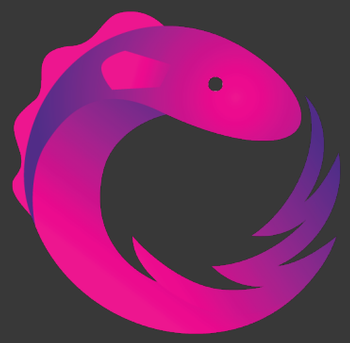
Learning Rx

by example

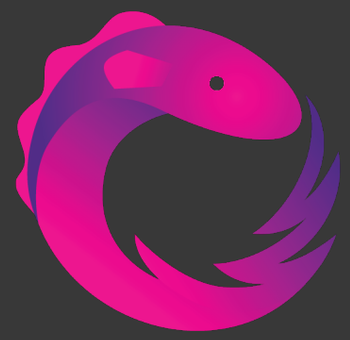


1 2 mt intro to Rx

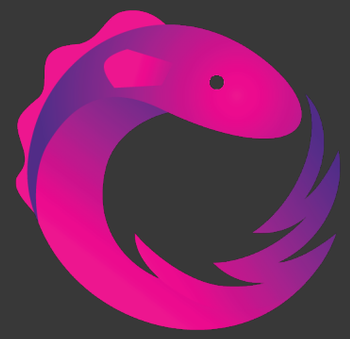
2 3 (intermediate) examples

2 mt

Intro to Rx



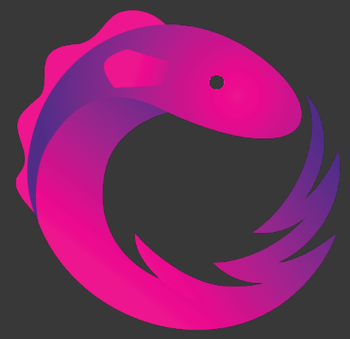
API for Asynchronous programming



Observer pattern done right

Best ideas from:

Observer pattern
Iterator pattern
Functional programming



Observer pattern done right

```
Observable.just(1)
        .useSomeFunkyOperators()
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(getSubscriber());
```

Example 1

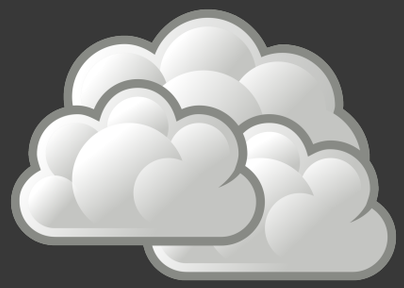
Loading from Disk Cache + Network Call

Requirement:

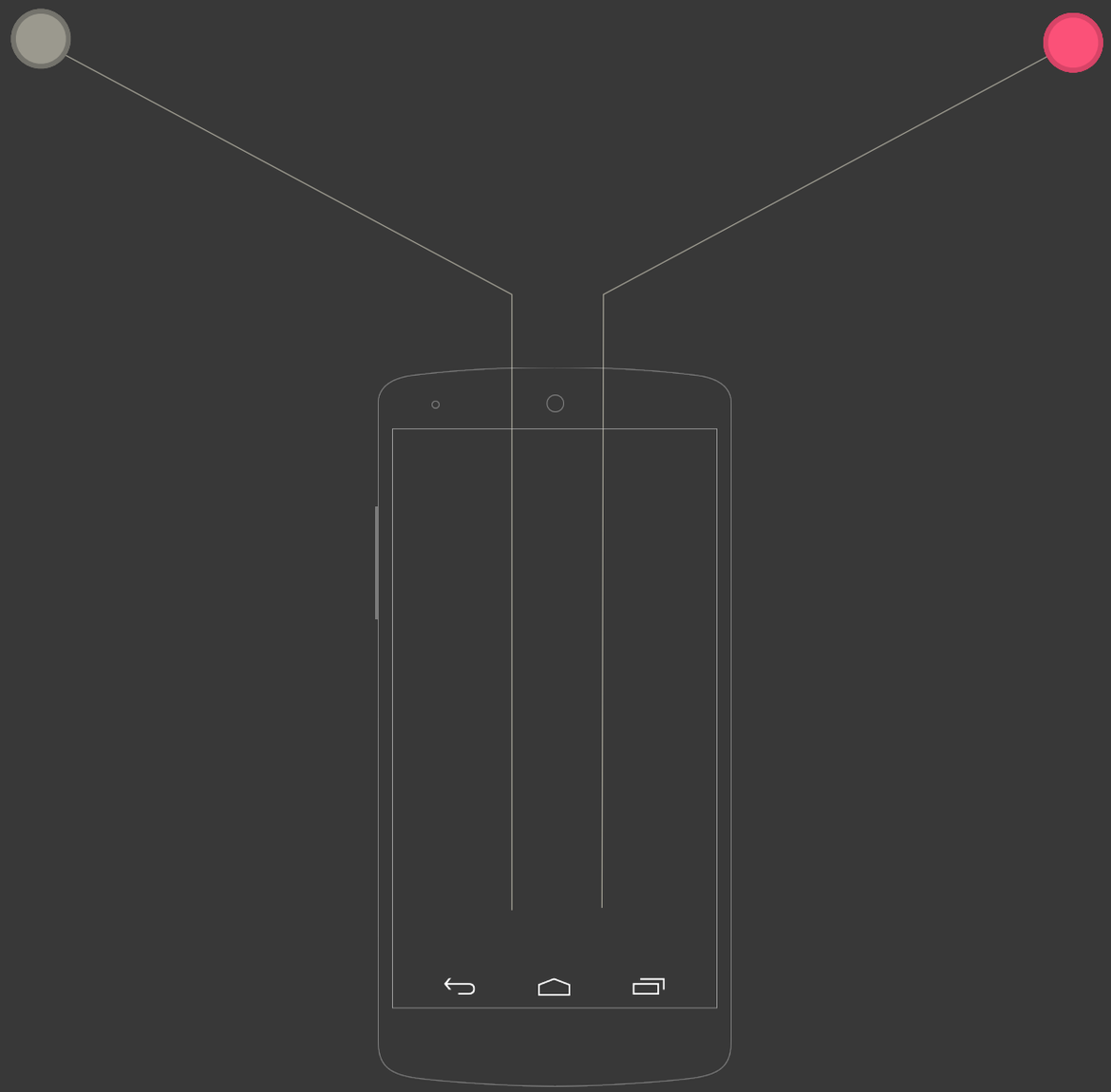
Database

Cached data
Fast

Server

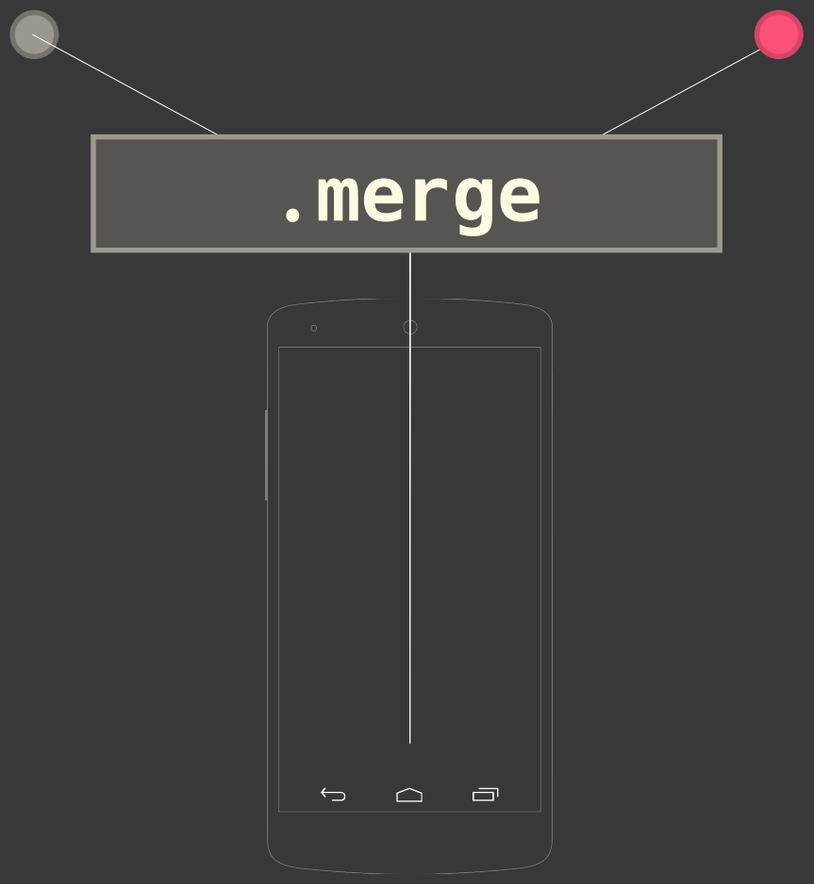


Fresh data
Slow



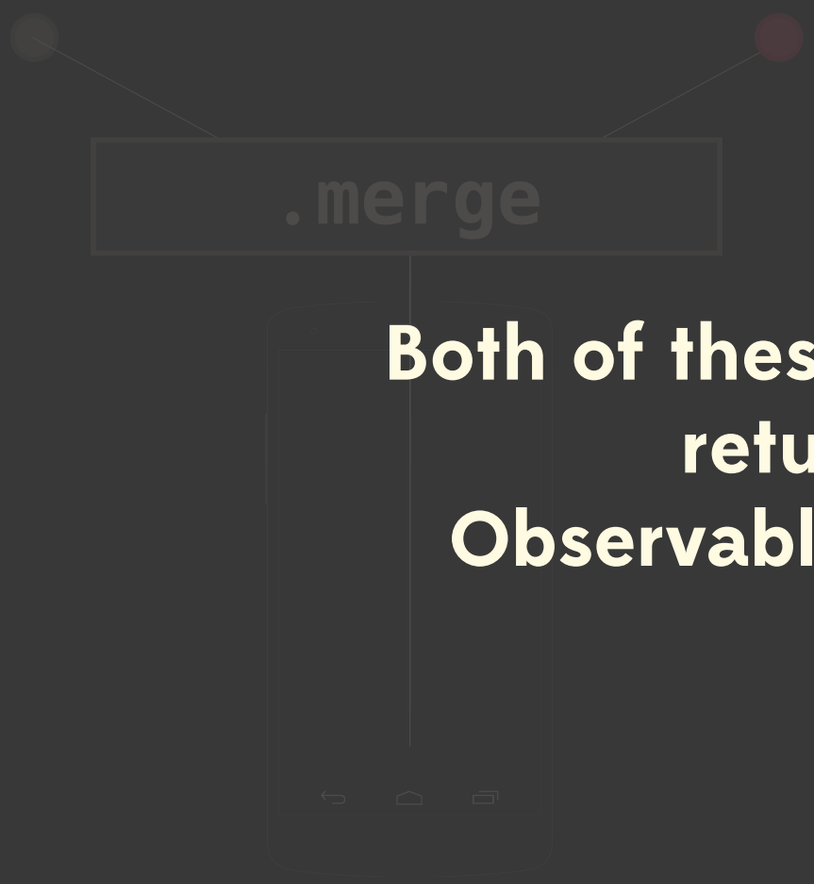
Example 1

Code:




```
Observable
        .merge(
                getDiskResults(),
                getNetworkResults())

.subscribeOn(Schedulers.io())
.observeOn(AndroidSchedulers.mainThrd())
.subscribe();
```

Code:



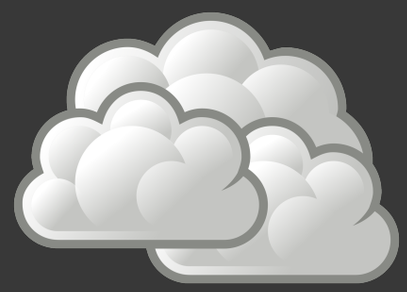
Both of these methods return Observable<Result>

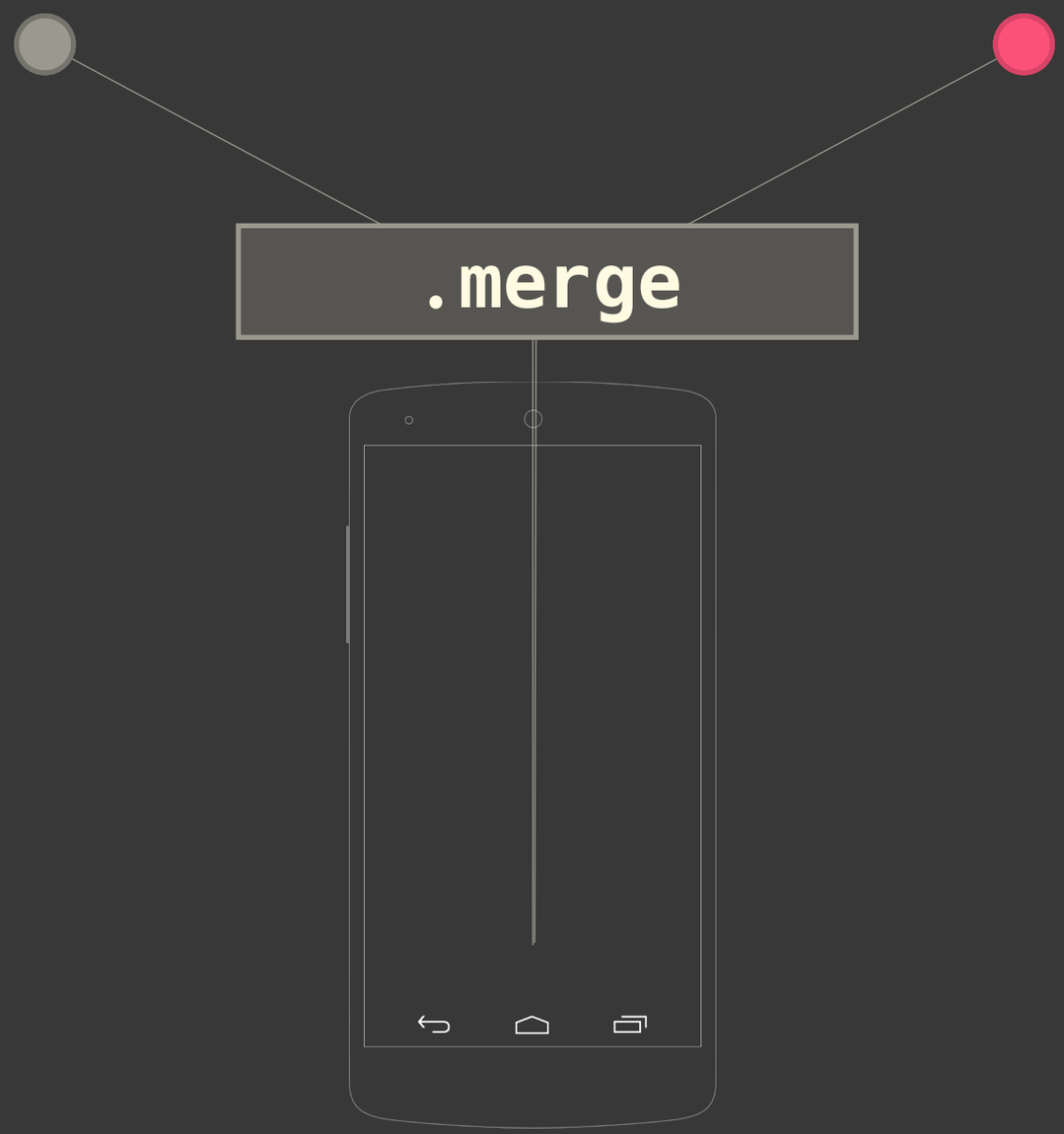
```
Observable
        .merge(
                getDiskResults(),
                getNetworkResults())

        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThrd())
        .subscribe();
```

Problem:

Database

Network request



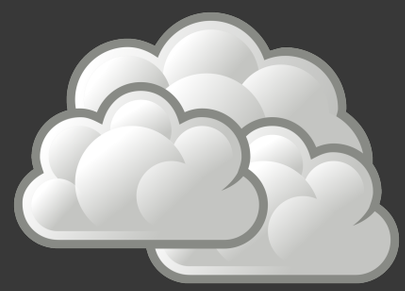


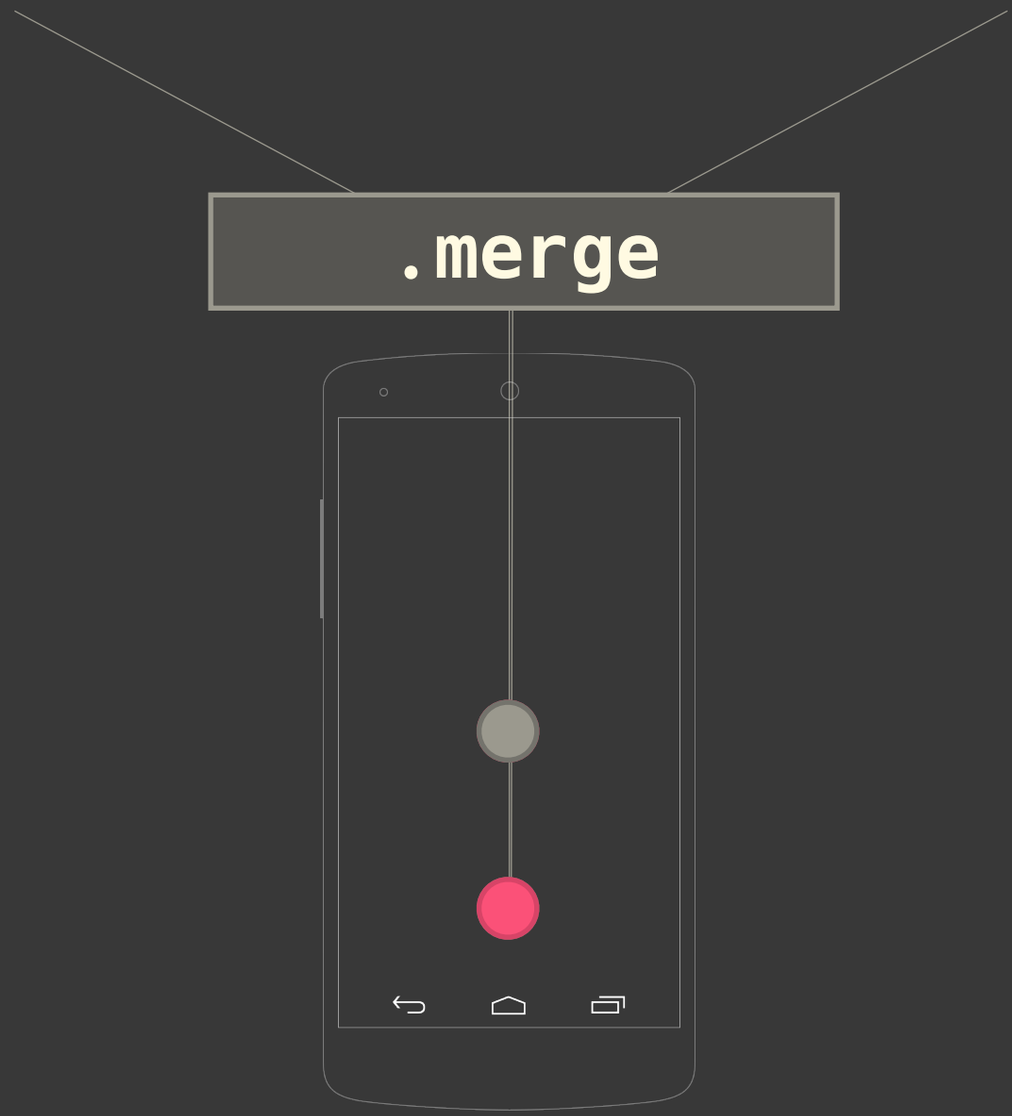
Example 1

Problem:

Database

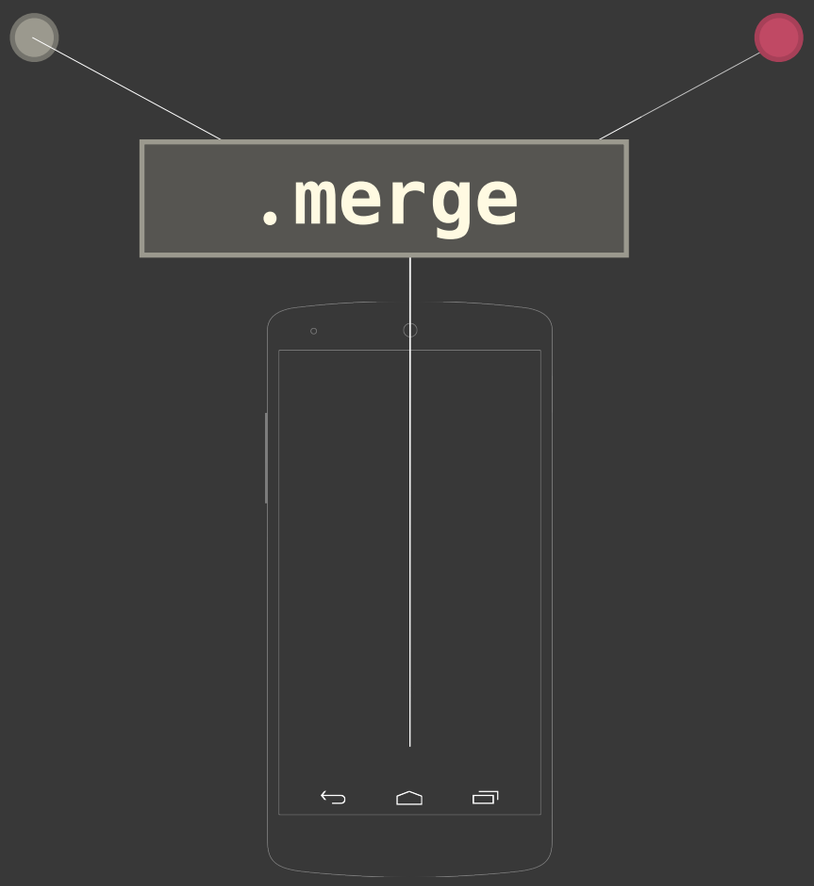
Network request





Example 1

Code:




```
Observable
        .merge(
                getDiskResults(),
                getNetworkResults())

.subscribeOn(Schedulers.io())
.observeOn(AndroidSchedulers.mainThrd())
.subscribe(new Subscriber<Result>() {
   @Override
    public void onCompleted() { //... }

   @Override
    public void onError(Throwable e) { //... }

   @Override
    public void onNext(Result result) {
     if (  list.contains(result) &&
           isExistingResultFromNetwork(result))
            return;

     // usual "add result to list" logic
     list.add(result);
     list.refresh();
    }
});
```

Example 1

Code:

Observable

.merge(

.merg

NO!

The Rx is not strong with this code

https://twitter.com/JakeWharton/status/786363146990649345

Code:

```
getNetworkData()
        .publish(
            network ->
                Observable
                    .merge(
                        network,
                        getDiskData()
                            .takeUntil(network)
                    )
        )
```

Example 1

Code:

Get database results
but stop after network results

```
getDiskButStopAfterNetwork()
        =   getDiskResults()
                .takeUntil(getNetworkResults())
```

Code:

Get disk results that occur before Network starts

+

Network results

```
Observable
        .merge( getNetworkResults()
                getDiskButStopAfterNetwork()

              )


getDiskButStopAfterNetwork()
              =     getDiskResults()
                        .takeUntil(getNetworkResults())
```

Example 1

Code:

Get disk results that occur before Network starts

+

Network results

```
.publish(
    Observable
        .merge( getNetworkResults()
                getDiskResults()
                  .takeUntil(getNetworkResults())
              )
```

Example 1

Code:

Get disk results that occur before Network starts
+
Network results

```
getNetworkResults()
            .publish(
        network ->
            Observable
                .merge(getNetworkResults()
                    getDiskResults()  ,
                      .takeUntil(getNetworkResults())
                )
```

Code:

Get disk results that occur before Network starts
+
Network results

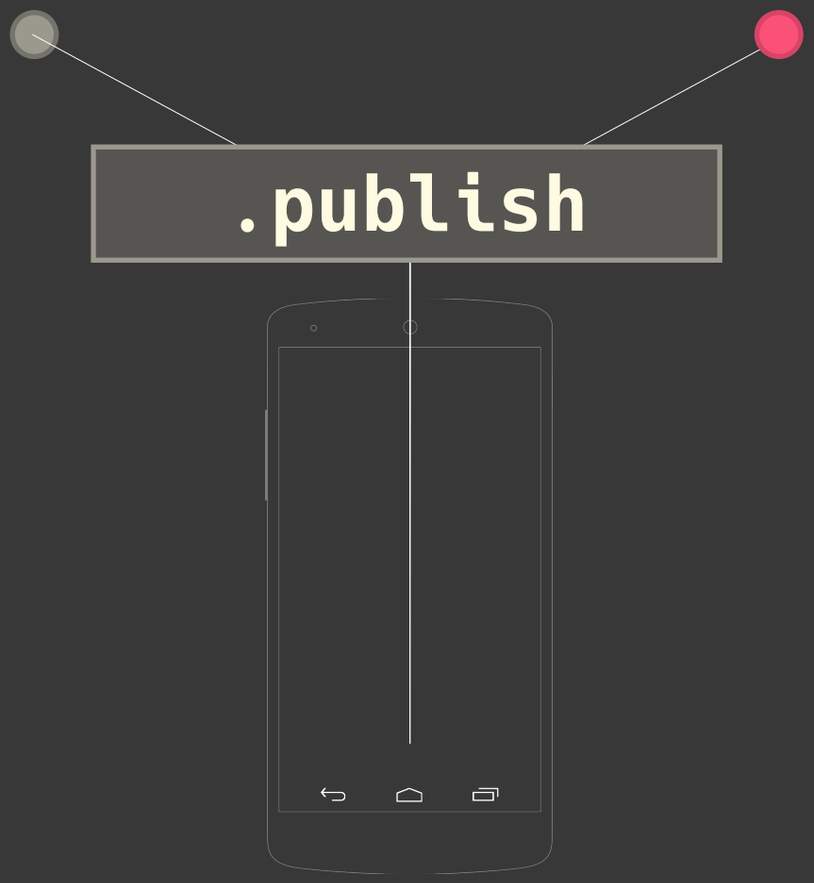
```
getNetworkResults()
            .publish(
        network ->
            Observable
                .merge(network,
                    getDiskResults()  ,
                      .takeUntil(getNetworkResults())
                )
```

Code:

Get disk results that occur before Network starts
+
Network results

```
getNetworkResults()
                .publish(
        network ->
            Observable
                    .merge(network,
                            getDiskResults()  ,
                              .takeUntil( network                    )
                    )
```

Code:




```
getNetworkData()
   .publish(
            network ->
               Observable
                     .merge(
                          network,
                          getDiskData()
                               .takeUntil(network)
                     )
      )

.subscribeOn(Schedulers.io())
.observeOn(AndroidSchedulers.mainThrd())
.subscribe();
```

Example 2

Pagination
(using Subjects)

Setup starting seq. once
(network call for results)

Observable

```
Observable
        .just(1)
        .flatMap(pageNo -> getNetworkResults(pageNo))
```

Holy trinity of RxConversion

map

R → T

flatmap

R → Observable<T>

compose

Observable<R> → Observable<T>

Setup starting seq. once
(network call for results)

Observable

Subscribe
here

```
Observable
        .just(1)
        .flatMap(pageNo -> getNetworkResults(pageNo))


private Observable<List<Item>> getNetworkResults(int pageNo) {
    // make network request
    // get List of Items for page number
}
.subscribe(new Subscriber<List<Item>>() {
    @Override
    public void onCompleted() {
        // all items downloaded
    }

    @Override
    public void onError(Throwable e) {
        // handle error
    }

    @Override
    public void onNext(List<Item> items) {
     addToList(items);
    }
});
```

Example 2b

Setup starting seq. once
(network call for results)

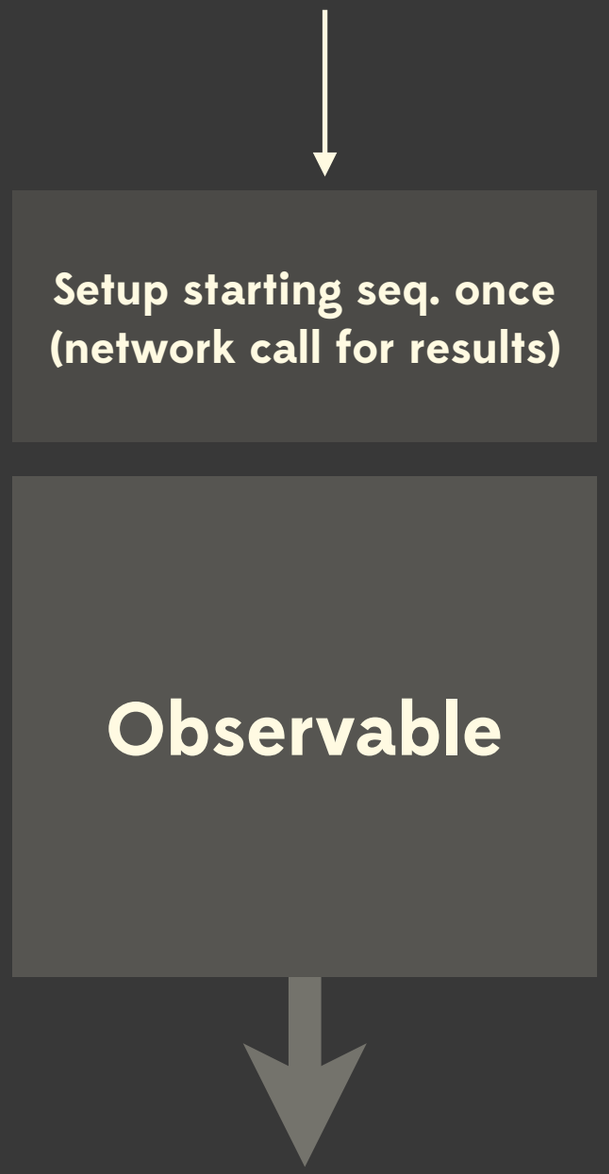
Observable

Subscribe
here

```
Observable
        .just(1)
        .concatMap(pageNo -> getNetworkResults(pageNo))
```

```
.subscribe(new Subscriber<List<Item>>() {
    @Override
    public void onCompleted() {
        // all items downloaded
    }

    @Override
    public void onError(Throwable e) {
        // handle error
    }

    @Override
    public void onNext(List<Item> items) {
     addToList(items);
    }
});
```

Example 2b

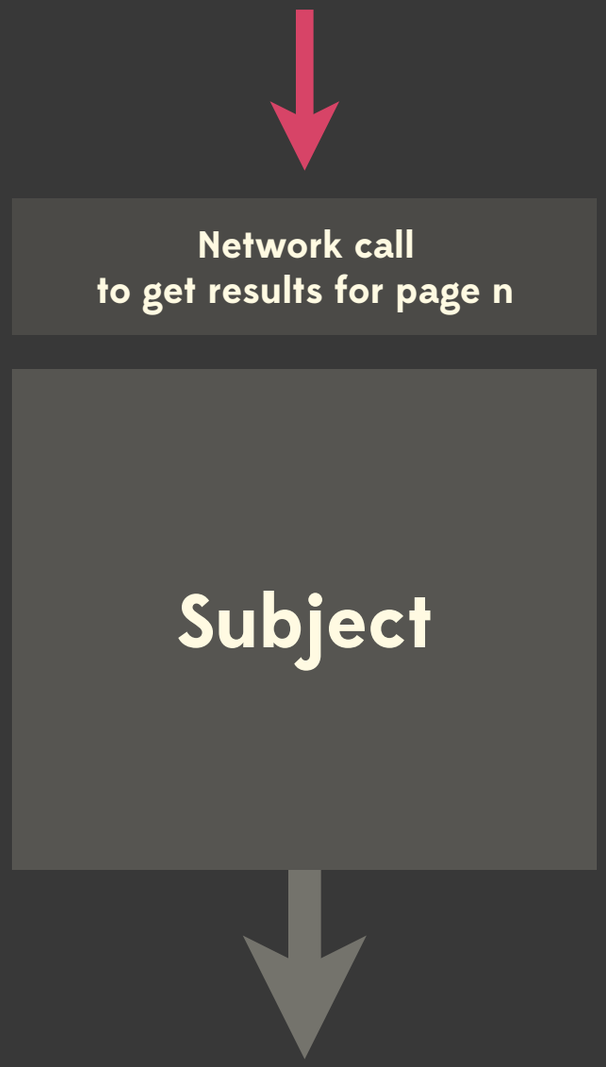


What if I could keep feeding
inputs to get the
next set of results?

like page numbers!

Subscribe here

Code:




```
PublishSubject<Integer> paginator =
                                    PublishSubject.create();

Observable
 paginator.start(1)
        .concatMap(pgNo -> getNetworkResults(pgNo))

        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new Subscriber<List<Item>>() {
            @Override
            public void onCompleted() {
                // all items downloaded
            }

            @Override
            public void onError(Throwable e) {
                // handle error
            }

            @Override
            public void onNext(List<Item> items) {
                addToList(items);
            }
        });
```

Example 2b

Code:

```
void onReachedEndOfList() {
    //...
    paginator.onNext(nextPage);
}
```

Network call to get results for page n

Subject

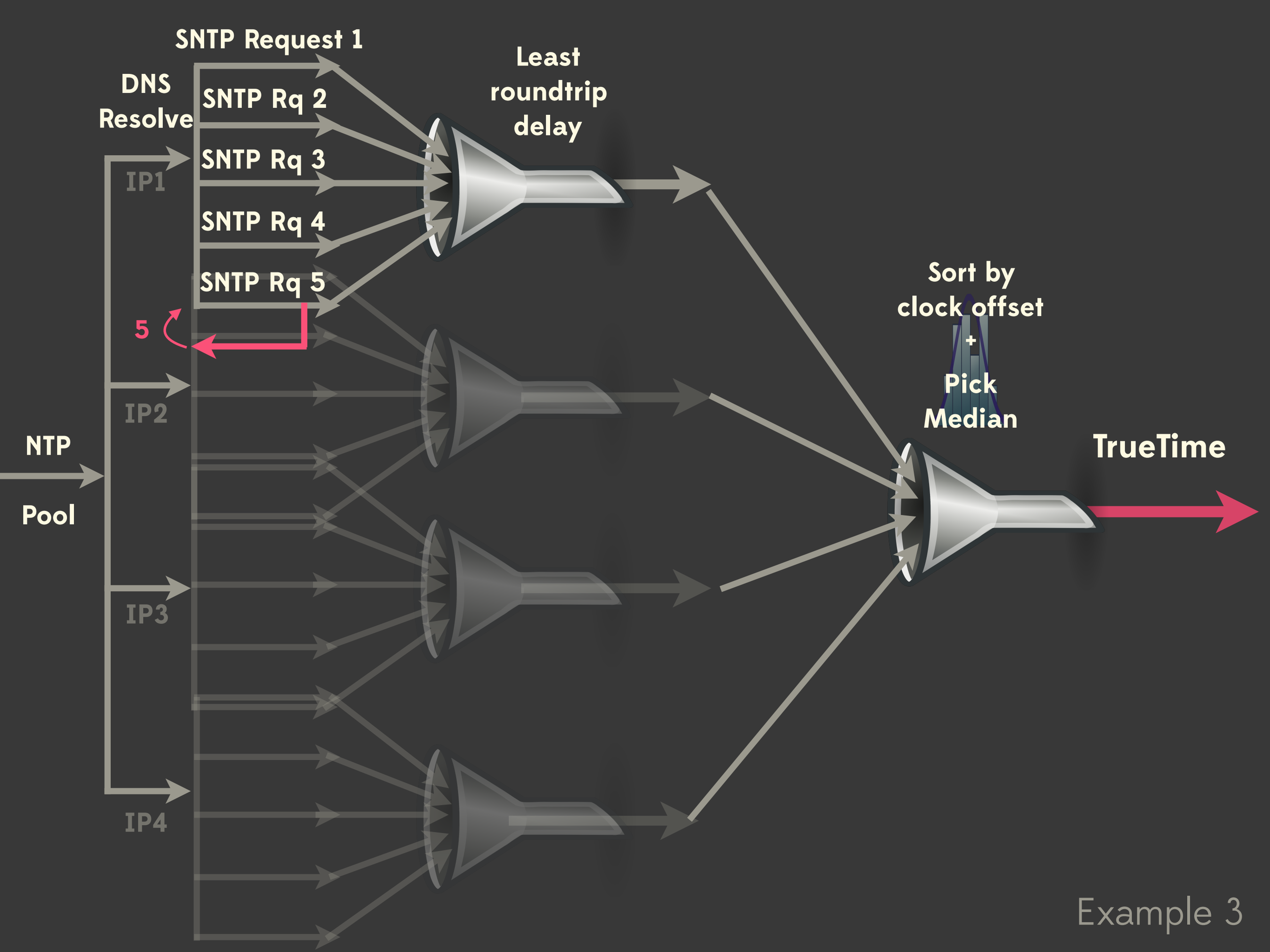
Subscribe here

```
paginator
        .concatMap(pgNo -> getNetworkResults(pgNo))

        .observeOn(AndroidSchedulers.mainThread())
.subscribe(new Subscriber<List<Item>>() {
    @Override
    public void onCompleted() {
        // all items downloaded
    }

    @Override
    public void onError(Throwable e) {
        // handle error
    }

    @Override
    public void onNext(List<Item> items) {
       addToList(items);
    }
});
```

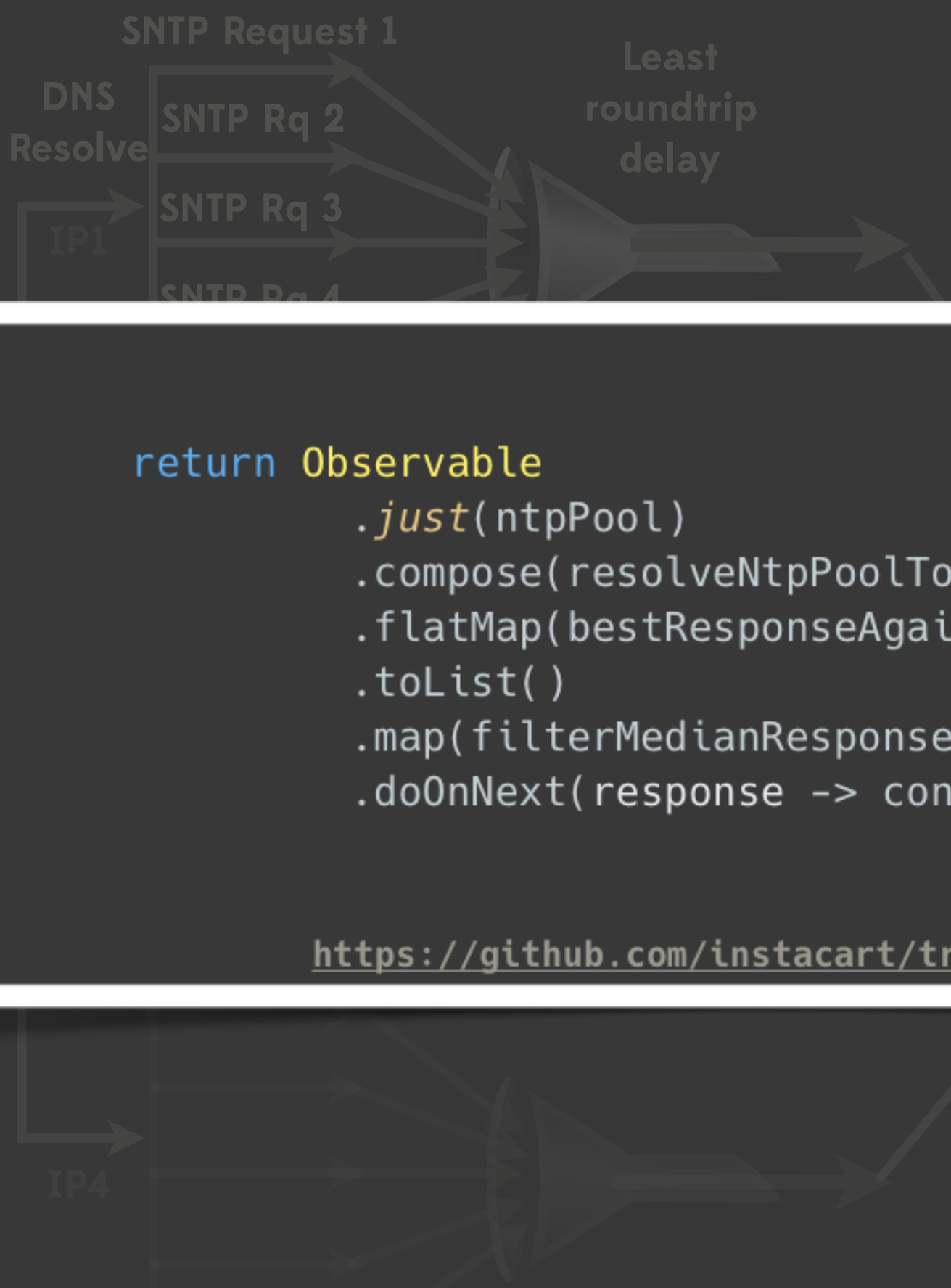
Example 3

TrueTime: Implementing NTP with Rx

github.com/instacart/truetime-android


SNTP Request 1
DNS Resolve
SNTP Rq 2
SNTP Rq 3
SNTP Rq 4
SNTP Rq 5
Least roundtrip delay
IP1
5
IP2
NTP
Pool
IP3
IP4
Sort by clock offset
+
Pick Median
TrueTime

Example 3

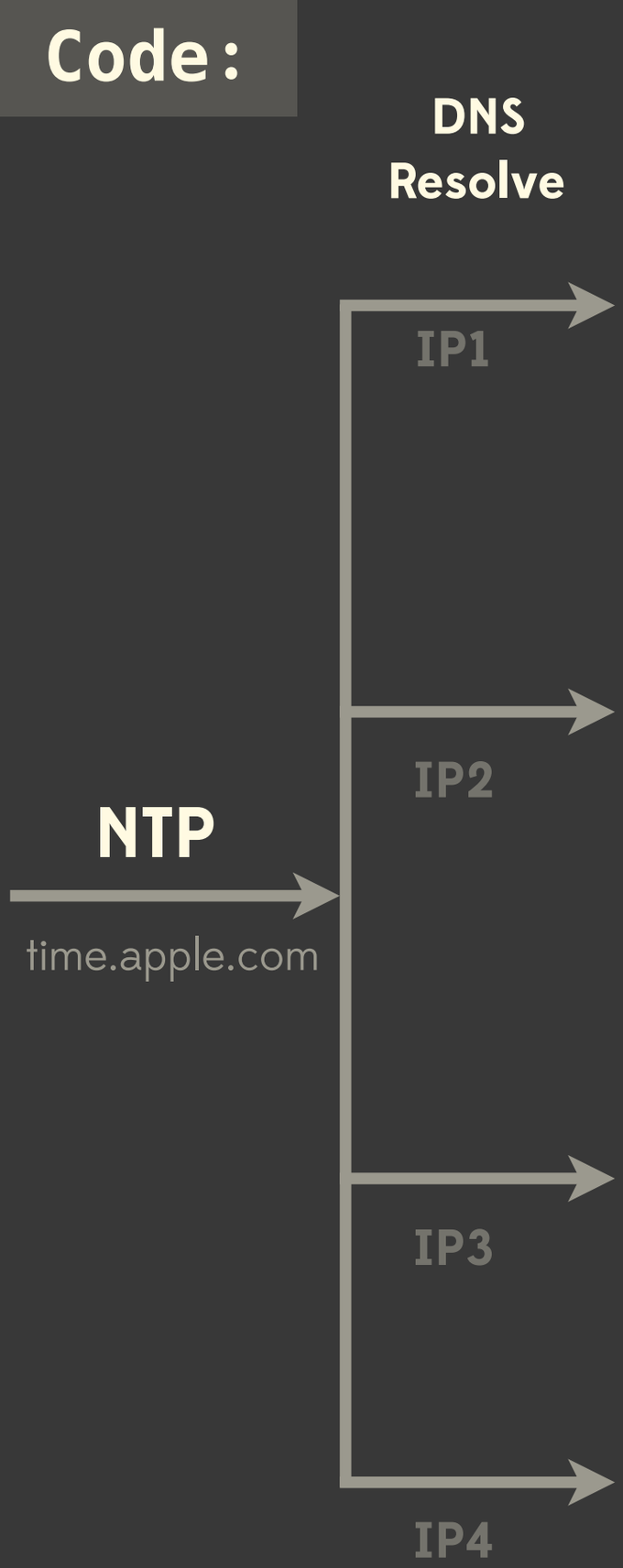



```
return Observable
        .just(ntpPool)
        .compose(resolveNtpPoolToIpAddresses())
        .flatMap(bestResponseAgainstSingleIp(5))
        .toList()
        .map(filterMedianResponse())
        .doOnNext(response -> convertToTime(response));
```

https://github.com/instacart/truetime-android

Code:

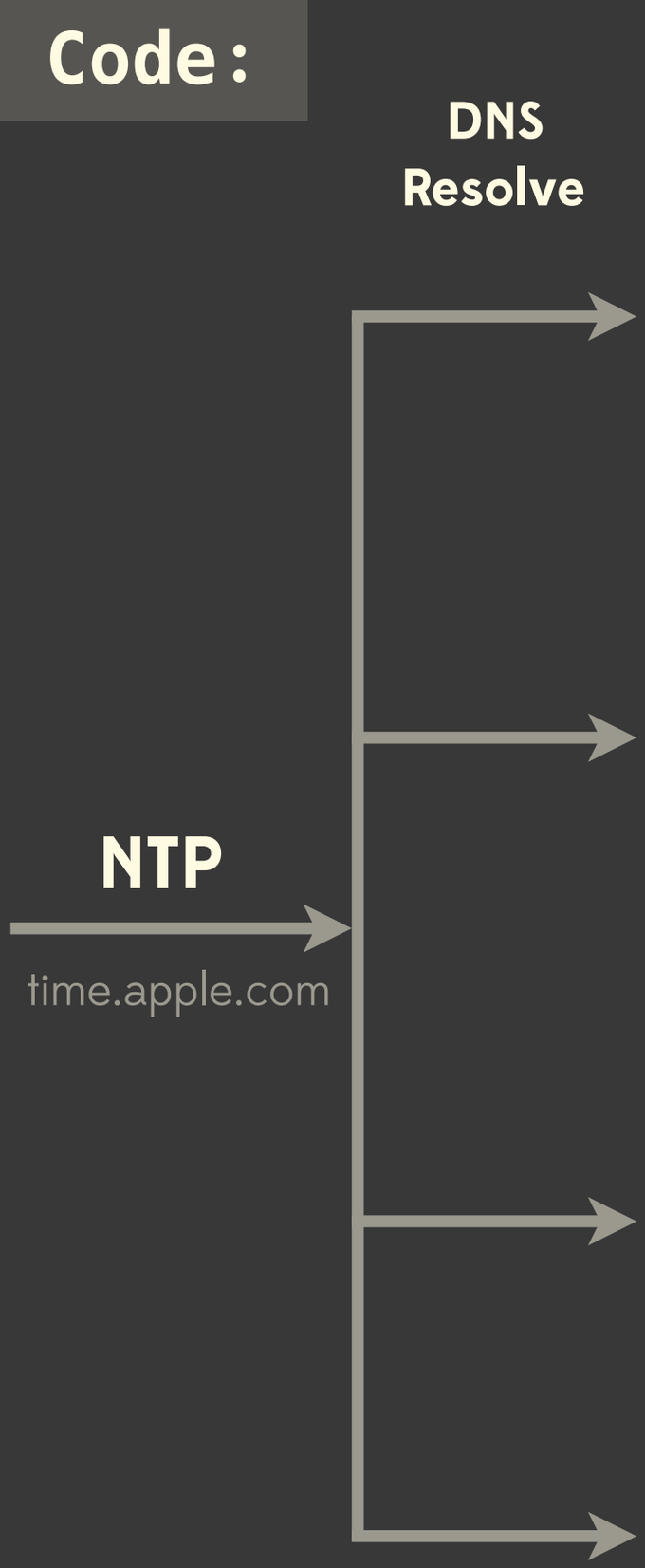
```
Observable
    .just(ntpPool)
    .compose(resolveNtpPool())
```



compose

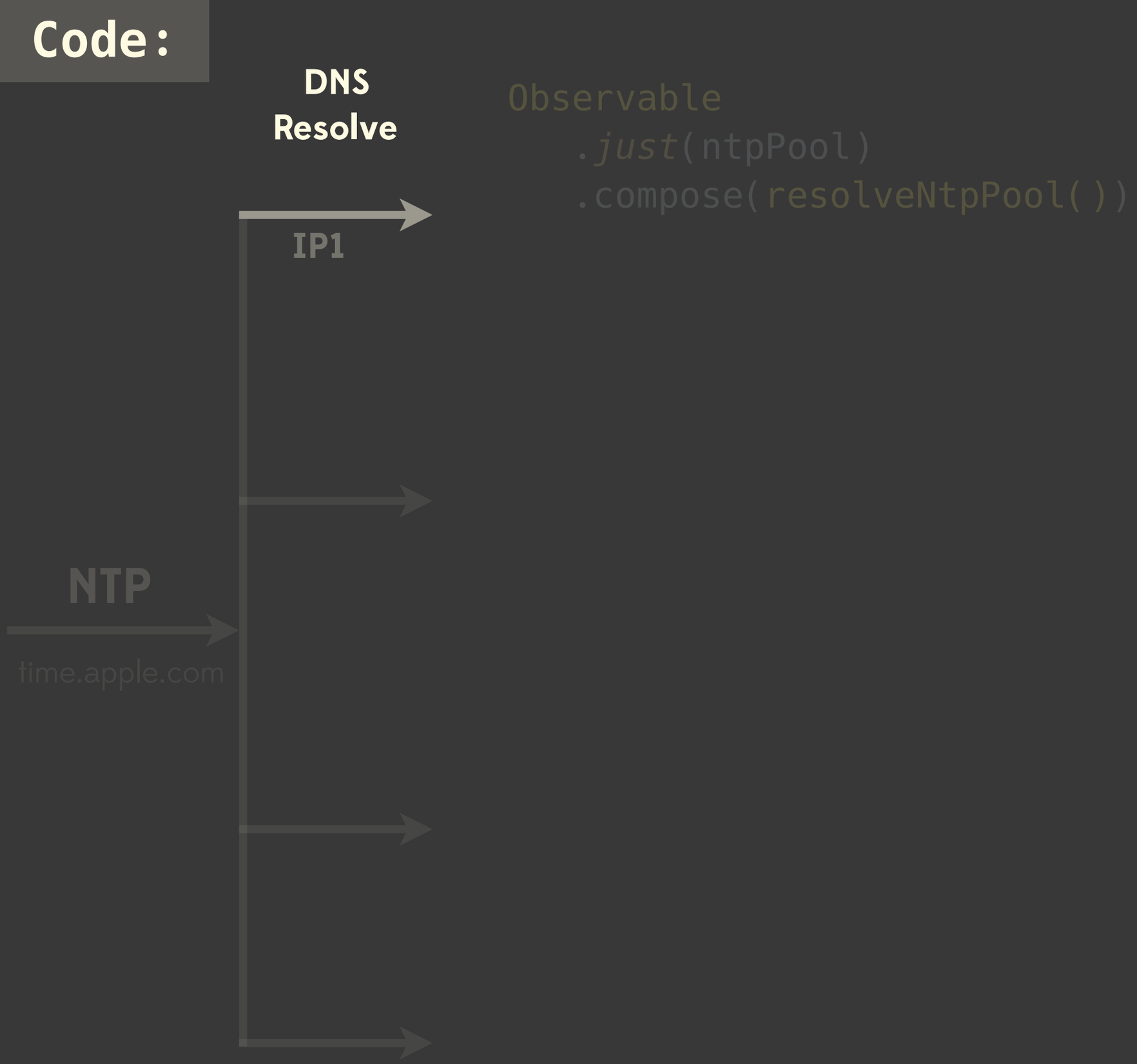
Observable‹Type1› → Observable‹Type 2›

Code:




```
Observable
    .just(ntpPool)
    .compose(resolveNtpPool())


private Transformer<String, String> resolveNtpPool(){
    return ntpPool -> {
        try {
            return Observable.from(
                    InetAddress.getAllByName(ntpPool));
        } catch (UnknownHostException e) {
            return Observable.error(e);
        }
    })
}
```

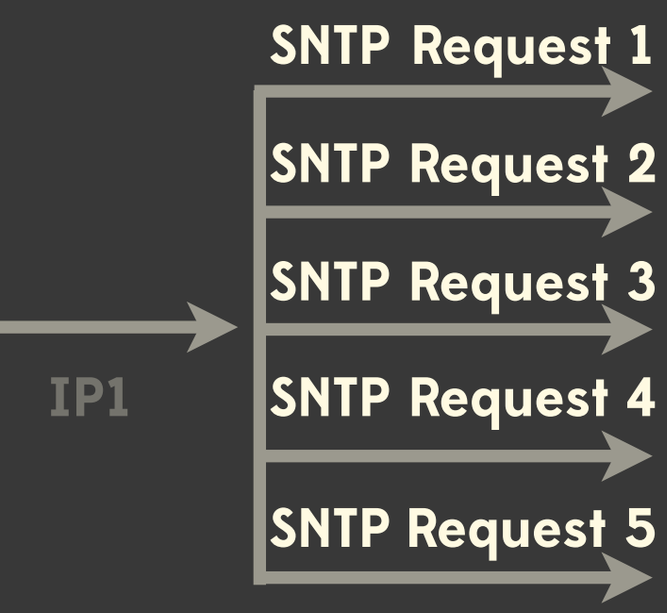
Code:




```
Observable
    .just(ntpPool)
    .compose(resolveNtpPool())
```

Code:

```
Observable
    .just(ntpPool)
    .compose(resolveNtpPool())
    .flatMap(bestResponseAgainstSingleIp())
```



Holy trinity of RxConversion

map

R → T

flatmap

T → Observable<T>

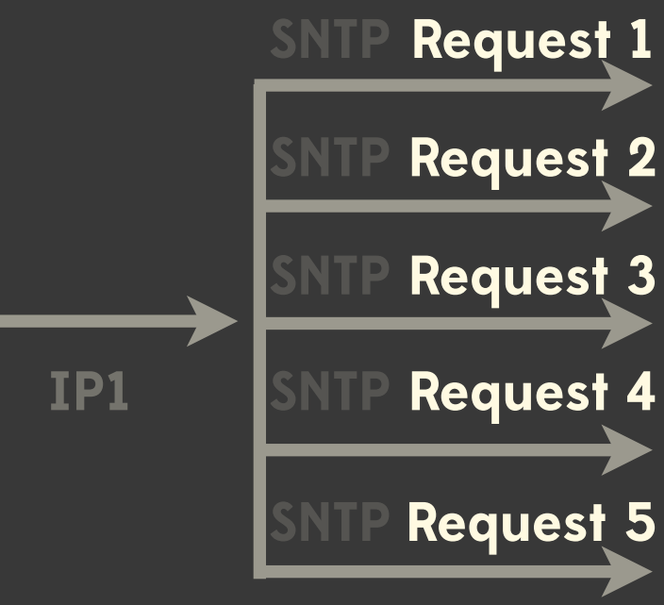
compose

Observable<R> → Observable<T>

Code:

```
Observable
    .just(ntpPool)
    .compose(resolveNtpPool())
    .flatMap(bestResponseAgainstSingleIp())
```

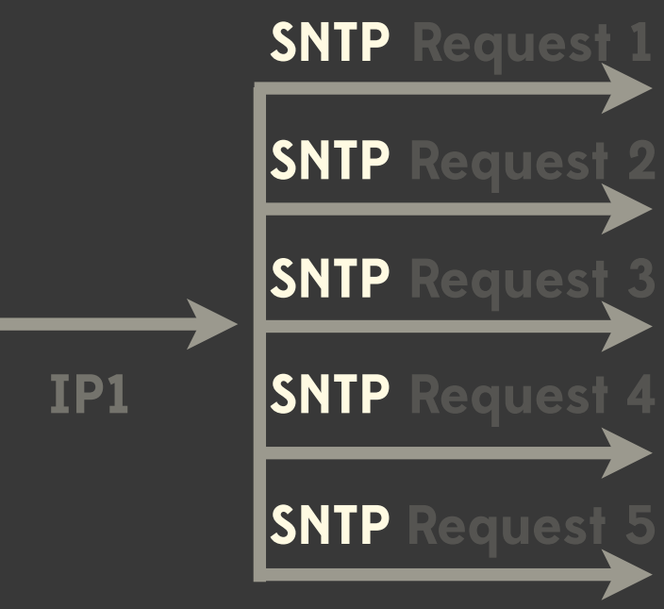
```
                    Func1 <String, Observable<Response>>
            bestResponseAgainstSingleIp()

    {   return Observable
                        .just(singleIp)
                        .repeat(5)
```



Code:

```
Observable
    .just(ntpPool)
    .compose(resolveNtpPool())
    .flatMap(bestResponseAgainstSingleIp())
```

```
                Func1 <String, Observable<Response>>
            bestResponseAgainstSingleIp()

    {   return Observable
                    .just(singleIp)
                    .repeat(5)

.flatMap(ipAddress ->
                sntpNtwrkReq(ipAddress))
```



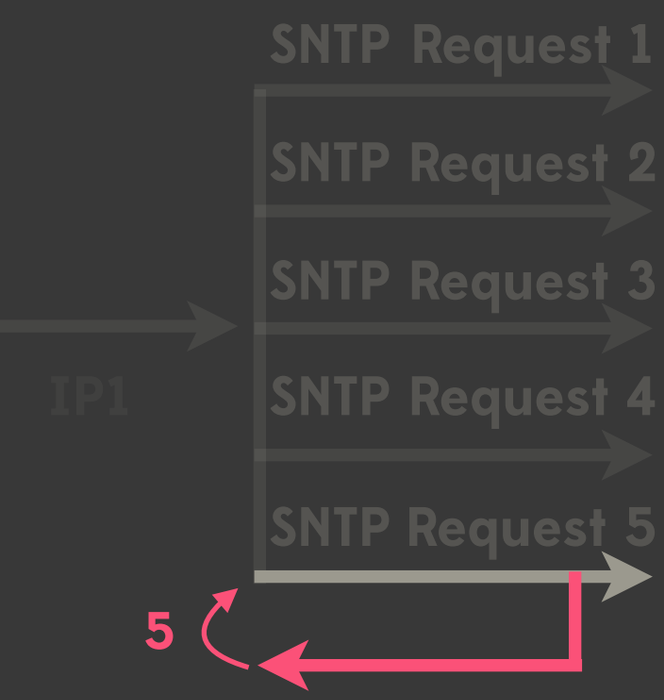
Code:

```
Observable
    .just(ntpPool)
    .compose(resolveNtpPool())
    .flatMap(bestResponseAgainstSingleIp())
```

```
Func1 <String, Observable<Response>>
bestResponseAgainstSingleIp()

{  return Observable
            .just(singleIp)
            .repeat(5)

    .flatMap(ipAddress ->
                sntpNtwrkReq(ipAddress))
```

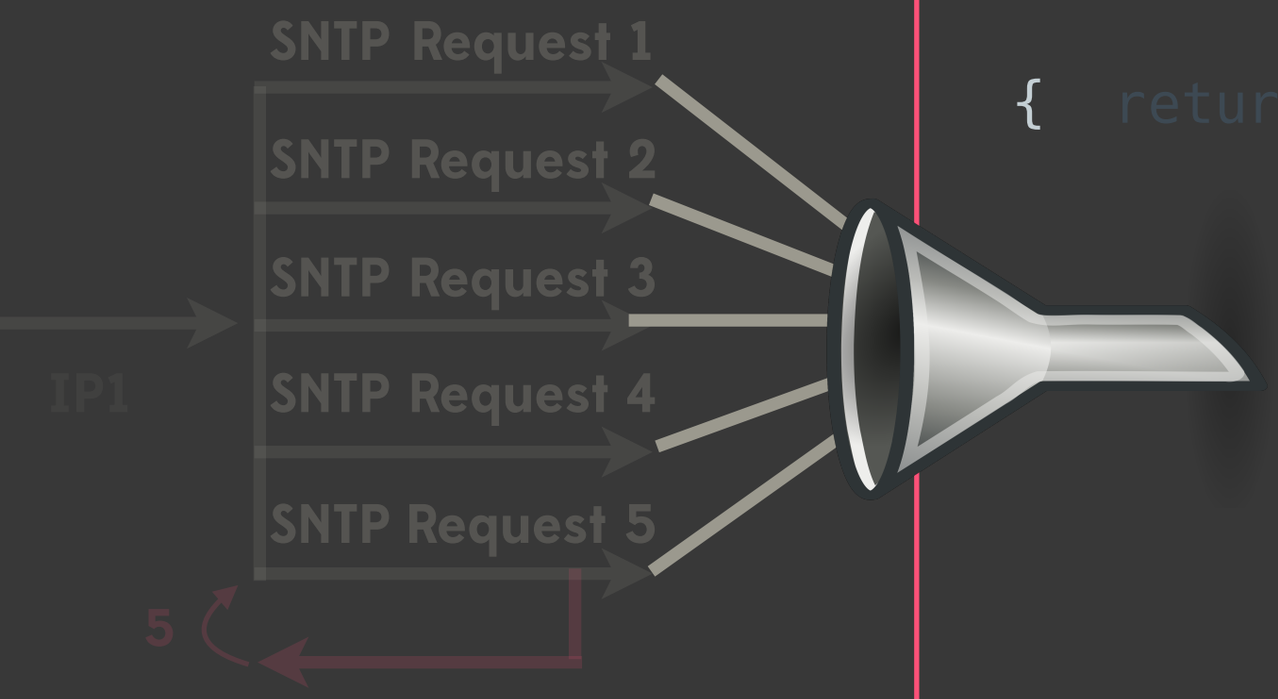
```
    .doOnError(throwable -> {
      // this request alone failed
       // retry this req alone
     })
    .retry(5))
```

```
}
```



Code:

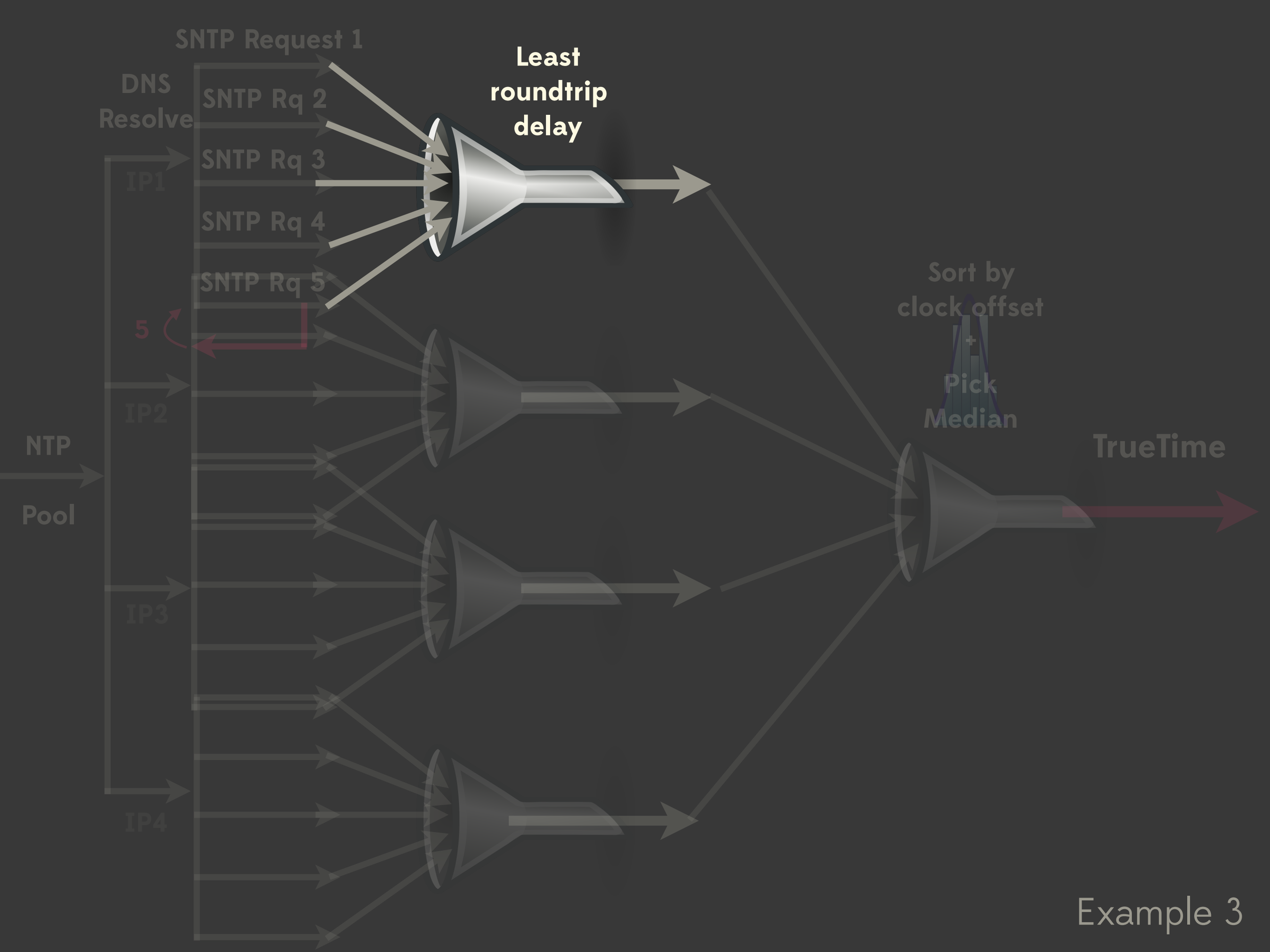
```
Observable
    .just(ntpPool)
    .compose(resolveNtpPool())
    .flatMap(bestResponseAgainstSingleIp())

                Func1 <String, Observable<Response>>
        bestResponseAgainstSingleIp()

    {   return Observable
                    .just(singleIp)
                    .repeat(5)

        .flatMap(ipAddress ->
                        sntpNtwrkReq(ipAddress))
                        .doOnError(throwable -> {
                          // this request alone failed
                            // retry this req alone
                         })
                        .retry(5))

                    )
            .toList()
    }
```



Code:

```
Observable
    .just(ntpPool)
    .compose(resolveNtpPool())
    .flatMap(bestResponseAgainstSingleIp())
```


```
Func1 <String, Observable<Response>>
        bestResponseAgainstSingleIp()

{
                .doOnError(throwable -> {
                    // this request alone failed
                    // retry this req alone
                })
                .retry(5))
            )
    .toList()
    .map(responseList -> {
            Collections.sort(responseList,
                             comparator);

            return responseList.get(0);
        })

}
```


SNTP Request 1
DNS Resolve
SNTP Rq 2
SNTP Rq 3
SNTP Rq 4
SNTP Rq 5
Least roundtrip delay
IP1
5
IP2
NTP
Pool
IP3
IP4
Sort by clock offset
+
Pick Median
TrueTime

Example 3

SNTP Request 1

DNS Resolve

SNTP Rq 2

SNTP Rq 3

IP1

SNTP Rq 4

SNTP Rq 5

5

IP2

NTP

Pool

IP3

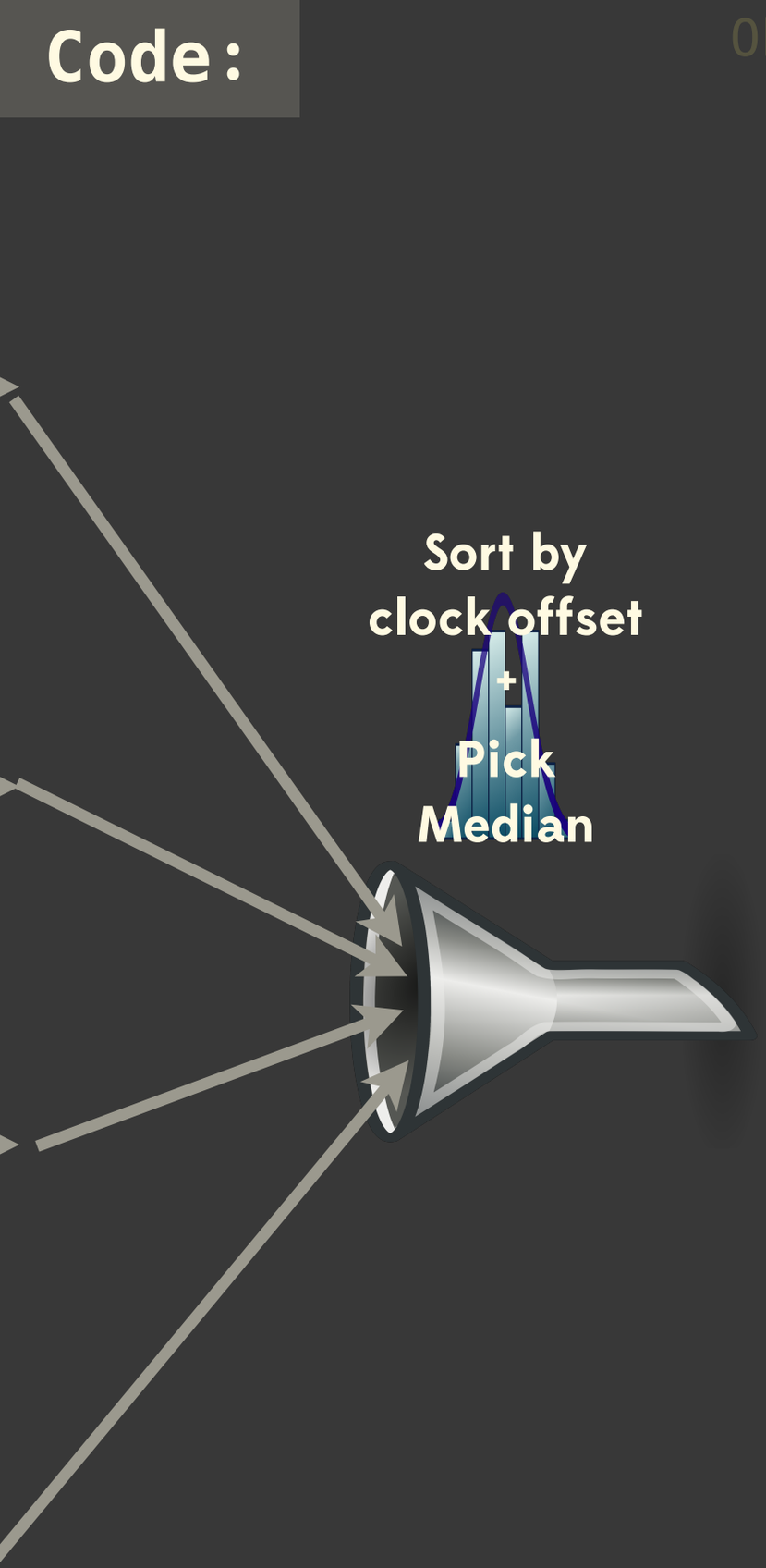
IP4

Sort by clock offset + Pick Median

TrueTime

Example 3

Code:




```
Observable
    .just(ntpPool)
    .compose(resolveNtpPool())
    .flatMap(bestResponseAgainstSingleIp())
    .toList()
    .map(filterMedianResponse())
```

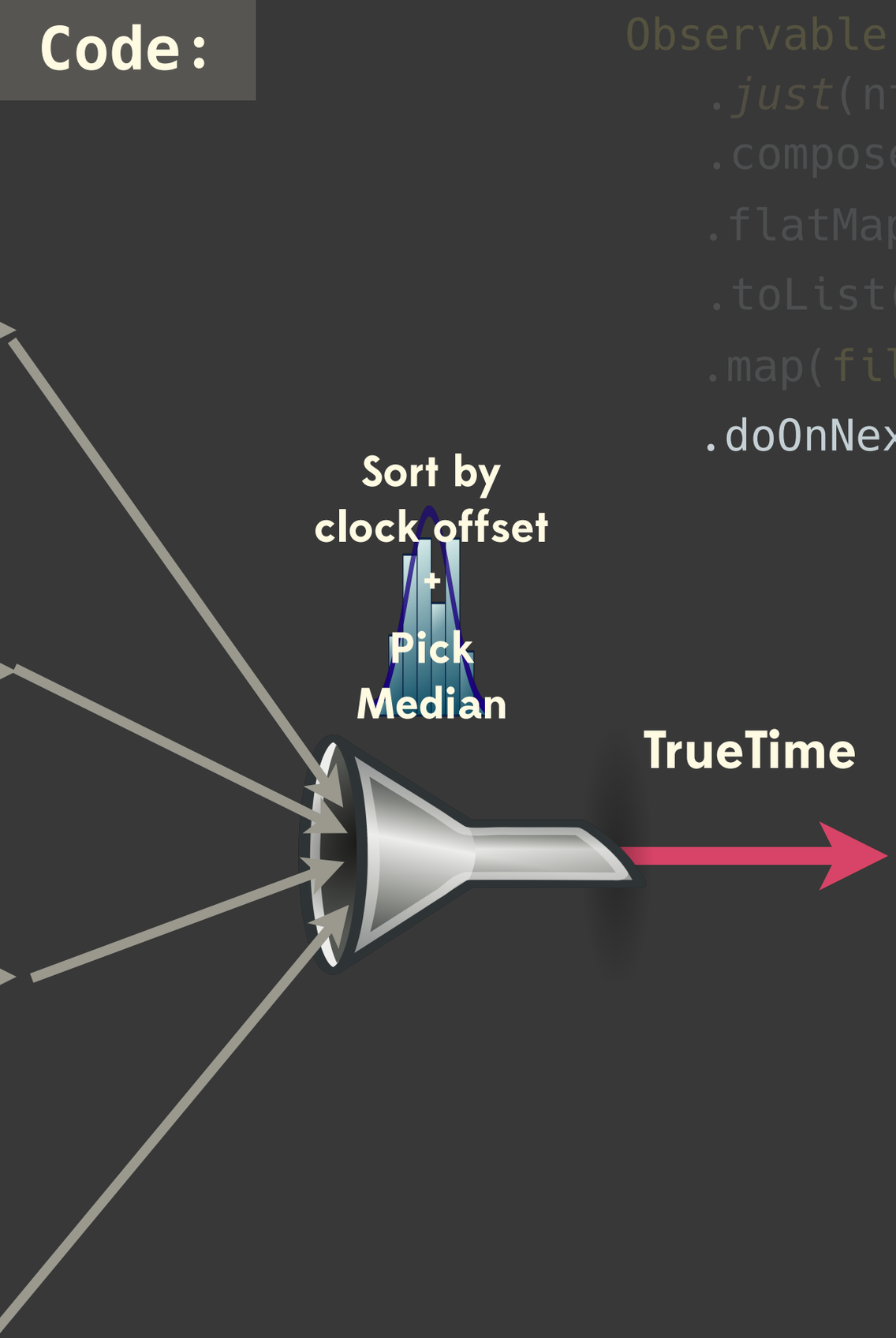
Func1 <List<Response>, Response>

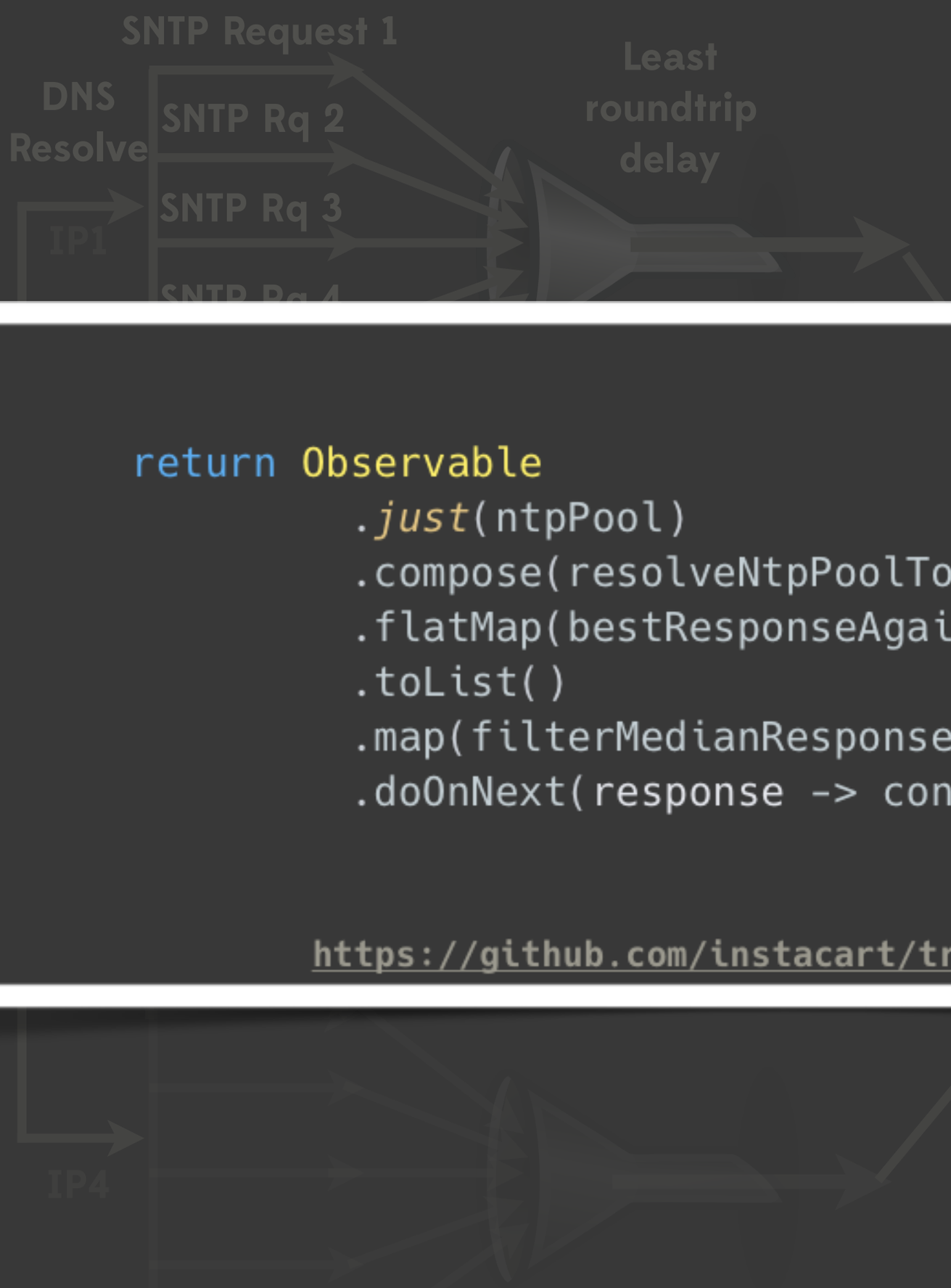
filterMedianResponse()

```
return responseList -> {

    Sort by clock offset:   Collections.sort(responses, comparator);

    Pick Median:            return bestResponses
                                    .get(bestResponses.size() / 2);
}
```

Code:

```
Observable
    .just(ntpPool)
    .compose(resolveNtpPool())
    .flatMap(bestResponseAgainstSingleIp())
    .toList()
    .map(filterMedianResponse())
    .doOnNext(response -> convertToTime(response));
```






```
return Observable
        .just(ntpPool)
        .compose(resolveNtpPoolToIpAddresses())
        .flatMap(bestResponseAgainstSingleIp(5))
        .toList()
        .map(filterMedianResponse())
        .doOnNext(response -> convertToTime(response));
```

https://github.com/instacart/truetime-android

@kaushikgopal

kaush.co

fragmentedpodcast.com

tech.instacart.com

My thanks to @cyrilmotier
who graciously allowed me to rip-off his slide deck theme