理流程



WebFlux处理流程

- webflux的请求链路？

响应式编程

- 响应式编程 和 Project Reactor
- webflux流程
- 异步非阻塞IO

响应式编程

响应式编程 和 Project Reactor

响应式编程

定义

响应式编程是一种关注于数据流（data streams）和

变化传递（propagation of change）的异步编程方式

响应式编程

Reactor

```
userService.getFavorites(userId) ❶
           .flatMap(favoriteService::getDetails)  ❷
           .switchIfEmpty(suggestionService.getSuggestions())  ❸
           .take(5)  ❹
           .publishOn(UiUtils.uiThreadScheduler())  ❺
           .subscribe(uiList::show, UiUtils::errorPopup);  ❻
```

响应式编程

响应式编程规范，又来了

```
public interface Publisher<T> {
    public void subscribe(Subscriber<? super T> s);
}

public interface Subscriber<T> {
    public void onSubscribe(Subscription s);
    public void onNext(T t);
    public void onError(Throwable t);
    public void onComplete();
}

public interface Subscription {
    public void request(long n);
    public void cancel();
}

public interface Processor<T,R> extends Subscriber<T>, Publisher<R> {
}
```

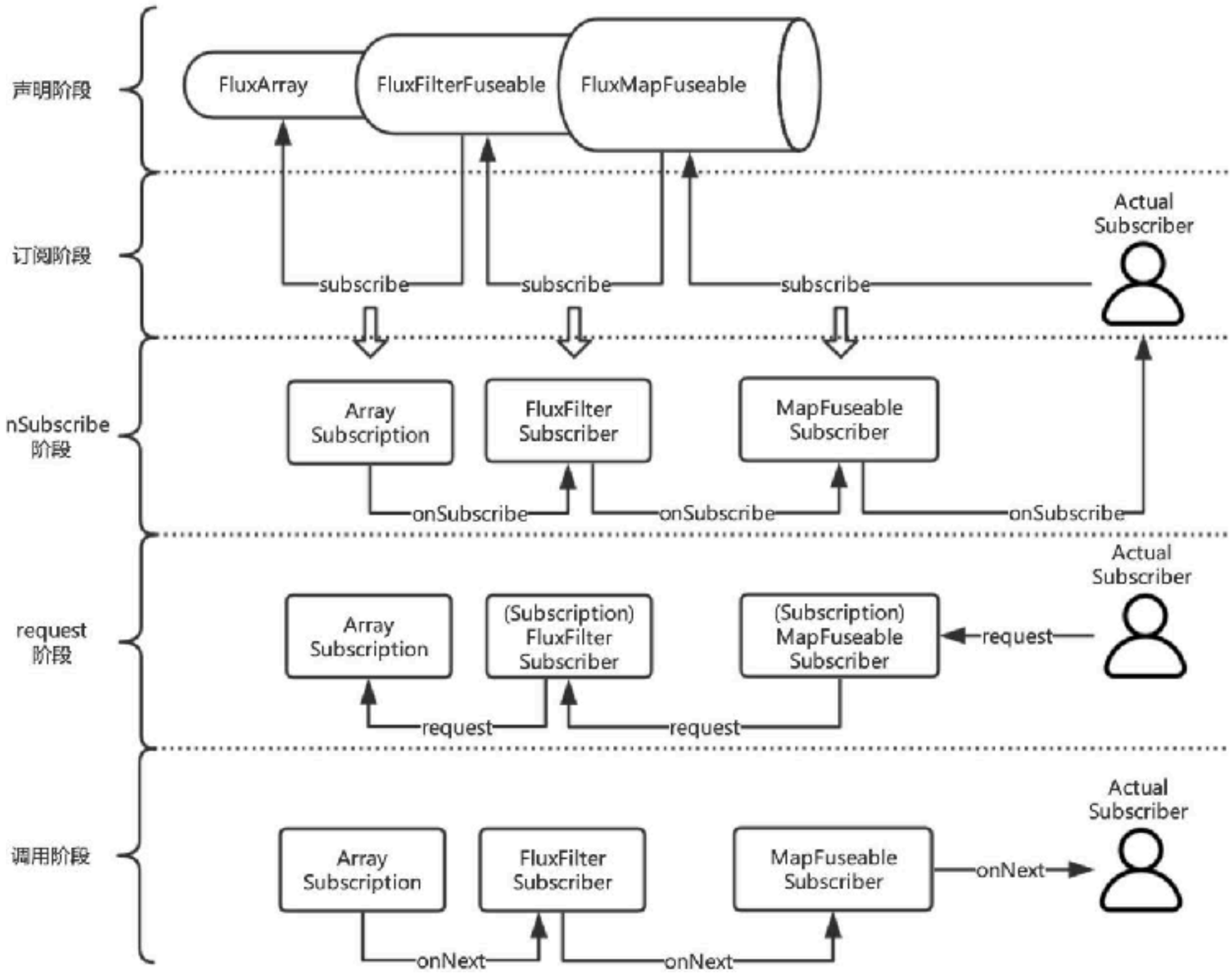
响应式编程

- 声明阶段
- subscribe 阶段
- onSubscribe 阶段
- request 阶段
- 调用阶段

响应式编程

```
Flux.just("tom", "jack", "allen")
        .filter(s -> s.length() > 3)
.map(s -> s.concat(" hello"))
.subscribe(System.out::println);
```

响应式编程


声明阶段
订阅阶段
nSubscribe
阶段
request
阶段
调用阶段
FluxArray
FluxFilterFuseable
FluxMapFuseable
Actual
Subscriber
subscribe
subscribe
subscribe
Array
Subscription
FluxFilter
Subscriber
MapFuseable
Subscriber
onSubscribe
onSubscribe
onSubscribe
Actual
Subscriber
Array
Subscription
(Subscription)
FluxFilter
Subscriber
(Subscription)
MapFuseable
Subscriber
request
request
request
Actual
Subscriber
Array
Subscription
FluxFilter
Subscriber
MapFuseable
Subscriber
onNext
onNext
onNext

响应式编程


Flux.just
发布者
onNext
订阅者
request(n)
.map
发布者
onNext
订阅者
request(n)
Subscriber

响应式编程

```
Mono<xxx>.xxx()
         .xxx()
         .subscribe(subscriber);
```

响应式编程

- webflux的是如何结合reactor实现异步化的?

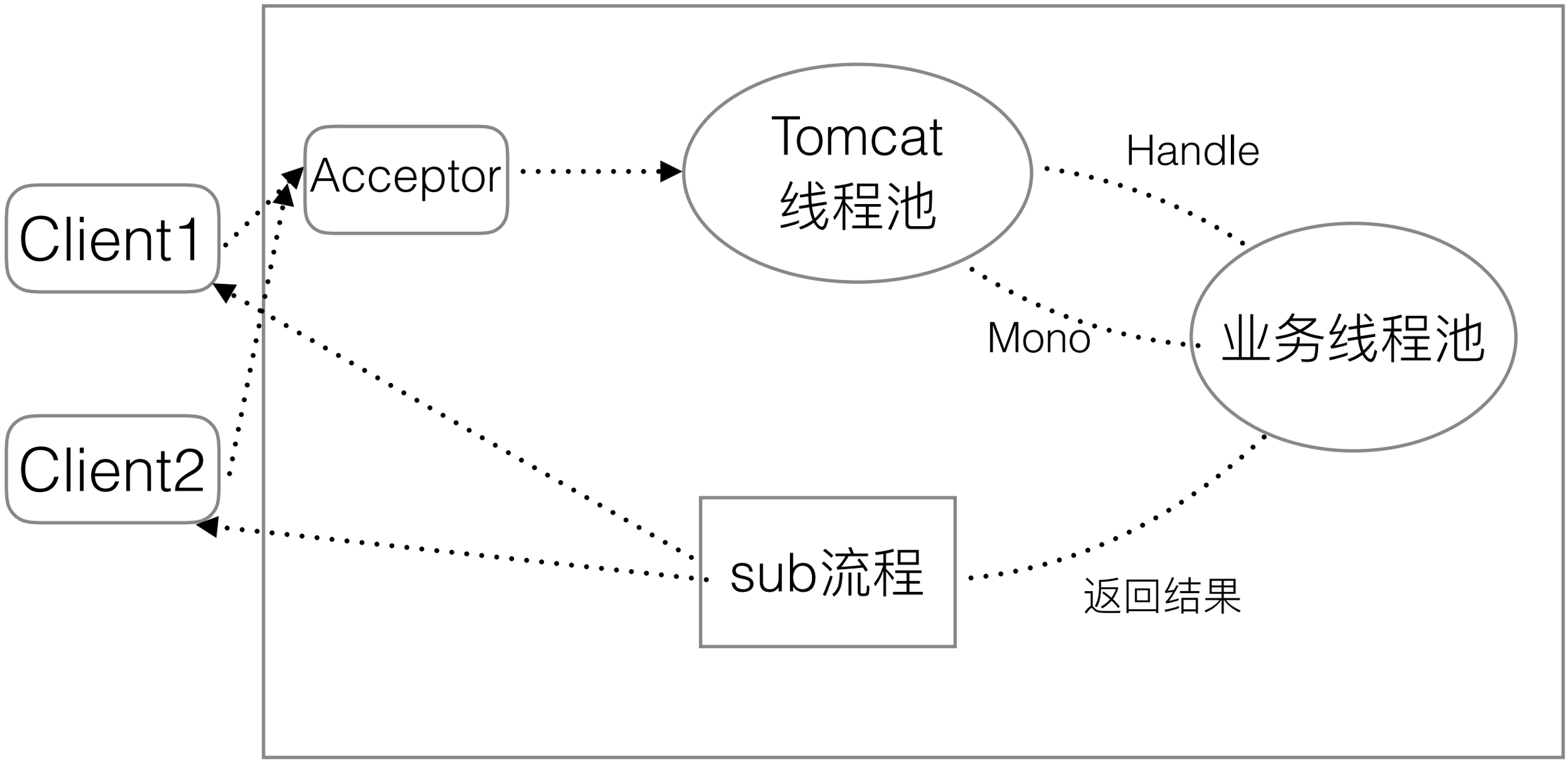
WebFlux处理流程

```
// org.springframework.http.server.reactive.ServletHttpHandlerAdapter#service


public void service(ServletRequest request, ServletResponse response) throws ServletException, IOException {
    // Check for existing error attribute first
    // ...

    // Start async before Read/WriteListener registration
    AsyncContext asyncContext = request.startAsync();
    asyncContext.setTimeout(-1);

    // 构建ServletServerHttpRequest和 ServerHttpResponse
    // ...

    AtomicBoolean isCompleted = new AtomicBoolean();
    HandlerResultAsyncListener listener = new HandlerResultAsyncListener(isCompleted, httpRequest);
    asyncContext.addListener(listener);

    HandlerResultSubscriber subscriber = new HandlerResultSubscriber(asyncContext, isCompleted, httpRequest);
    this.httpHandler.handle(httpRequest, httpResponse).subscribe(subscriber);
}
```

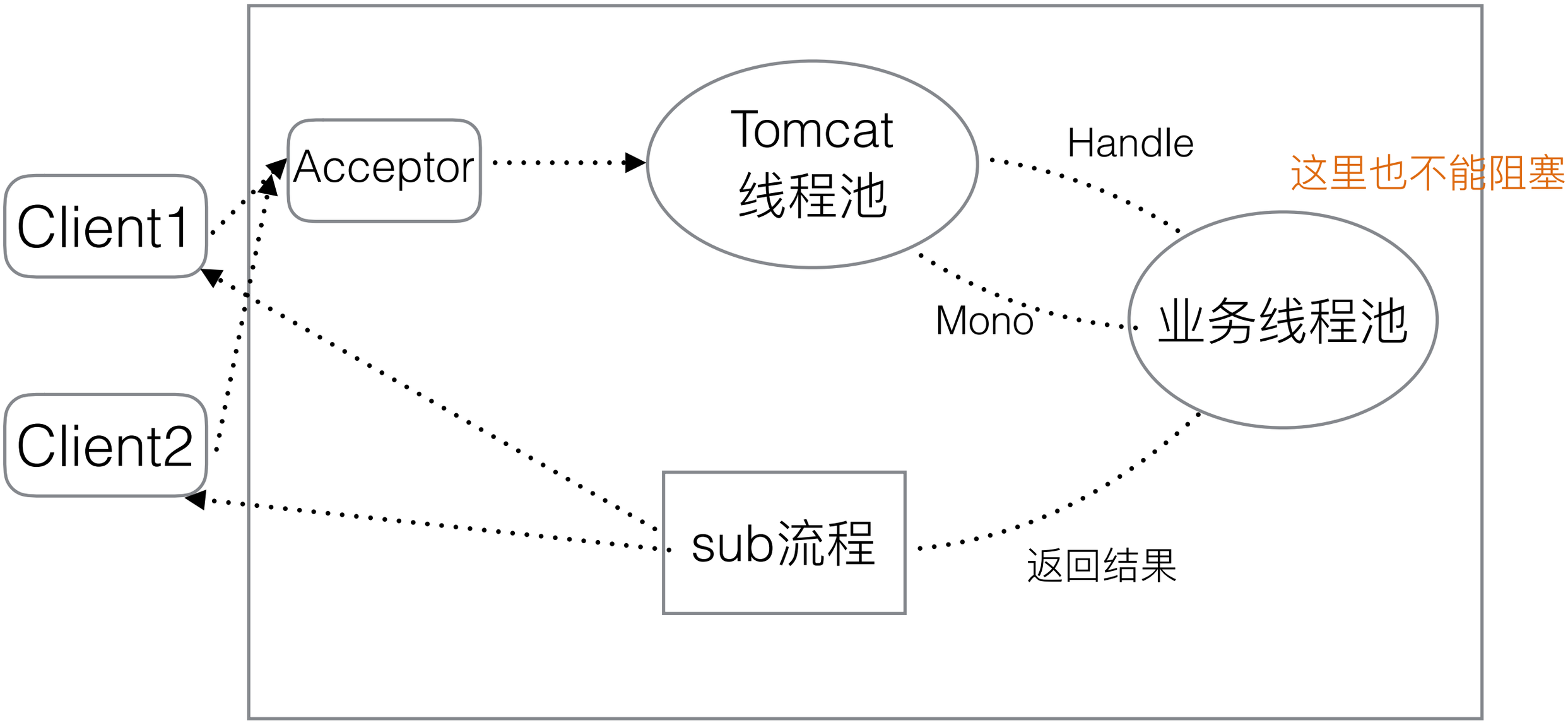
WebFlux处理流程

```
this.httpHandler
    .handle(httpRequest, httpResponse)
    .subscribe(subscriber);
```

WebFlux处理流程

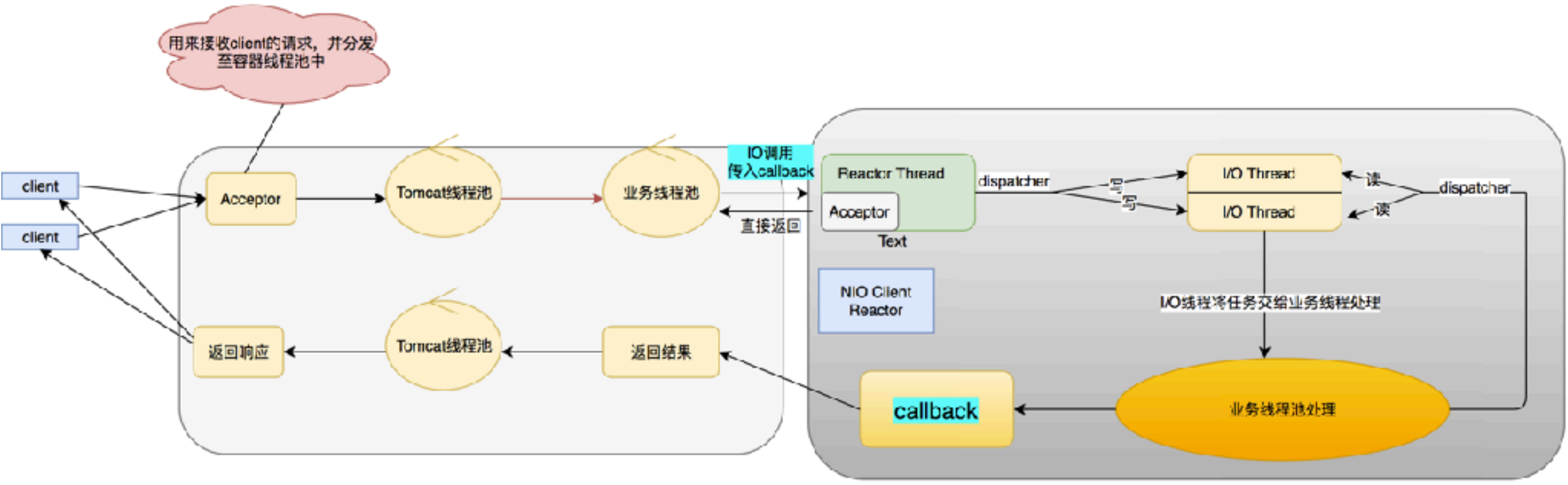


WebFlux处理流程

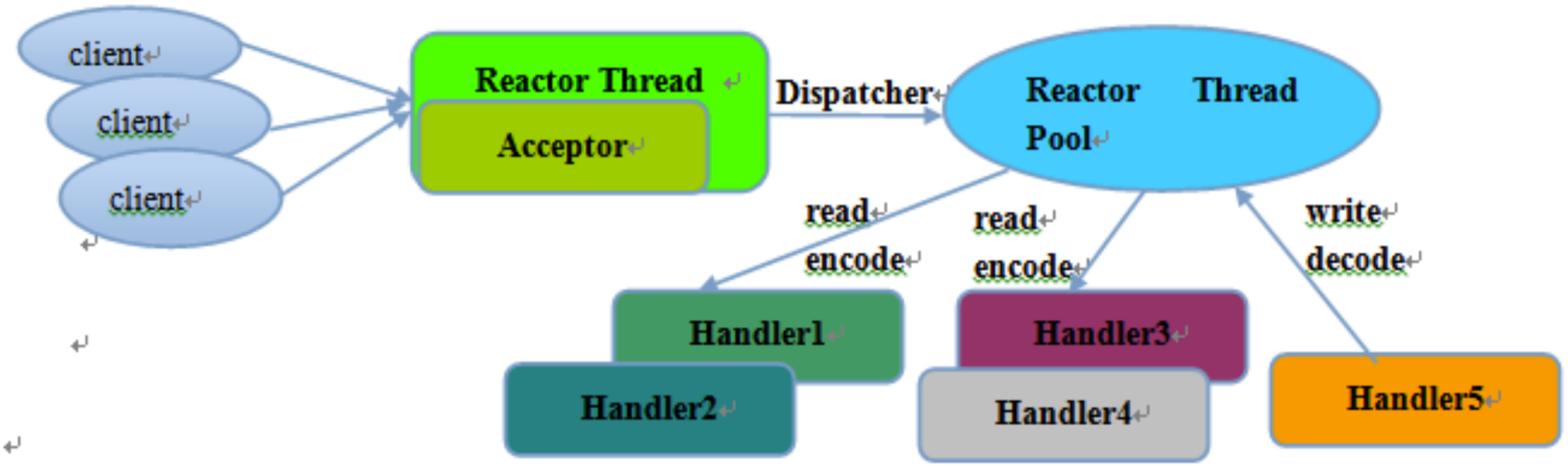


异步非阻塞IO

WebFlux处理流程



异步非阻塞IO


client
client
client
Reactor Thread
Acceptor
Dispatcher
Reactor Thread Pool
read
encode
read
encode
write
decode
Handler1
Handler2
Handler3
Handler4
Handler5

异步非阻塞IO

- http调用异步化
- rpc调用
- 数据库调用
- 缓存调用异步化

总结

总结

Webflux借助reactor实现了spring mvc级别的异步化

WebClient等实现了资源、远程调用的异步化

谢谢！