

Effective Network Programming



Ben Scheirman

- Director of Development, ChaiONE
- Houston, TX
- Guitar Player
- Beer Geek

benscheirman.com

[@subdigital](https://twitter.com/subdigital)

Bite-sized Screencasts

NS*Screencast*.com

Why is this important?

99% of our apps
are not self-contained

Why is this important?

Doing it wrong
RUINS user experience

Why is this important?

Everyone does it wrong

Why is this important?

(almost)

Everyone does it wrong

Exhibit A: Stuttery Scroller

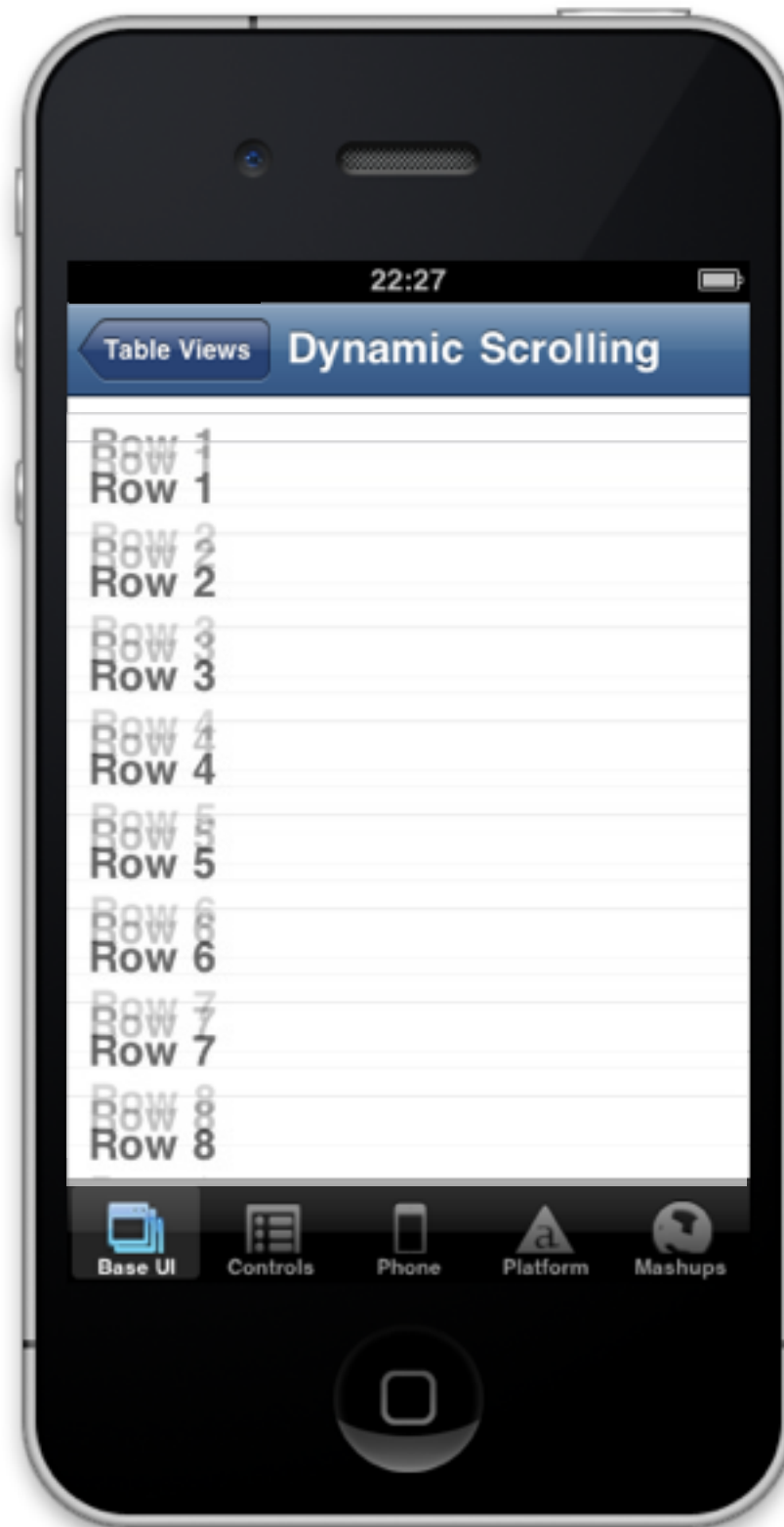
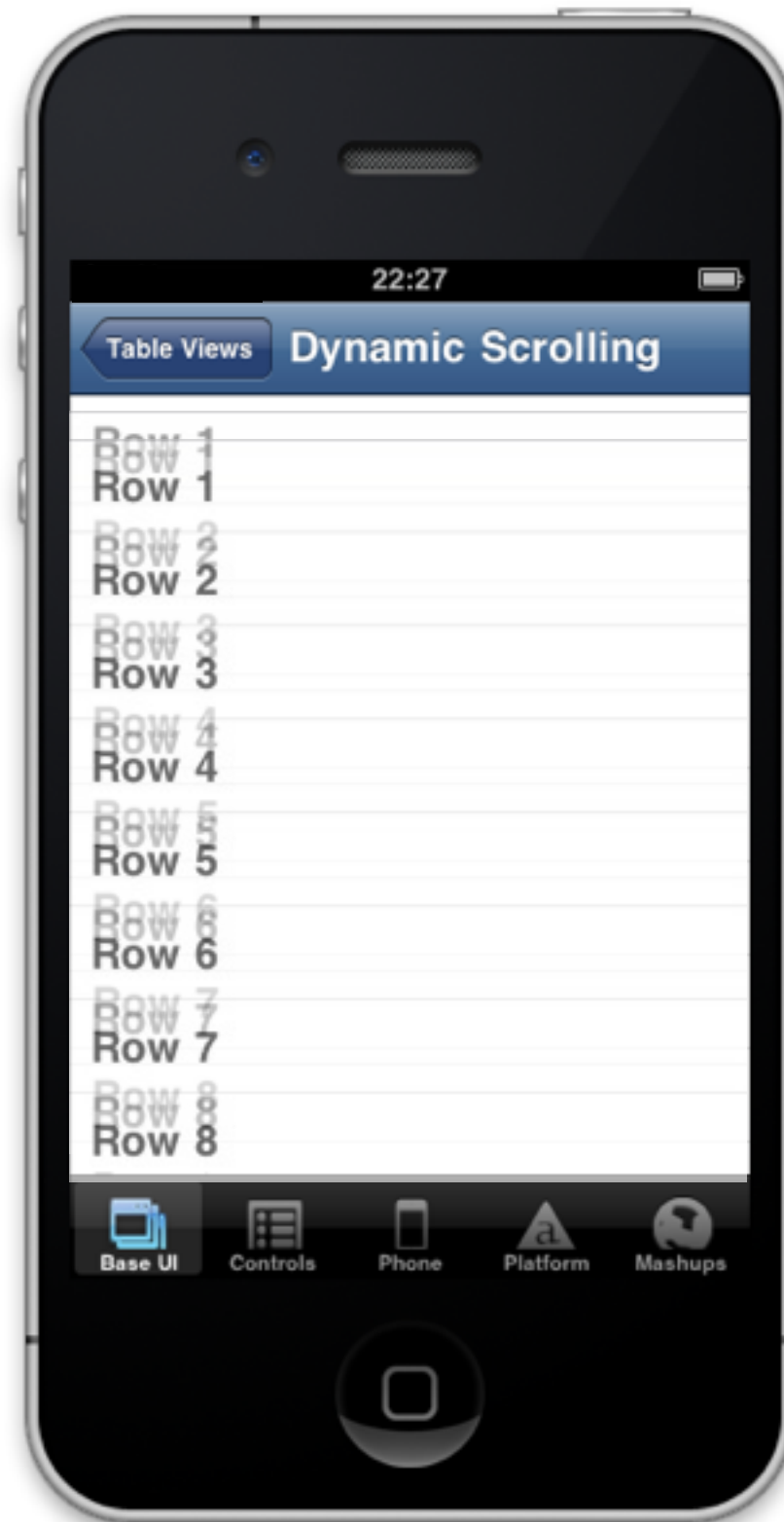


Exhibit A: Stuttery Scroller



Accessing the network on the main thread?

Exhibit B: The Perpetual Loader

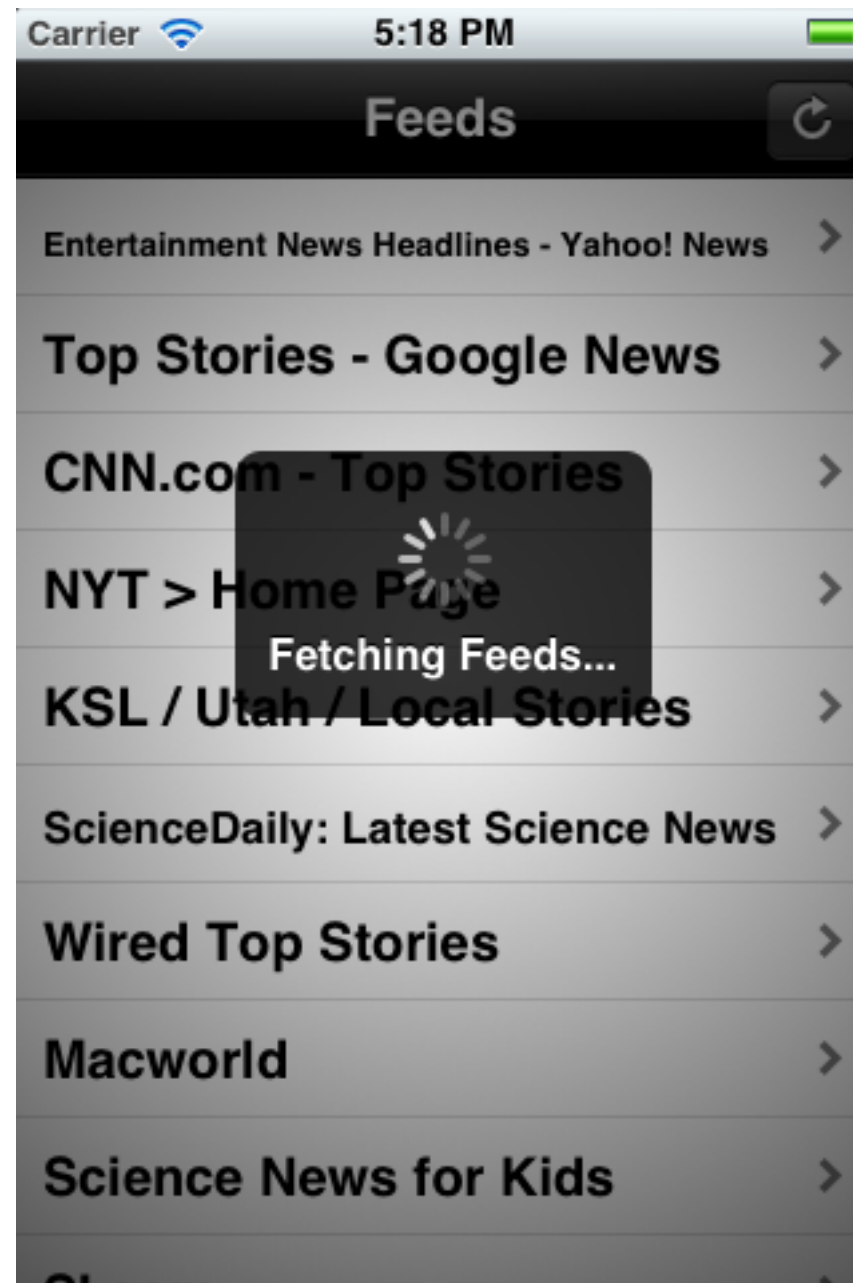
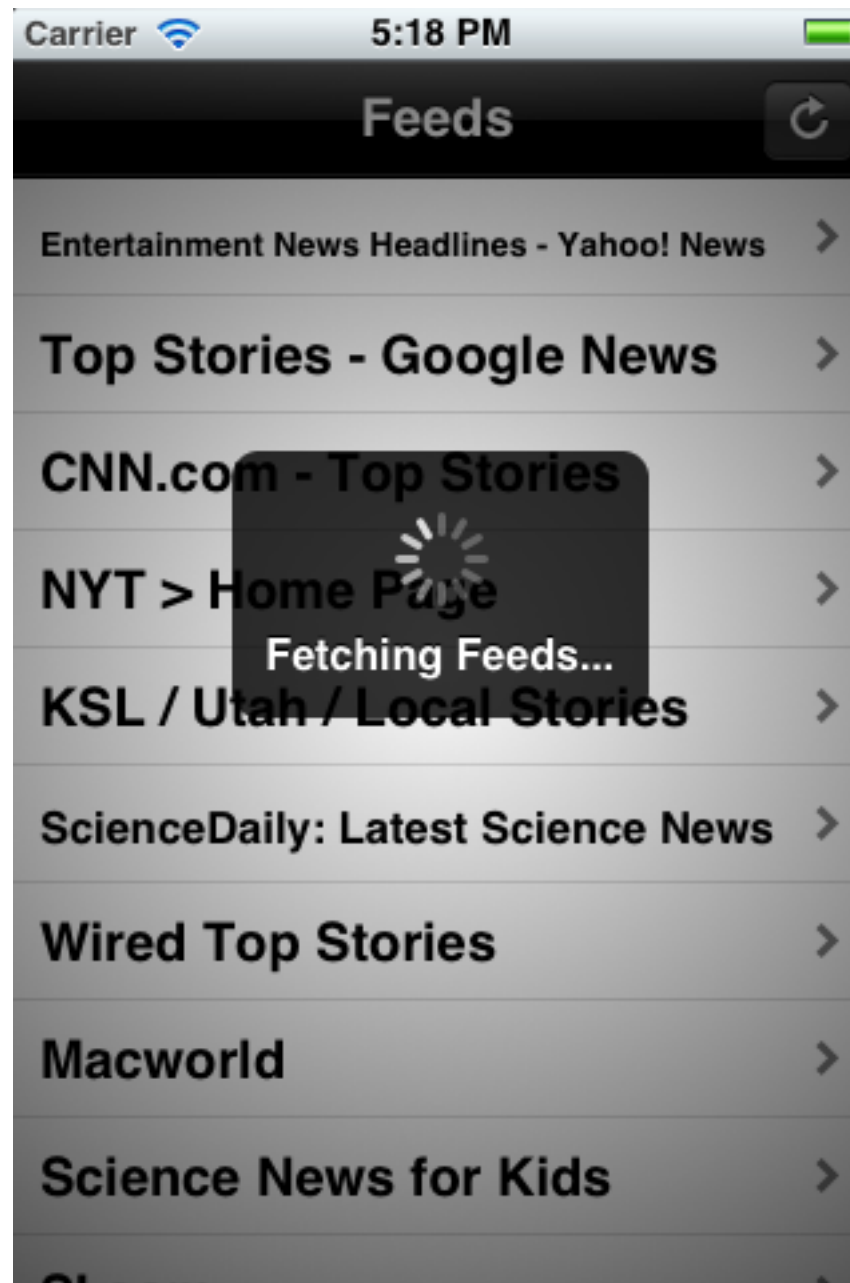


Exhibit B: The Perpetual Loader



How about
you cache
some data?

Exhibit C: Slow Loader

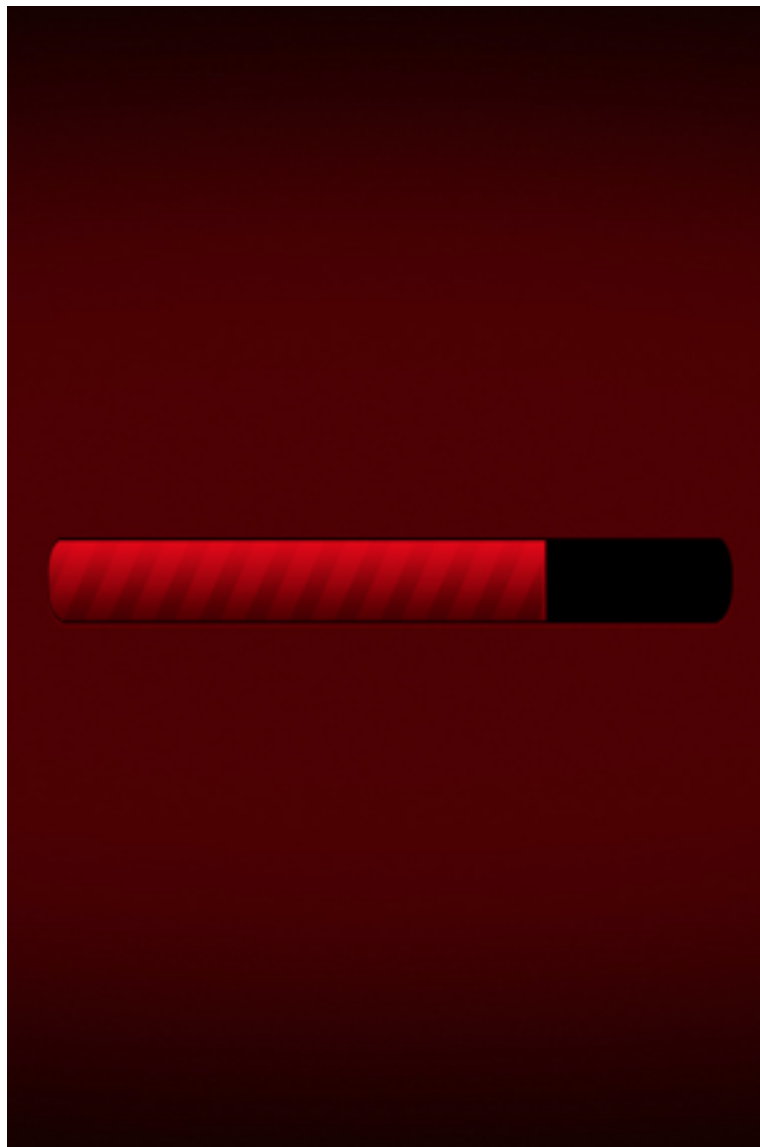
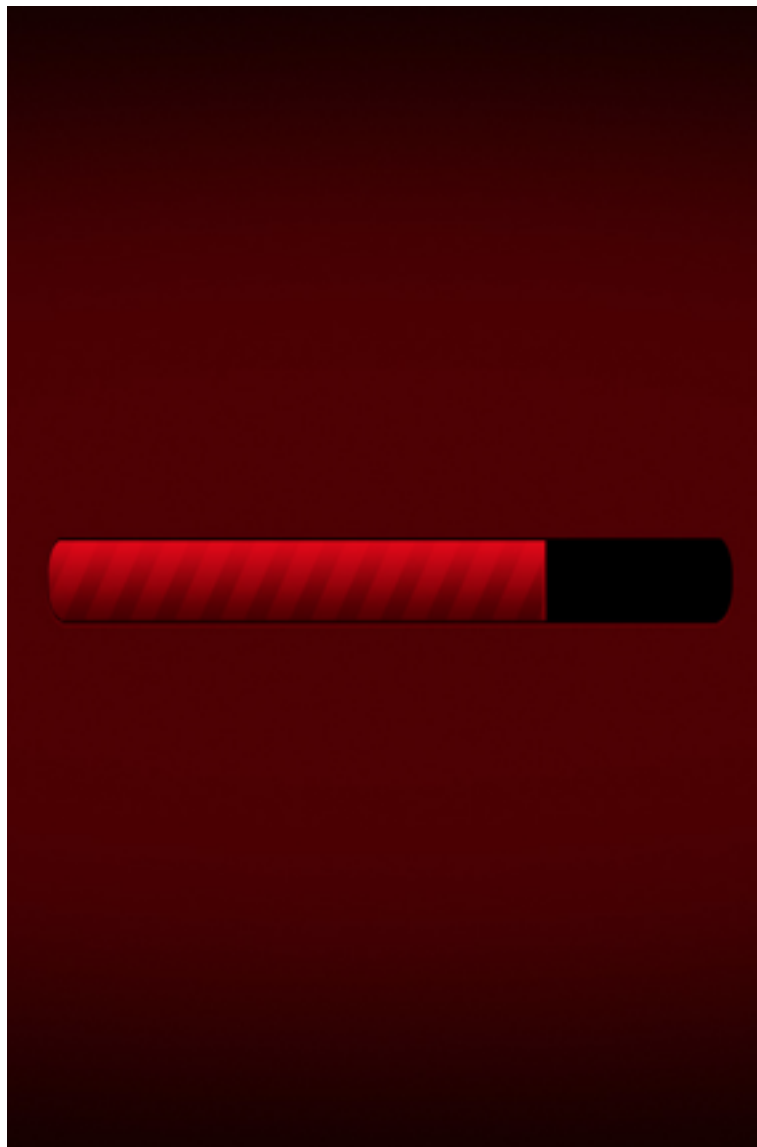


Exhibit C: Slow Loader



Loading too
much data?

Exhibit D: Yesterday's News

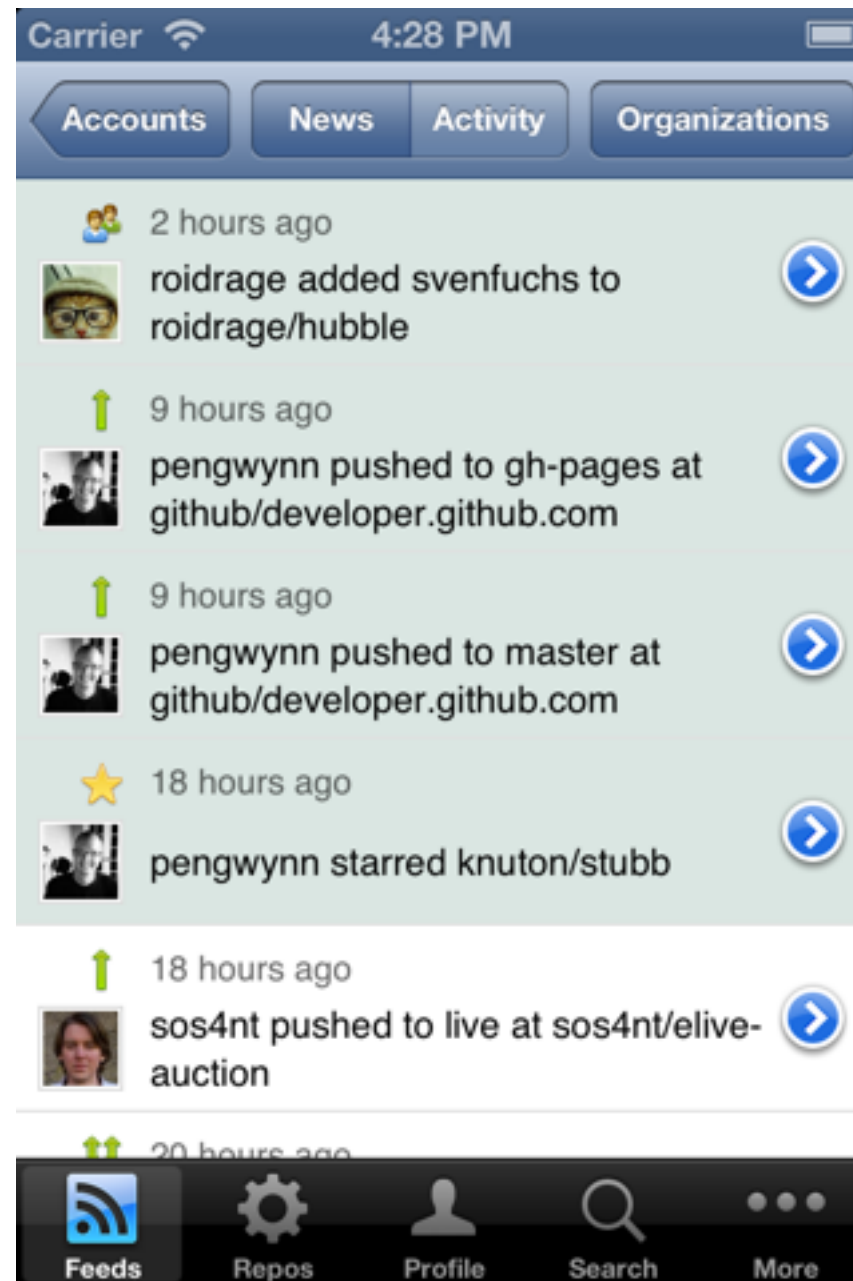
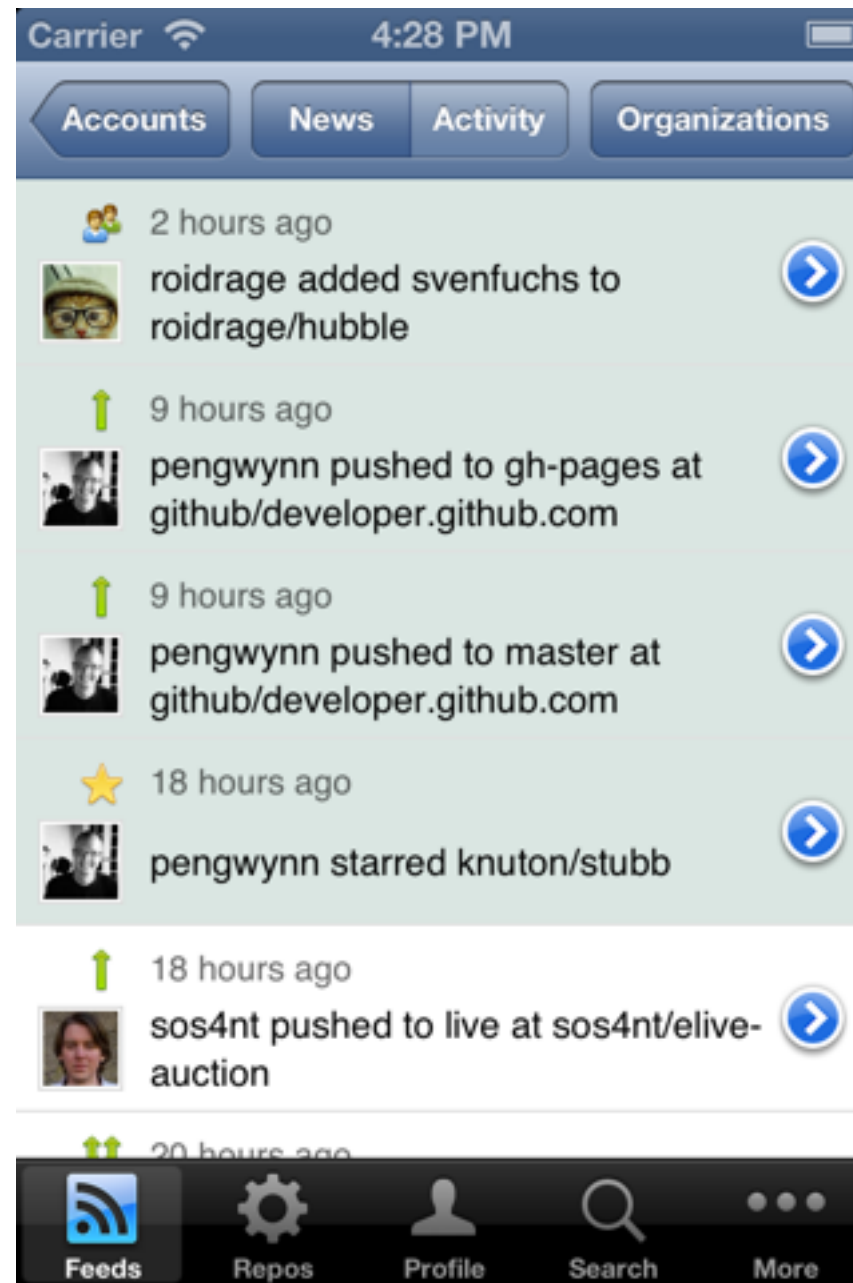


Exhibit D: Yesterday's News



Caching for
too long?

What we'll cover

- **AFNetworking Basics**
- **LIVE CODING (wish me luck!)**
- **HTTP Caching**
- **Caching Patterns**
- **API Design Tips**

AFNetworking





Installation

```
gem install cocoapods
```

Podfile

```
platform :ios, "5.0"  
pod 'AFNetworking'
```

```
pod install
```



Installation

```
gem install cocoapods
```

Podfile

```
platform :ios, "5.0"  
pod 'AFNetworking'
```

```
pod install
```



Installation

```
gem install cocoapods
```

Podfile

```
platform :ios, "5.0"  
pod 'AFNetworking'
```

```
pod install
```



Installation

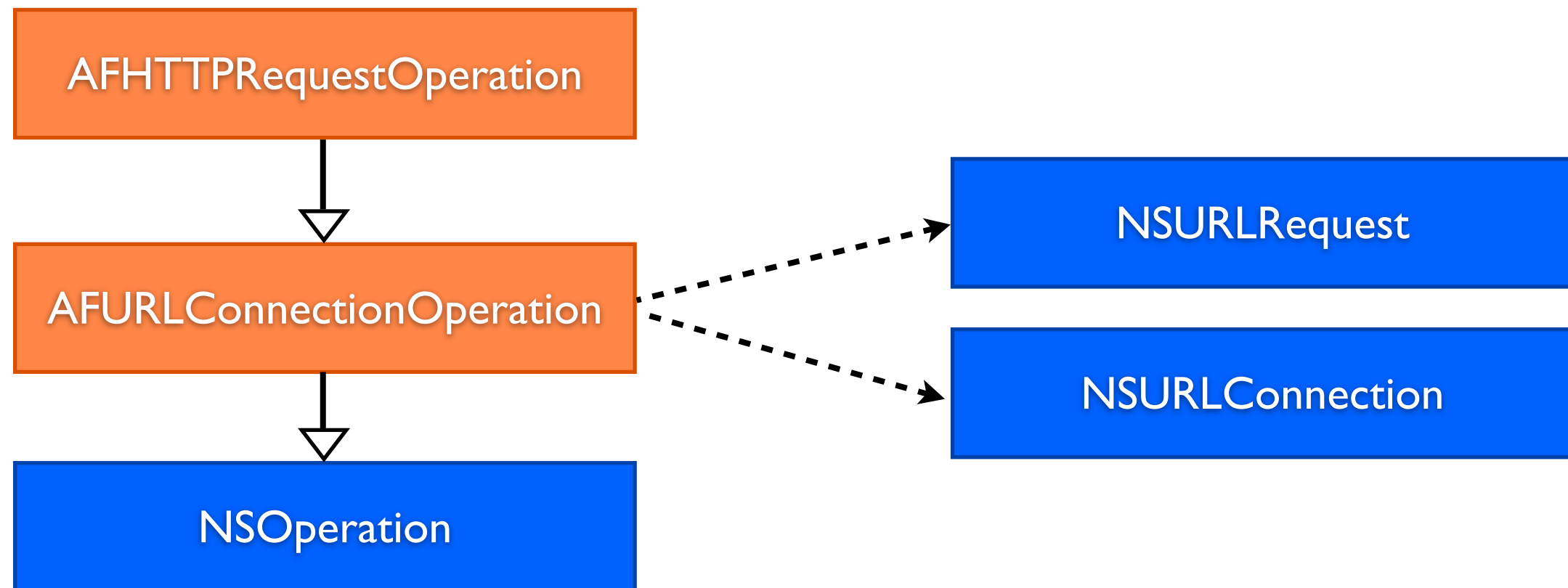
```
gem install cocoapods
```

Podfile

```
platform :ios, "5.0"  
pod 'AFNetworking'
```

```
pod install
```

AFNetworking Design



Why NSOperation?

- Queues (yours or internal)
- Priority
- Control concurrency
- Notify of completion, progress
- Cancelable

Usage

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
```

```
NSURLRequest *request = [NSURLRequest requestWithURL:url];
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
```

```
NSURLRequest *request = [NSURLRequest requestWithURL:url];
```

```
AFHTTPRequestOperation *operation;
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
```

```
NSURLRequest *request = [NSURLRequest requestWithURL:url];
```

```
AFHTTPRequestOperation *operation;  
operation = [[AFHTTPRequestOperation alloc]
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
```

```
NSURLRequest *request = [NSURLRequest requestWithURL:url];
```

```
AFHTTPRequestOperation *operation;  
operation = [[AFHTTPRequestOperation alloc]  
    initWithRequest:request];
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
```

```
NSURLRequest *request = [NSURLRequest requestWithURL:url];
```

```
AFHTTPRequestOperation *operation;
```

```
operation = [[AFHTTPRequestOperation alloc]
```

```
initWithRequest:request];
```

```
[operation setCompletionWithSuccess:^(AFHTTPRequestOperation *operation,
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
```

```
NSURLRequest *request = [NSURLRequest requestWithURL:url];
```

```
AFHTTPRequestOperation *operation;
```

```
operation = [[AFHTTPRequestOperation alloc]
```

```
initWithRequest:request];
```

```
[operation setCompletionWithSuccess:^(AFHTTPRequestOperation *operation,  
    id responseObject) {
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];
```

```
NSURLRequest *request = [NSURLRequest requestWithURL:url];
```

```
AFHTTPRequestOperation *operation;
```

```
operation = [[AFHTTPRequestOperation alloc]
```

```
initWithRequest:request];
```

```
[operation setCompletionWithSuccess:^(AFHTTPRequestOperation *operation,
```

```
id responseObject) {
```

```
// handle success
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLRequest *request = [NSURLRequest requestWithURL:url];

AFHTTPRequestOperation *operation;
operation = [[AFHTTPRequestOperation alloc]
initWithRequest:request];
[operation setCompletionWithSuccess:^(AFHTTPRequestOperation *operation,
    id responseObject) {
    // handle success
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLRequest *request = [NSURLRequest requestWithURL:url];

AFHTTPRequestOperation *operation;
operation = [[AFHTTPRequestOperation alloc]
initWithRequest:request];
[operation setCompletionWithSuccess:^(AFHTTPRequestOperation *operation,
    id responseObject) {
    // handle success
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    // handle failure
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLRequest *request = [NSURLRequest requestWithURL:url];

AFHTTPRequestOperation *operation;
operation = [[AFHTTPRequestOperation alloc]
initWithRequest:request];
[operation setCompletionWithSuccess:^(AFHTTPRequestOperation *operation,
    id responseObject) {
    // handle success
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    // handle failure
}];
```

Usage

```
NSURL *url = [NSURL URLWithString:@"http://google.com"];

NSURLRequest *request = [NSURLRequest requestWithURL:url];

AFHTTPRequestOperation *operation;
operation = [[AFHTTPRequestOperation alloc]
initWithRequest:request];
[operation setCompletionWithSuccess:^(AFHTTPRequestOperation *operation,
    id responseObject) {
    // handle success
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    // handle failure
}];

[operation start];
```

Requesting JSON

Requesting JSON

```
NSURL *url = [NSURL URLWithString:@"http://jsonip.com"];
```

Requesting JSON

```
NSURL *url = [NSURL URLWithString:@"http://jsonip.com"];  
NSURLRequest *request = [NSURLRequest requestWithURL:url];
```

Requesting JSON

```
NSURL *url = [NSURL URLWithString:@"http://jsonip.com"];  
NSURLRequest *request = [NSURLRequest requestWithURL:url];  
AFJSONRequestOperation *operation;
```

Requesting JSON

```
NSURL *url = [NSURL URLWithString:@"http://jsonip.com"];  
  
NSURLRequest *request = [NSURLRequest requestWithURL:url];  
  
AFJSONRequestOperation *operation;  
operation = [[AFJSONRequestOperation alloc]
```

Requesting JSON

```
NSURL *url = [NSURL URLWithString:@"http://jsonip.com"];  
  
NSURLRequest *request = [NSURLRequest requestWithURL:url];  
  
AFJSONRequestOperation *operation;  
operation = [[AFJSONRequestOperation alloc]  
    initWithRequest:request];
```

Requesting JSON

```
NSURL *url = [NSURL URLWithString:@"http://jsonip.com"];  
  
NSURLRequest *request = [NSURLRequest requestWithURL:url];  
  
AFJSONRequestOperation *operation;  
operation = [[AFJSONRequestOperation alloc]  
    initWithRequest:request];  
[operation setCompletionWithSuccess:^(
```

Requesting JSON

```
NSURL *url = [NSURL URLWithString:@"http://jsonip.com"];  
  
NSURLRequest *request = [NSURLRequest requestWithURL:url];  
  
AFJSONRequestOperation *operation;  
operation = [[AFJSONRequestOperation alloc]  
    initWithRequest:request];  
[operation setCompletionWithSuccess:^(  
    AFHTTPRequestOperation *operation, id responseObject) {
```

Requesting JSON

```
NSURL *url = [NSURL URLWithString:@"http://jsonip.com"];

NSURLRequest *request = [NSURLRequest requestWithURL:url];

AFJSONRequestOperation *operation;
operation = [[AFJSONRequestOperation alloc]
initWithRequest:request];
[operation setCompletionWithSuccess:^(
    AFHTTPRequestOperation *operation, id responseObject) {
    // responseObject is already parsed
```

Requesting JSON

```
NSURL *url = [NSURL URLWithString:@"http://jsonip.com"];

NSURLRequest *request = [NSURLRequest requestWithURL:url];

AFJSONRequestOperation *operation;
operation = [[AFJSONRequestOperation alloc]
initWithRequest:request];
[operation setCompletionWithSuccess:^(
    AFHTTPRequestOperation *operation, id responseObject) {
    // responseObject is already parsed
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
```

Requesting JSON

```
NSURL *url = [NSURL URLWithString:@"http://jsonip.com"];

NSURLRequest *request = [NSURLRequest requestWithURL:url];

AFJSONRequestOperation *operation;
operation = [[AFJSONRequestOperation alloc]
initWithRequest:request];
[operation setCompletionWithSuccess:^(
    AFHTTPRequestOperation *operation, id responseObject) {
    // responseObject is already parsed
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    // handle failure
```

Requesting JSON

```
NSURL *url = [NSURL URLWithString:@"http://jsonip.com"];

NSURLRequest *request = [NSURLRequest requestWithURL:url];

AFJSONRequestOperation *operation;
operation = [[AFJSONRequestOperation alloc]
initWithRequest:request];
[operation setCompletionWithSuccess:^(
    AFHTTPRequestOperation *operation, id responseObject) {
    // responseObject is already parsed
} failure:^(AFHTTPRequestOperation *operation, NSError *error) {
    // handle failure
}];
```

Requesting Images

Requesting Images

```
NSURL *imageUrl = [NSURL URLWithString:
```

Requesting Images

```
NSURL *imageUrl = [NSURL URLWithString:  
    @"http://i.imgur.com/RH1Si.jpg"];
```

Requesting Images

```
NSURL *imageUrl = [NSURL URLWithString:  
    @"http://i.imgur.com/RH1Si.jpg"];  
  
[imageView setImageWithURL:imageUrl];
```

Say it with me now...



Have you ever seen this?

```
NSURL *imageUrl = [NSURL URLWithString:  
    @"http://i.imgur.com/RH1Si.jpg"];  
  
NSData *imageData = [NSData  
    dataWithContentsOfURL:imageUrl];  
  
UIImage *image = [UIImage imageData:imageData];  
  
[imageView setImage:image];
```

Have you ever seen this?

```
NSURL *imageUrl = [NSURL URLWithString:  
    @"http://i.imgur.com/RH1Si.jpg"];  
  
NSData *imageData = [NSData  
    dataWithContentsOfURL:imageUrl];  
  
UIImage *image = [UIImage imageData:imageData];  
[imageView setImage:image];
```

WRONG

Have you ever seen this?

```
NSURL *imageUrl = [NSURL URLWithString:  
    @"http://i.imgur.com/RH1Si.jpg"];
```



```
NSData *imageData = [NSData  
    dataWithContentsOfURL:imageUrl];
```



```
UIImage *image = [UIImage imageData:imageData];  
[imageView setImage:image];
```

Requesting XML?

Requesting XML?

AFXMLRequestOperation

Requesting plist?

Requesting plist?

AFPropertyListRequestOperation

Requesting raw data?

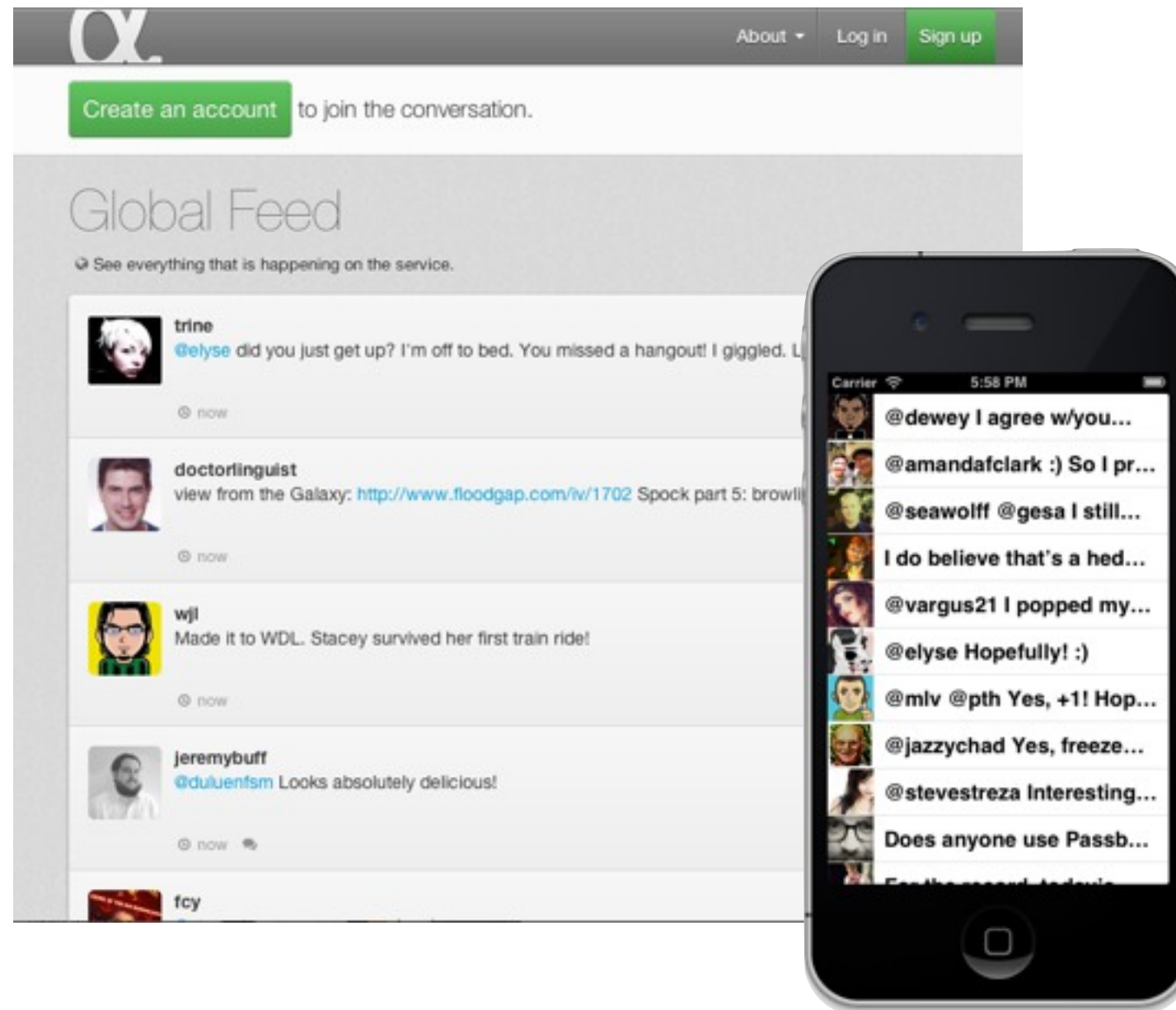
Requesting raw data?

AFHTTPRequestOperation

API Clients

-  Inherit from AFHTTPClient
-  Define shared instance
-  Register your operation class
-  Set up default headers

Demo: app.net client



What is success?

200-299

What about 404?

calls failure block

**AFNetworkingErrorDomain
(check response code)**

What about 302?

transparently handled for you

**call `setRedirectResponseBlock:`
to alter behavior**

Other errors

Timeout

No network available

401 Unauthorized

403 Forbidden

5xx server error

HTTP Caching

Client

Server

HTTP Caching

Client

Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com
```



HTTP Caching

Client

Server

GET /customer/152.json HTTP 1.1
Host: example.com



HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 37142
Etag: "e6482391249374127ca94128"
Cache-Control: max-age=60, public
Date: Sun, 21 Oct 2012 16:15:37

HTTP Caching

Client

Server

GET /customer/152.json HTTP 1.1
Host: example.com



HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 37142
Etag: "e6482391249374127ca94128"
Cache-Control: max-age=60, public
Date: Sun, 21 Oct 2012 16:15:37

HTTP Caching

Client

Server

GET /customer/152.json HTTP 1.1
Host: example.com



HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 37142
Etag: "e6482391249374127ca94128"
Cache-Control: max-age=60, public
Date: Sun, 21 Oct 2012 16:15:37

HTTP Caching

Client

Server

GET /customer/152.json HTTP 1.1
Host: example.com



HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 37142
Etag: "e6482391249374127ca94128"
Cache-Control: max-age=60, public
Date: Sun, 21 Oct 2012 16:15:37

Leveraging Etag on the client

Client

Server

Leveraging Etag on the client

Client

Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com  
If-None-Match: "e6482391249374127ca94128"
```



Leveraging Etag on the client

Client

Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com  
If-None-Match: "e6482391249374127ca94128"
```



HTTP/1.1 304 Not Modified

< no response body >



Leveraging Modified Date

Client

Server

Leveraging Modified Date

Client

Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com  
If-Modified-Since: <date>
```



Leveraging Modified Date

Client

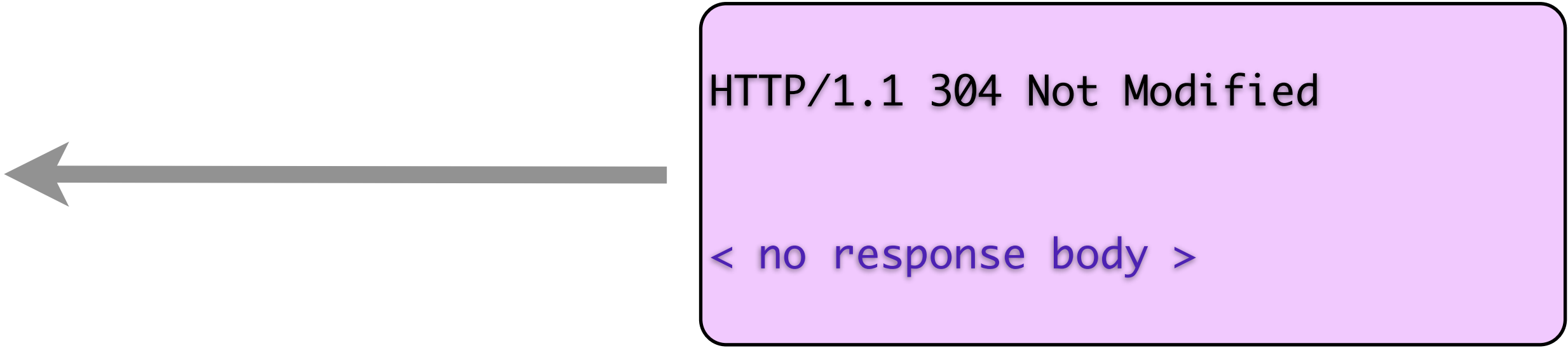
Server

```
GET /customer/152.json HTTP 1.1  
Host: example.com  
If-Modified-Since: <date>
```



```
HTTP/1.1 304 Not Modified
```

```
< no response body >
```



Etag/Last Modified on the Server

Etag/Last Modified on the Server

```
def show
```

Etag/Last Modified on the Server

```
def show
  @customer = Customer.find(params[:id])
```

Etag/Last Modified on the Server

```
def show
  @customer = Customer.find(params[:id])
  fresh_when(:etag => @customer,
```

Etag/Last Modified on the Server

```
def show
  @customer = Customer.find(params[:id])
  fresh_when(:etag => @customer,
             :last_modified => @customer,
```

Etag/Last Modified on the Server

```
def show
  @customer = Customer.find(params[:id])
  fresh_when(:etag => @customer,
             :last_modified => @customer,
             :public => true)
```

Etag/Last Modified on the Server

```
def show
  @customer = Customer.find(params[:id])
  fresh_when(:etag => @customer,
             :last_modified => @customer,
             :public => true)
  expires_in 10.minutes, :public => true
```

Etag/Last Modified on the Server

```
def show
  @customer = Customer.find(params[:id])
  fresh_when(:etag => @customer,
             :last_modified => @customer,
             :public => true)
  expires_in 10.minutes, :public => true
end
```

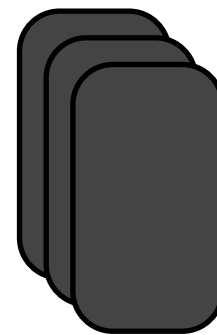
Etag/Last Modified on the Server

```
def show
  @customer = Customer.find(params[:id])
  fresh_when(:etag => @customer,
             :last_modified => @customer,
             :public => true)
  expires_in 10.minutes, :public => true
end
```

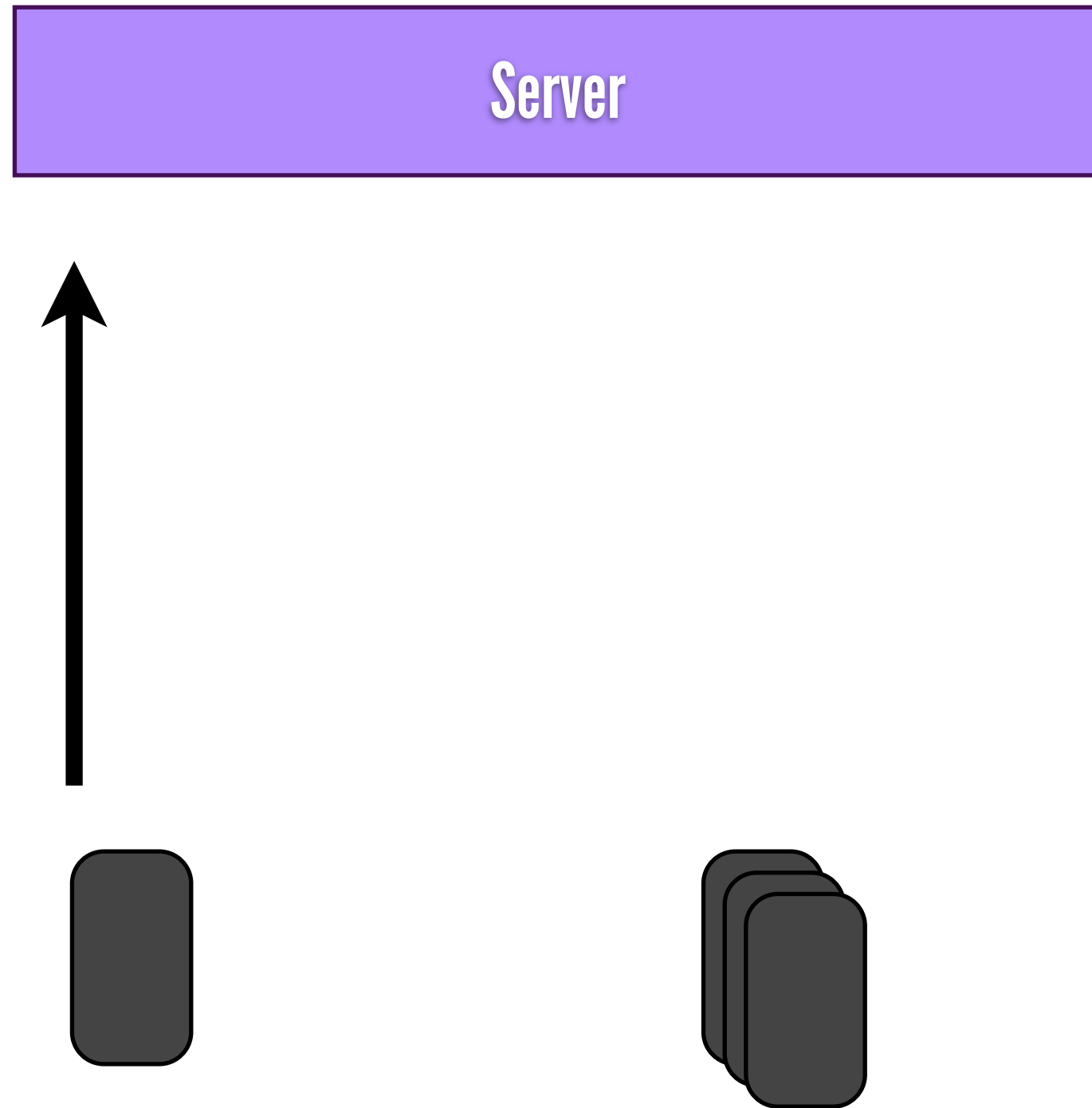
**Note: server still does a db query,
but doesn't render a template**

Saves on rendering & bandwidth for response transfer

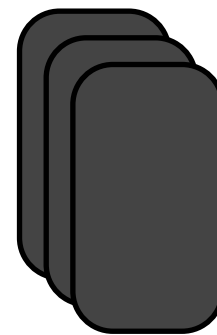
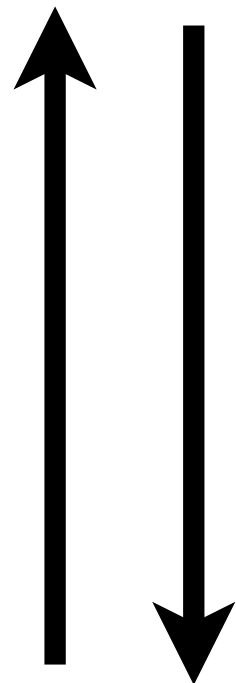
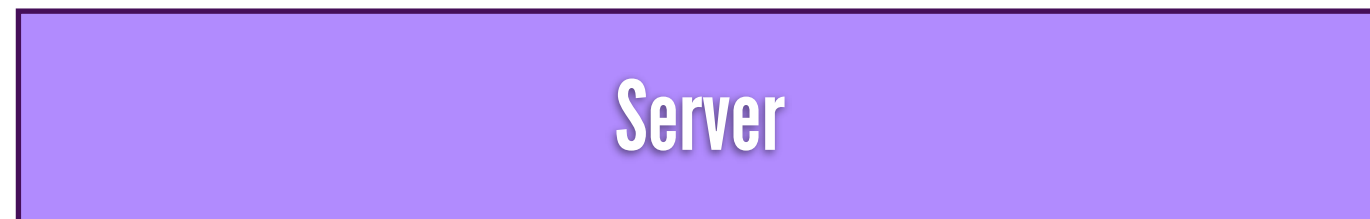
Cache-Control limitations



Cache-Control limitations

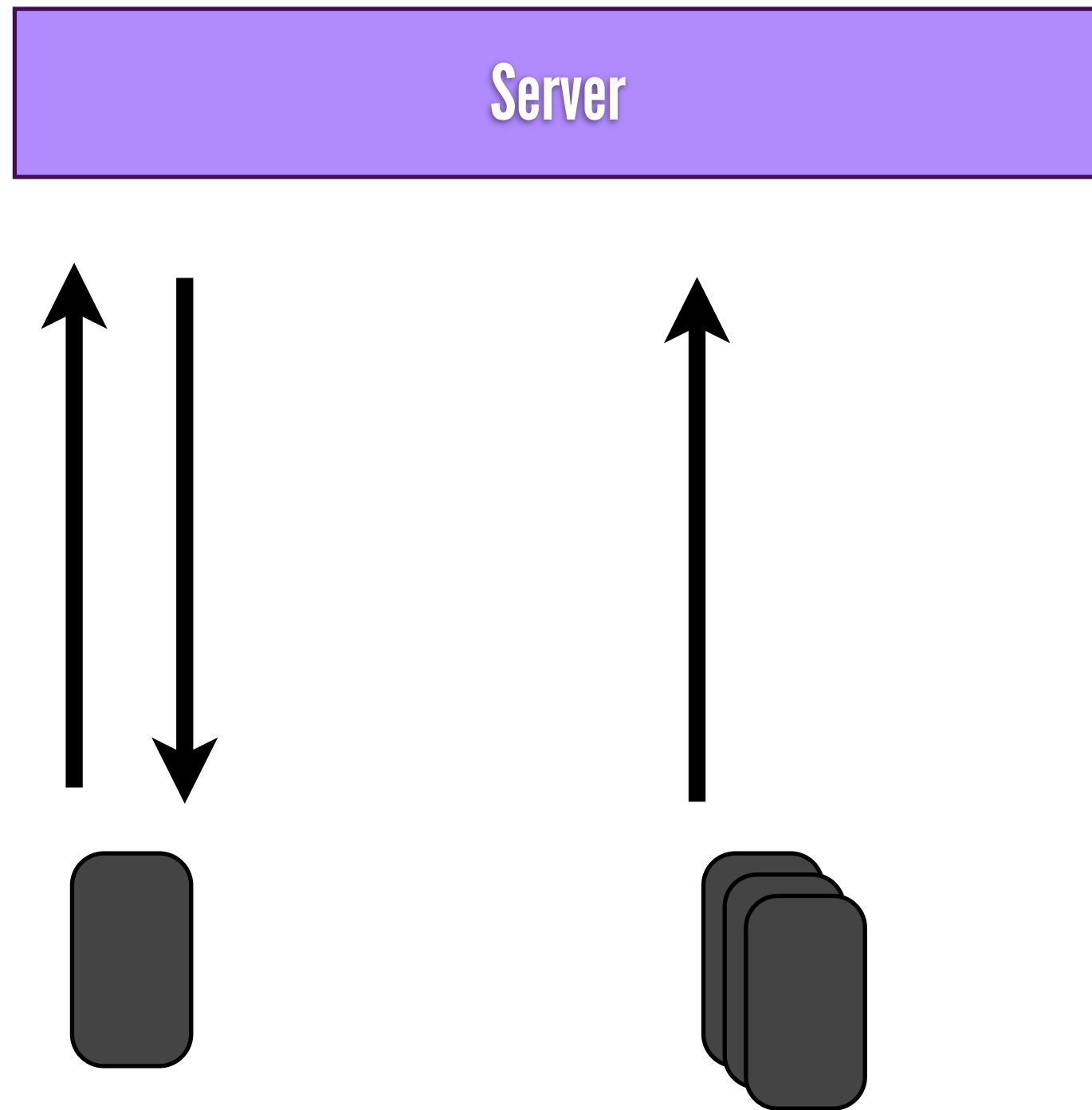


Cache-Control limitations



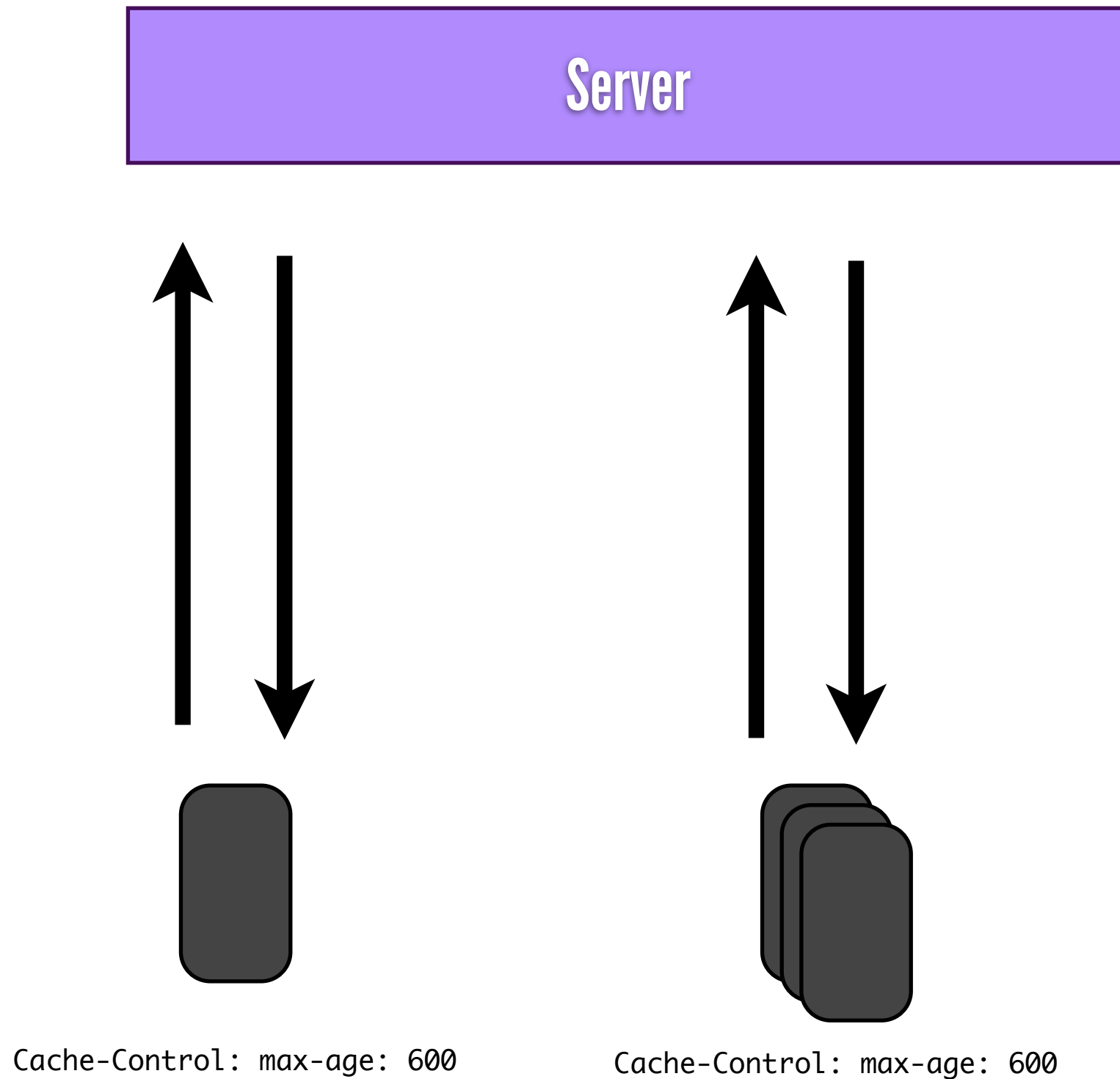
Cache-Control: max-age: 600

Cache-Control limitations



Cache-Control: max-age: 600

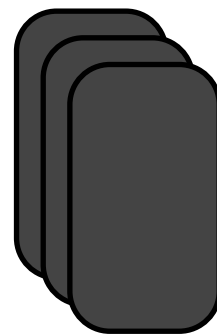
Cache-Control limitations



Reverse-Proxy Cache FTW

Server

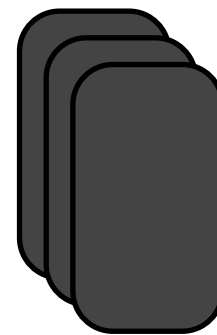
Varnish / Squid / Nginx



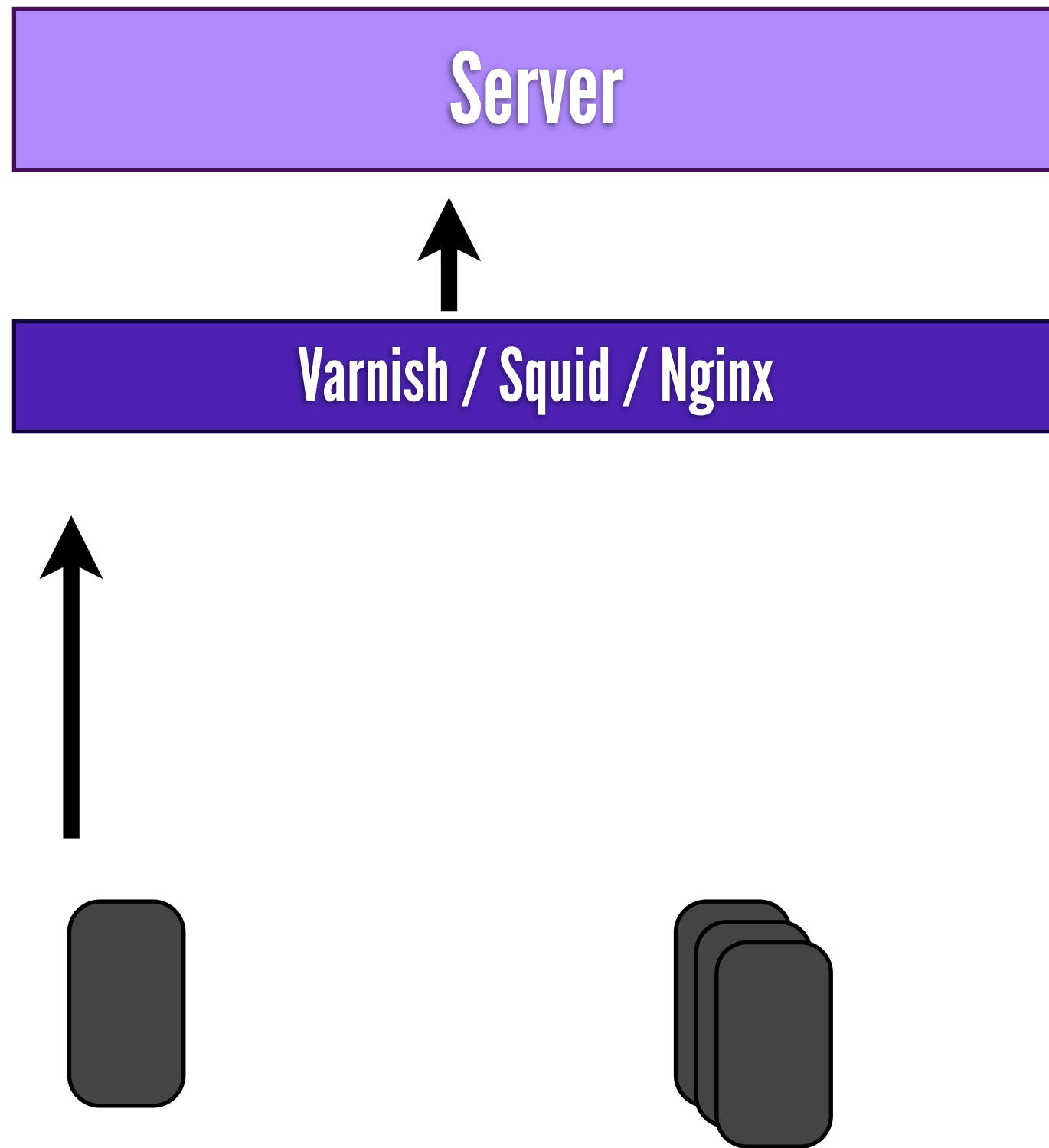
Reverse-Proxy Cache FTW

Server

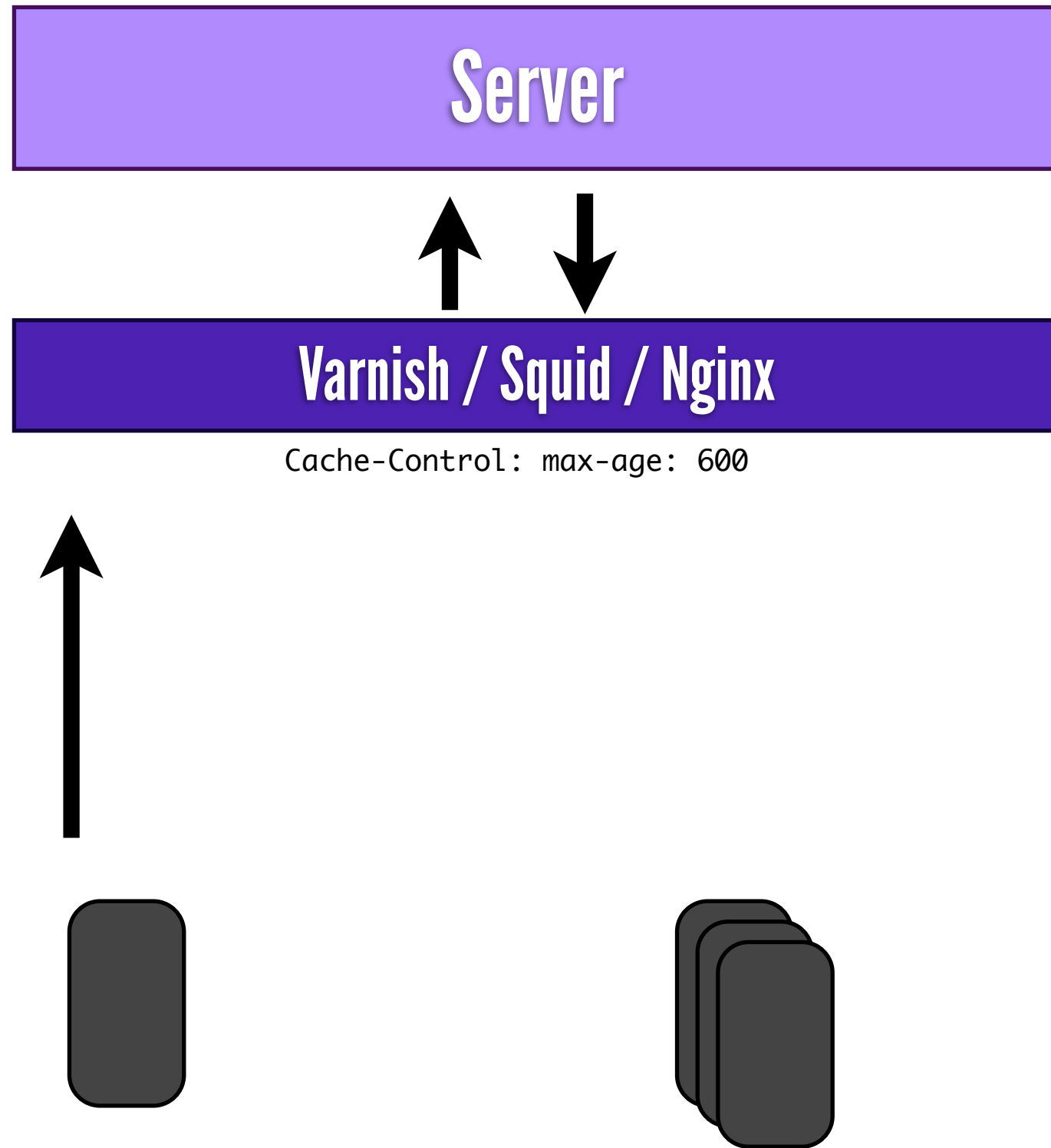
Varnish / Squid / Nginx



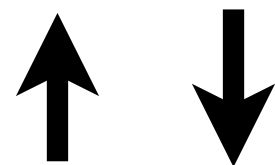
Reverse-Proxy Cache FTW



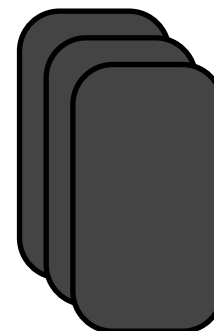
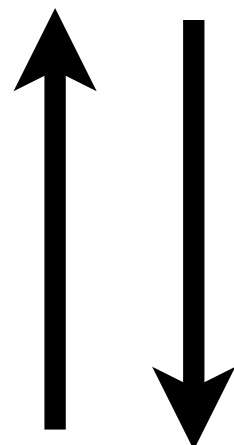
Reverse-Proxy Cache FTW



Reverse-Proxy Cache FTW

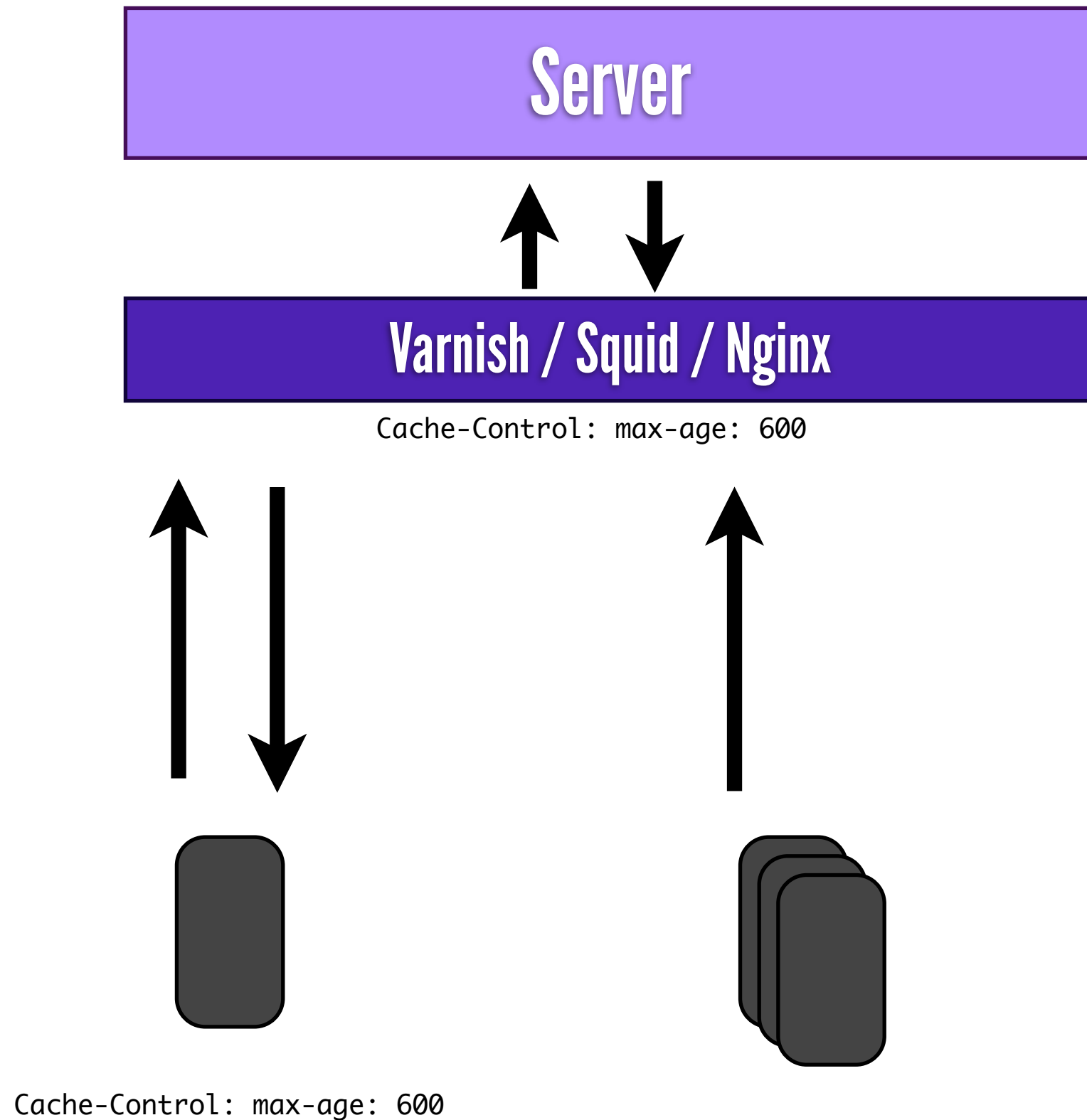


Cache-Control: max-age: 600

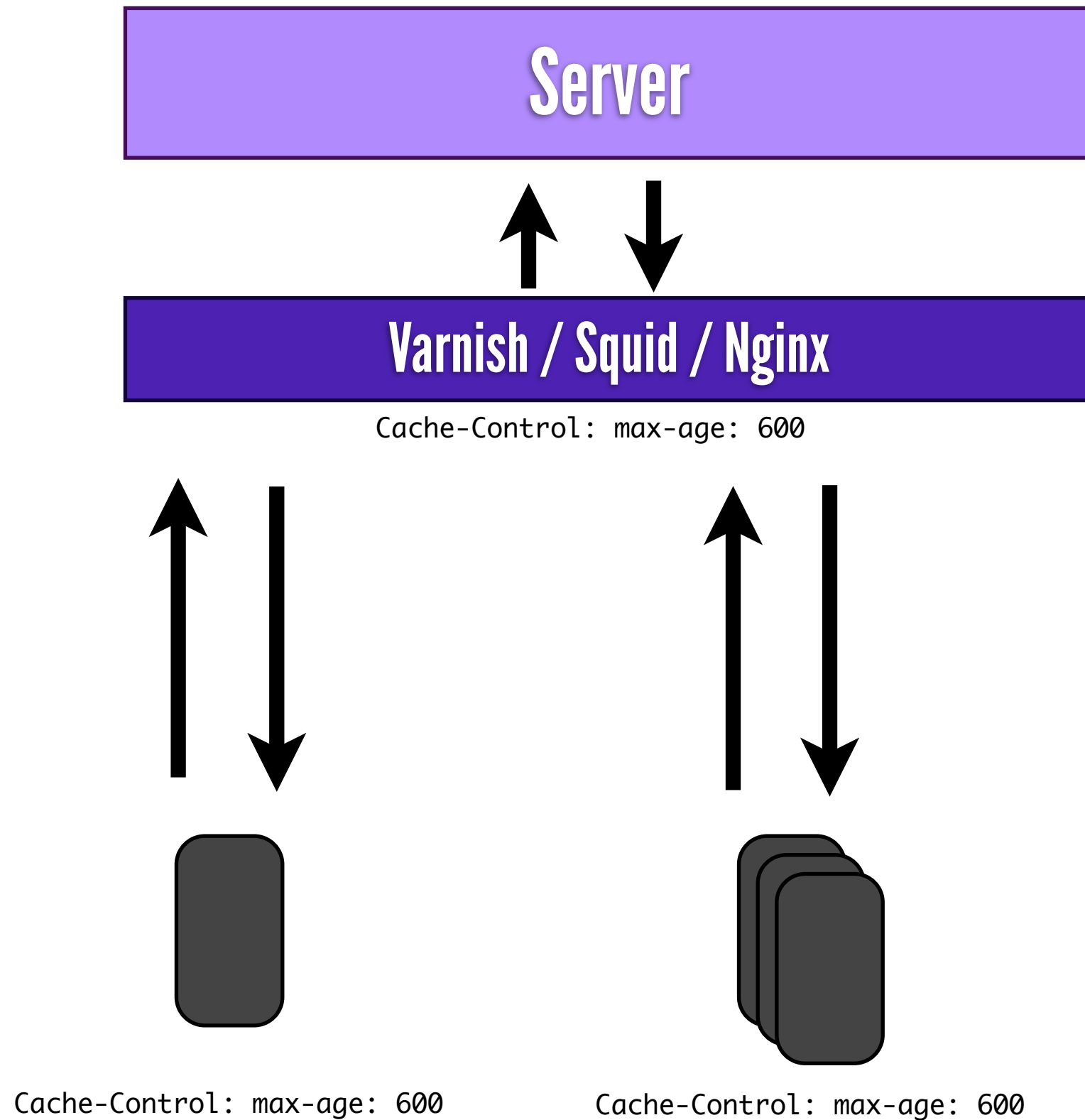


Cache-Control: max-age: 600

Reverse-Proxy Cache FTW



Reverse-Proxy Cache FTW



Leverage Cache-Control?



```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 37142
Etag: "e6482391249374127ca94128"
Cache-Control: max-age=9999999999, public
Date: Sun, 21 Oct 2012 16:15:37 GMT
```





Mike? Don't ever call me again.

Pros / Cons of Cache-Control

Client doesn't wait for response

Server spends no resources

But, clients may render stale data

Pros / Cons of Etag / IMS

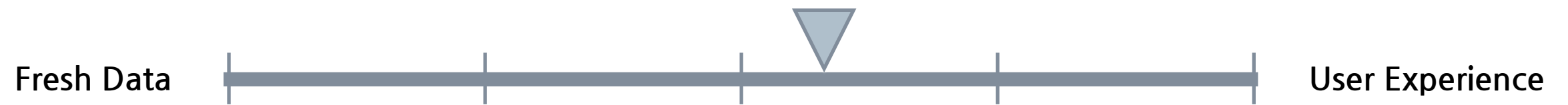
Server skips rendering
Miniscule data transfer
Appears instantaneous*

But, server still doing a db query

Reason about your data

- **List of US States**
- **User's Profile**
- **Customer's Order**
- **Activity Timeline**
- **Yesterday's stats**

Tradeoff



Tradeoff



Caching w/ AFNetworking

**Leverages NSURLCache
already**

Caching w/ AFNetworking

Customize if needed

```
NSURLCache *cache =  
    [[NSURLCache alloc] initWithMemoryCapacity:1024*1024*10  
                                     diskCapacity:1024*1024*50  
                                     diskPath:nil];  
[NSURLCache setSharedURLCache:cache];
```

Consider alternatives

SDURLCache

<http://github.com/steipete/SDURLCache>

Default Cache Location

~/Library/Caches/com.acme.app/Cache.db

Cache

main

Tables

cfurl_cache_blob_data

cfurl_cache_receiver_data

cfurl_cache_response

cfurl_cache_schema_version

sqlite_sequence

Views

Indexes

Triggers

Queries

- cfurl_cache_blob_data

cfurl_cache_receiver_data

cfurl_cache_response

cfurl_cache_schema_version

sqlite_sequence

cfurl_cache_response @main (Cache)

Image

Text

Hex

Import Wizard

Export Wizard

Sort Ascending

Sort Descending

Remove Sorting

	entry_ID	version	hash_value	storage_policy	request_key	time_stamp
	1	0	1424005679	0	https://alpha-api.	2012-10-19 17:2
	2	0	-1305208724	0	https://d2rfichhc	2012-10-19 17:3
	3	0	1572491858	0	https://d2rfichhc	2012-10-19 17:3
	4	0	-1997901260	0	https://d2rfichhc	2012-10-19 17:3
	5	0	-777154780	0	https://d2rfichhc	2012-10-19 17:3
	6	0	66153744	0	https://d2rfichhc	2012-10-19 17:3
	7	0	-404209947	0	https://d2rfichhc	2012-10-19 17:3
	8	0	-1931015091	0	https://d2rfichhc	2012-10-19 17:3
	9	0	338276037	0	https://d2rfichhc	2012-10-19 17:3
	10	0	1772610311	0	https://d2rfichhc	2012-10-19 17:3
	11	0	-769696739	0	https://d2rfichhc	2012-10-19 17:3
	12	0	-1381255225	0	https://d2rfichhc	2012-10-19 17:3
	13	0	2140559844	0	https://d2rfichhc	2012-10-19 17:3
	14	0	-358207116	0	https://d2rfichhc	2012-10-19 17:3
	15	0	-1622839865	0	https://d2rfichhc	2012-10-19 17:3
	16	0	1750647411	0	https://d2rfichhc	2012-10-19 17:3
	17	0	352022890	0	https://d2rfichhc	2012-10-19 17:3
	18	0	-1578321287	0	https://d2rfichhc	2012-10-19 17:3
	19	0	1302241355	0	https://d2rfichhc	2012-10-19 17:3
	20	0	1535007898	0	https://d2rfichhc	2012-10-19 17:3

select *, rowid "NAVICAT_ROWID" from "main"."cfurl_cache_response"

1

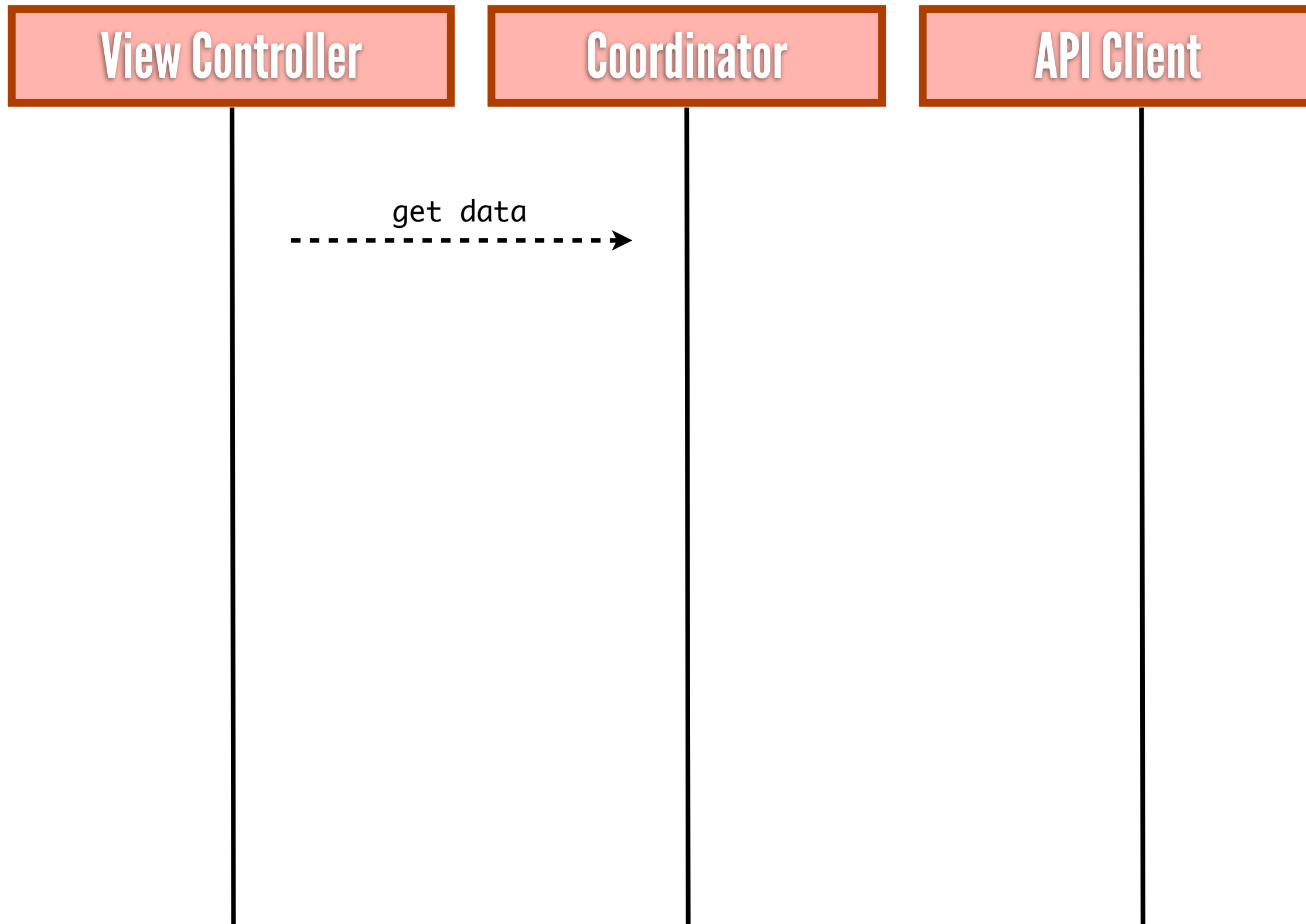
Switcheroo

View Controller

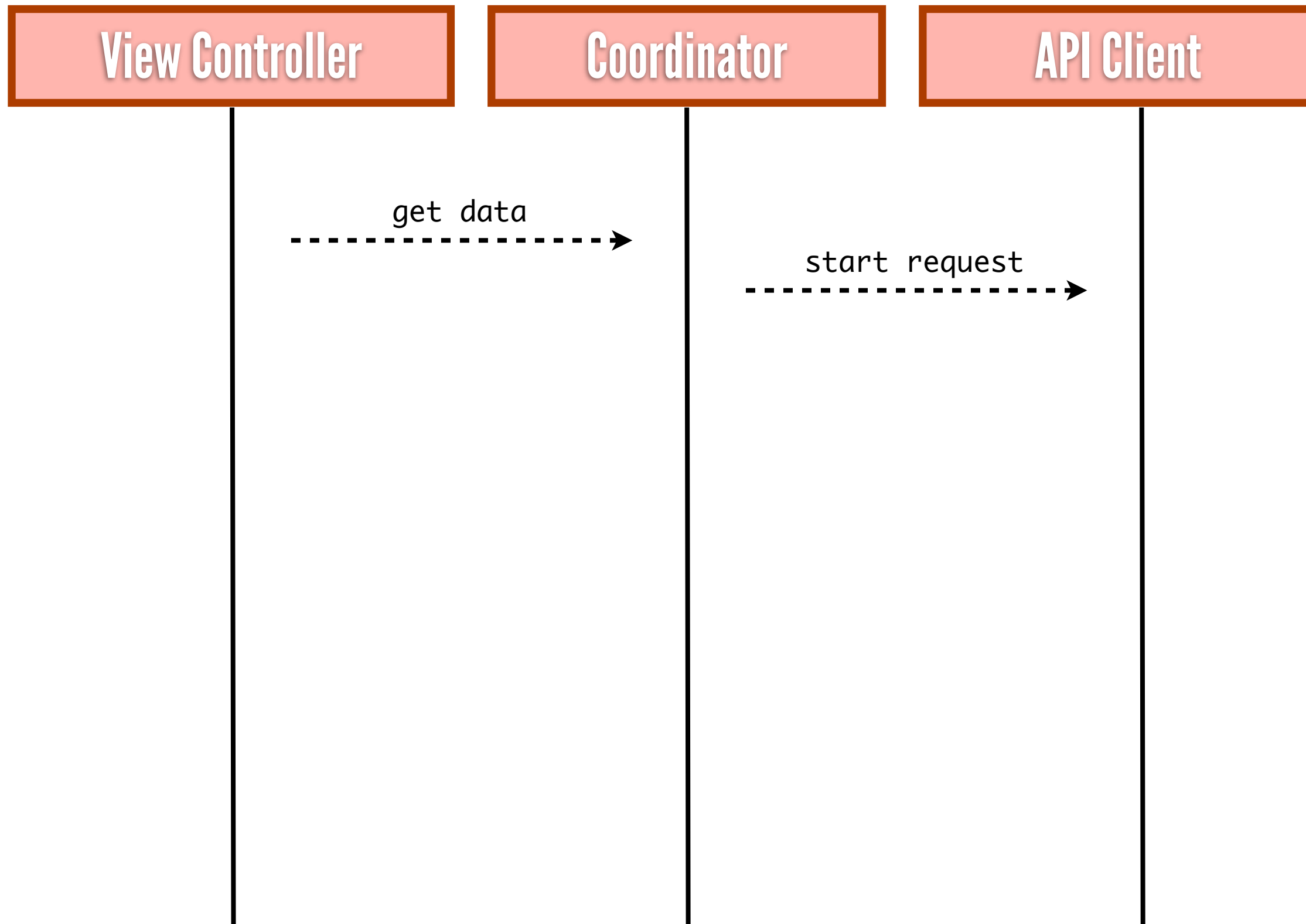
Coordinator

API Client

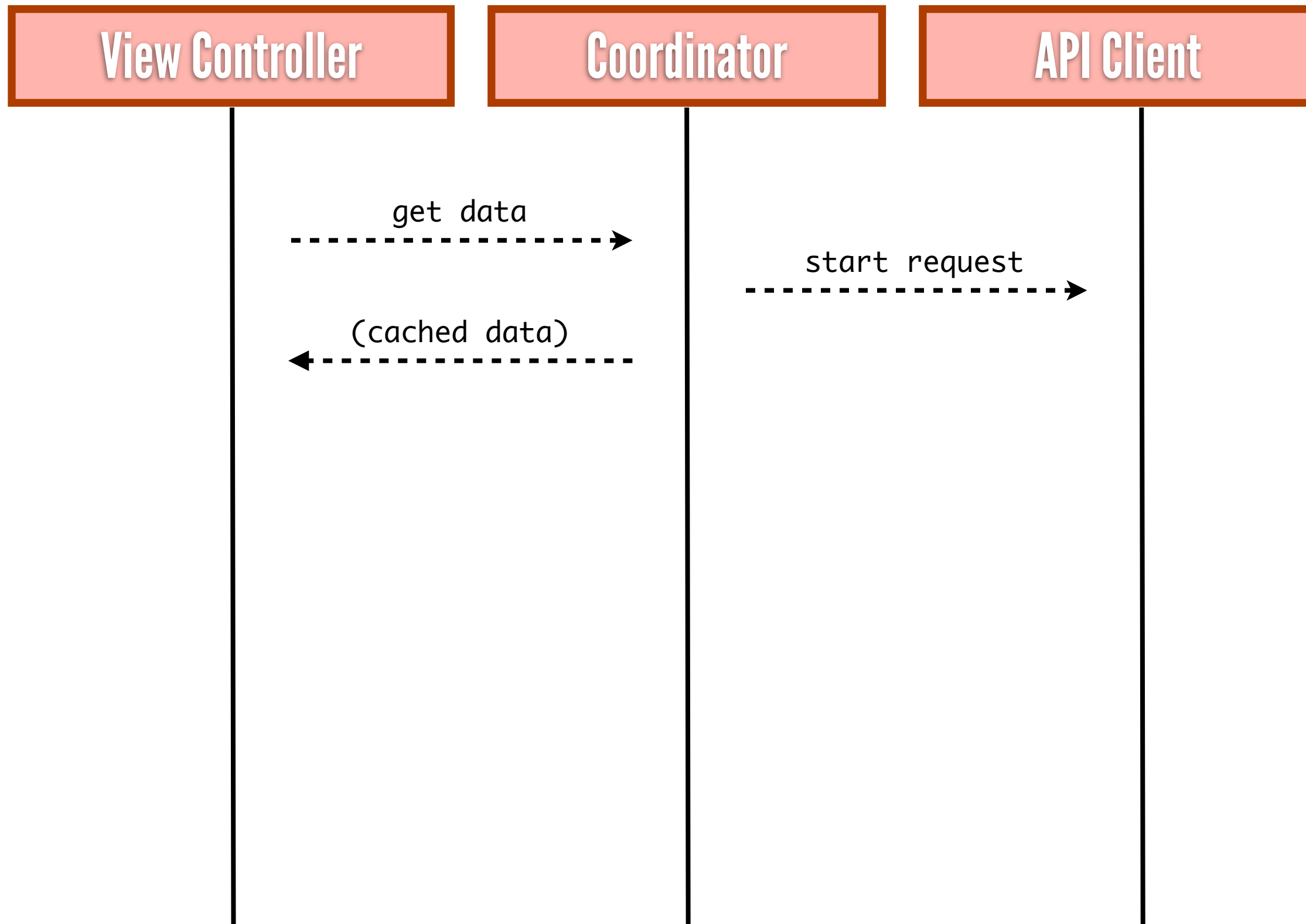
Switcheroo



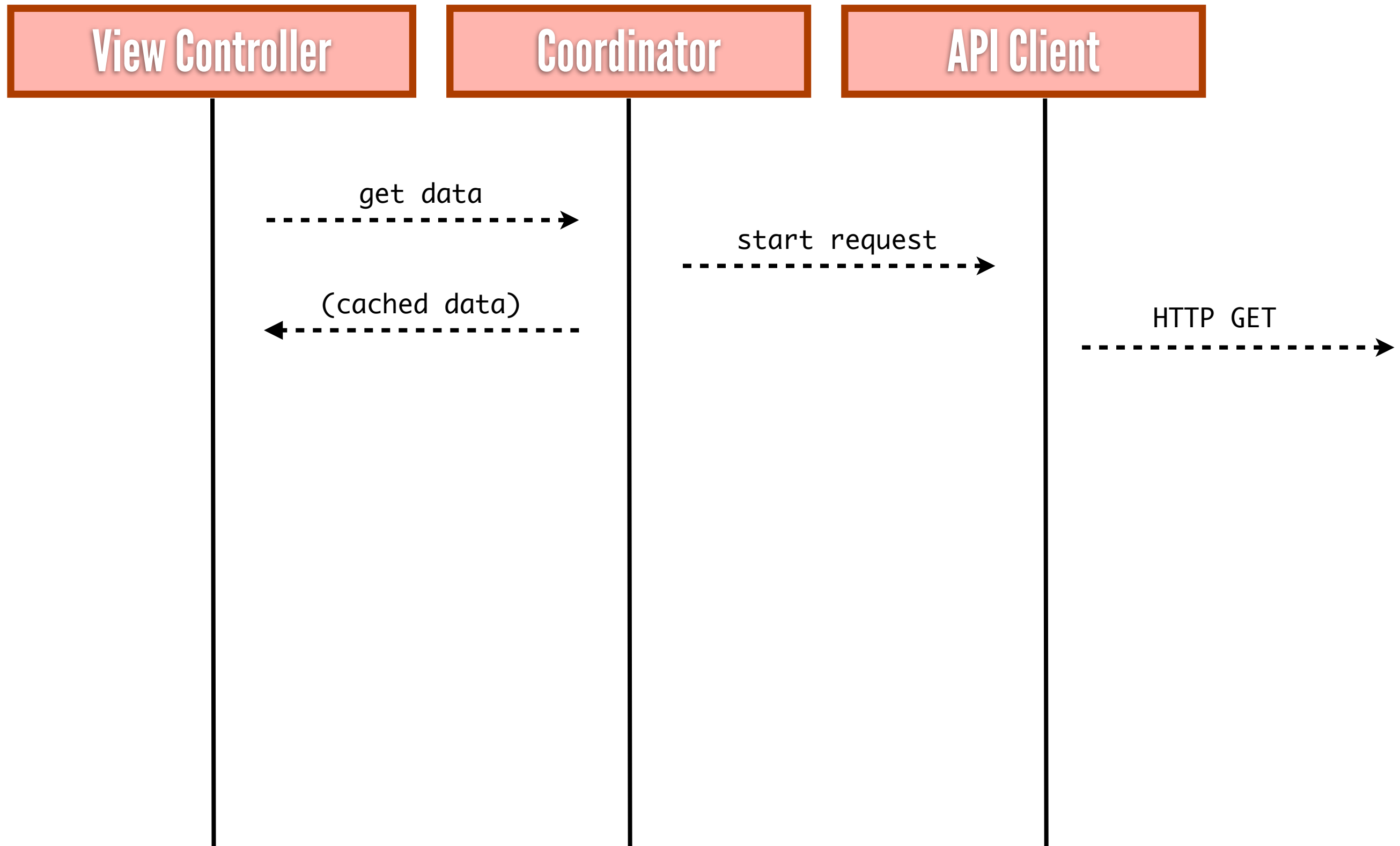
Switcheroo



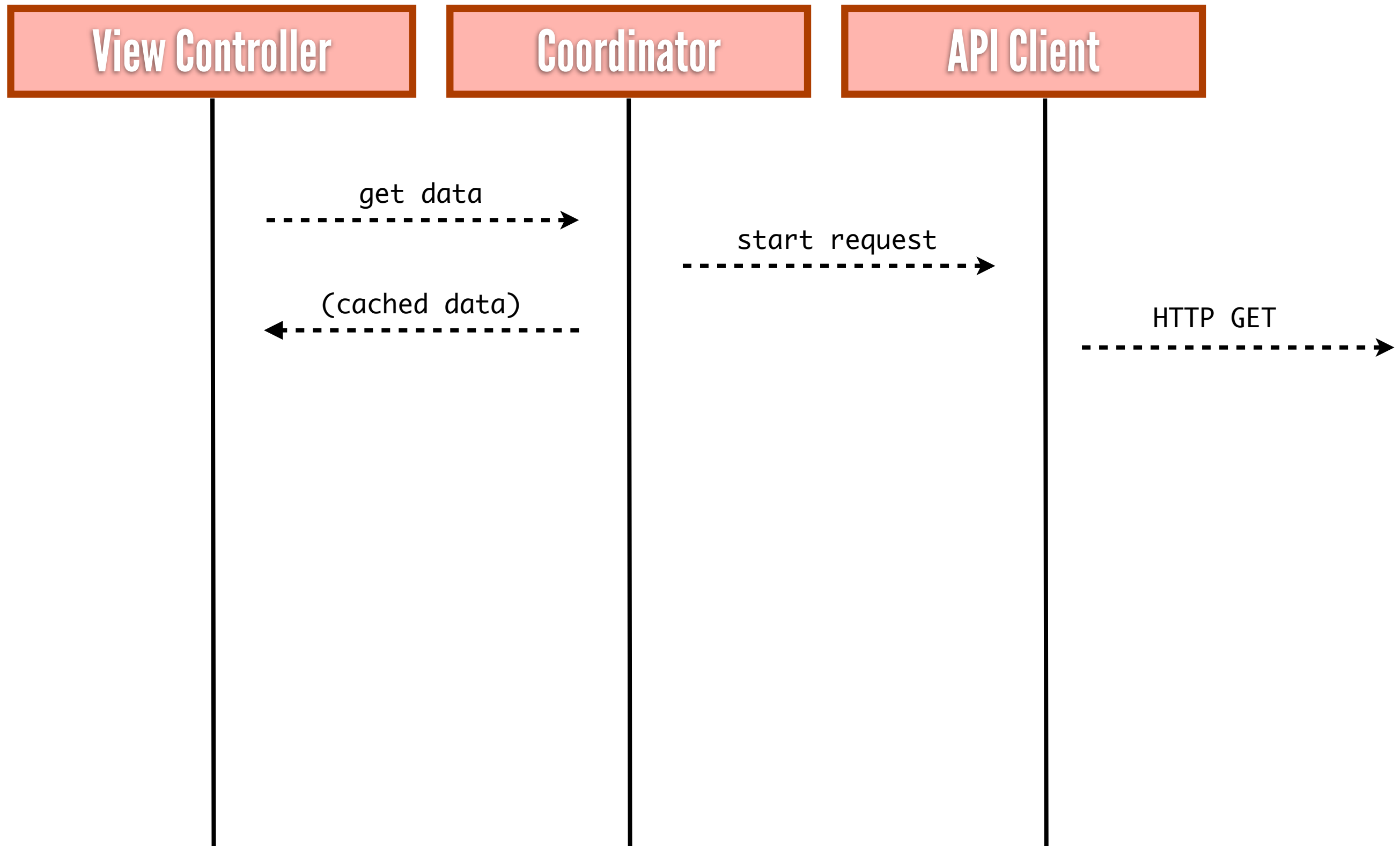
Switcheroo



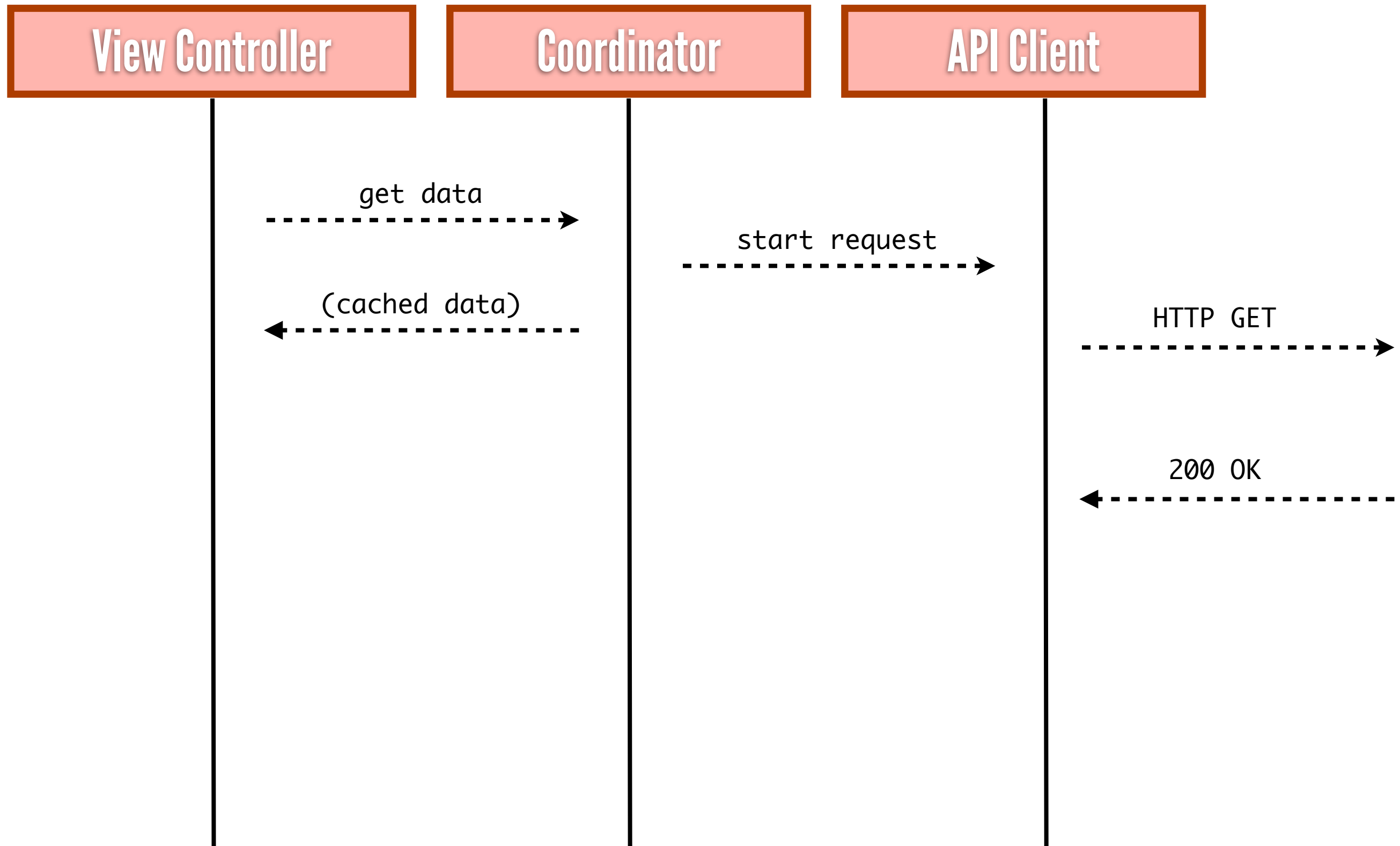
Switcheroo



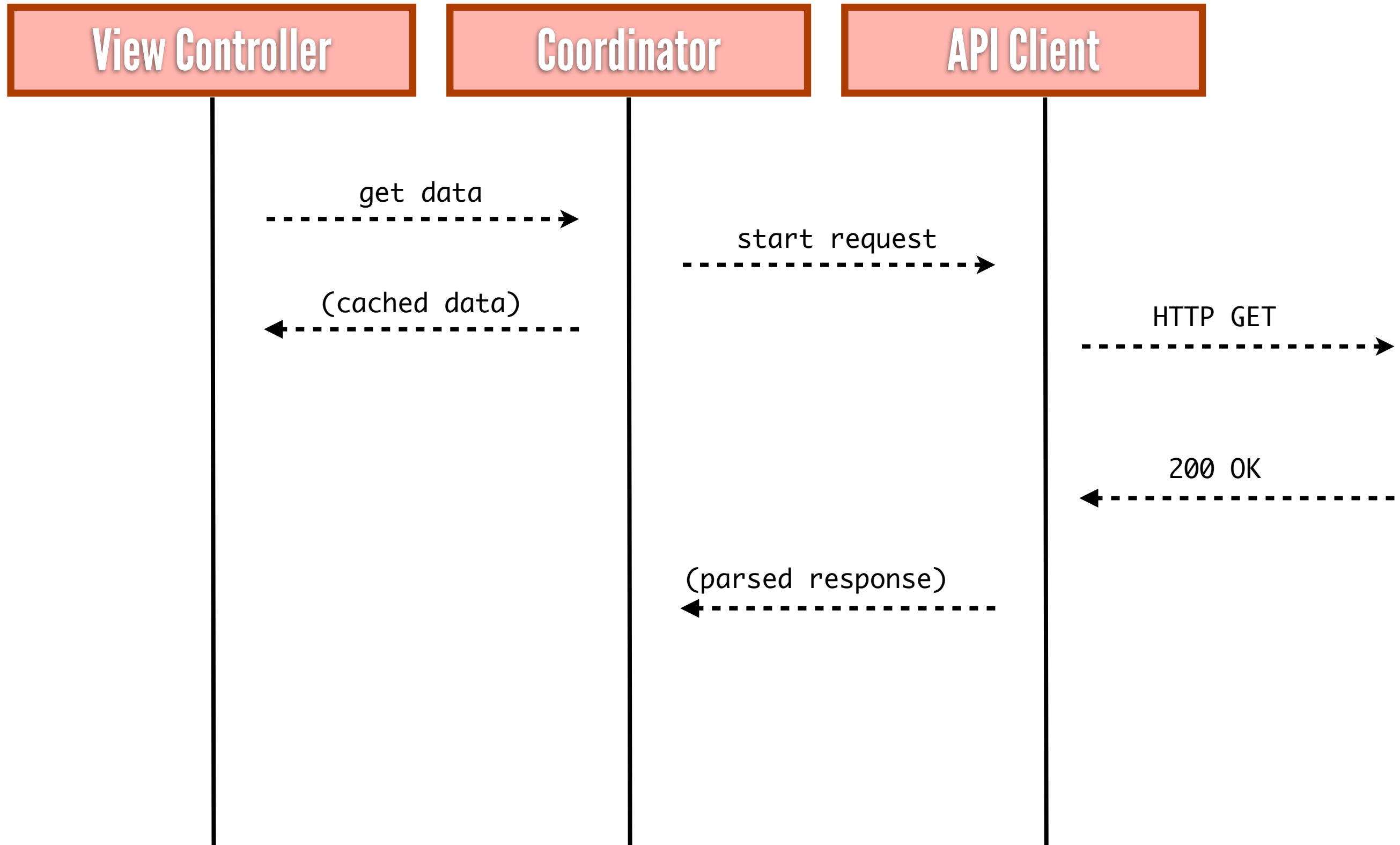
Switcheroo



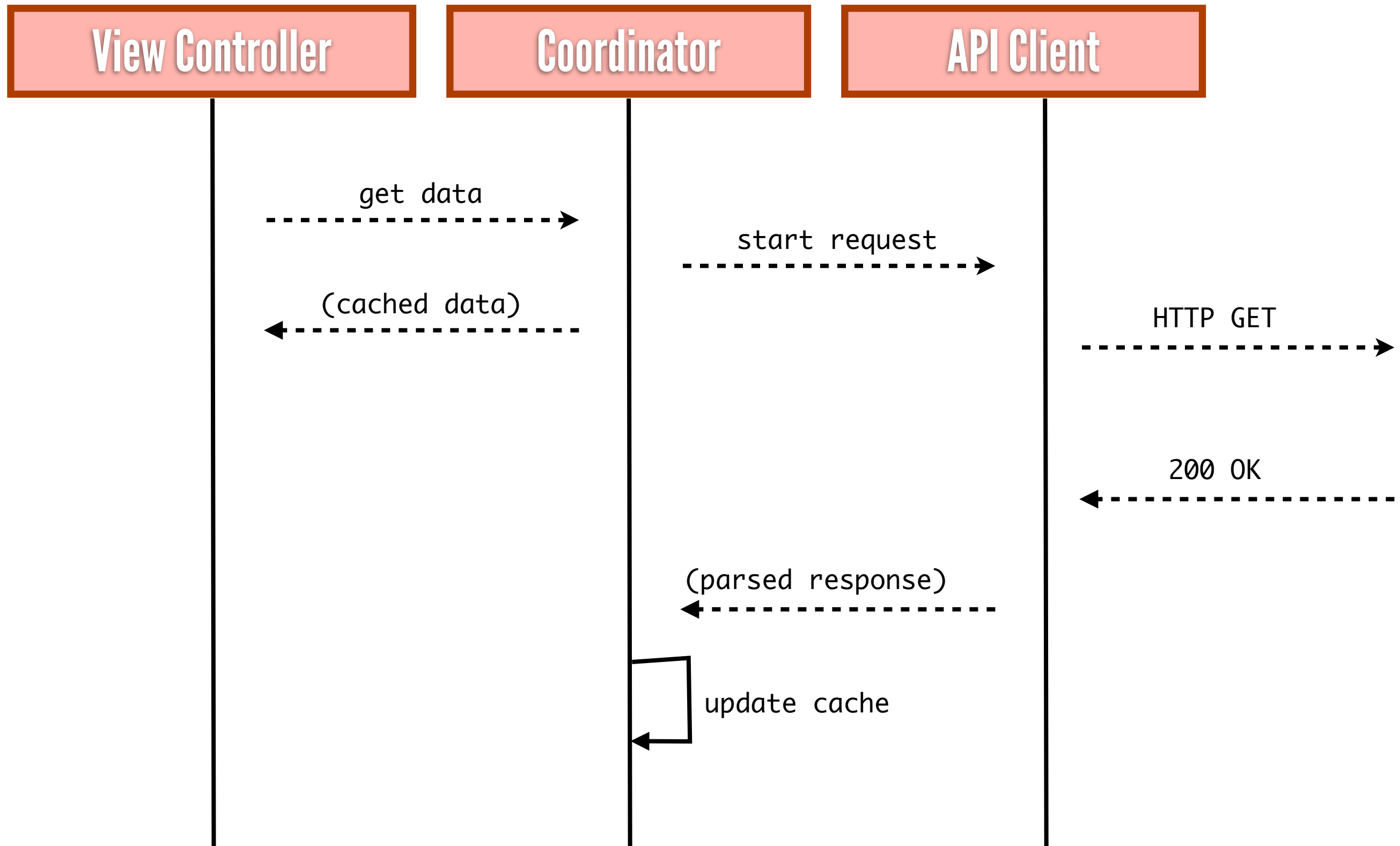
Switcheroo



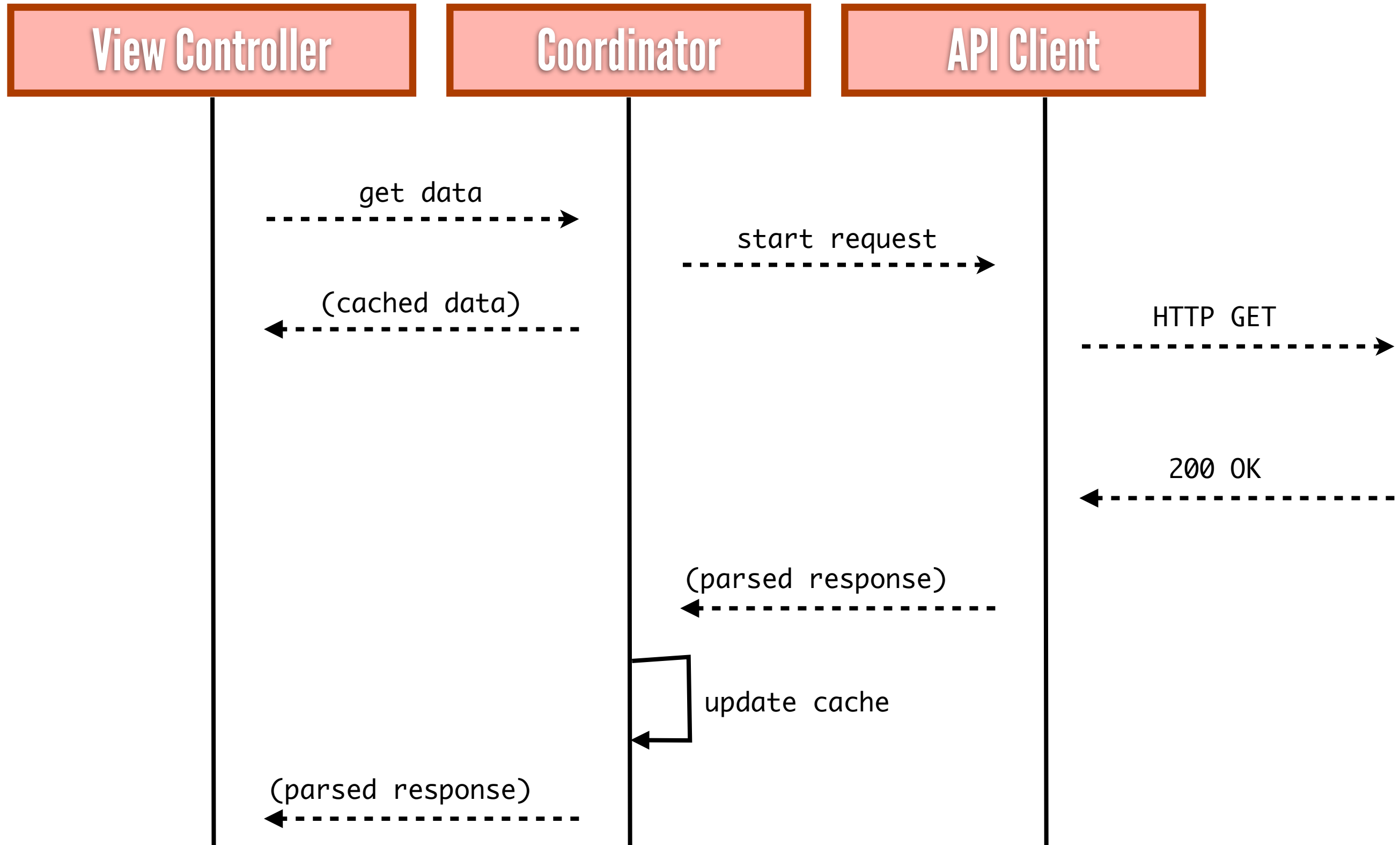
Switcheroo



Switcheroo



Switcheroo



API Design Tips

Don't expose your internal model



API Design Tips

Version your API



API Design Tips

**Send OS Version, App Version,
and Device Name as headers**



Device Name
OS Version
App Version

API Design Tips

**Have a way to notify clients
to upgrade (even force)**

API Design Tips

- **Techniques for permeating deletes**
 - **Valid ID Check**
 - **Client requests changes, server returns a set of new records, changed records, and deleted record ids**

API Design Tips

Page Large Datasets

API Design Tips

**Build-in HTTP Caching support
from day 1**

API Design Tips

**Take care when caching
user-specific data**

API Design Tips

Turn on Gzip Compression

API Design Tips

**Measure your API response
times**

Thank you!

Ben Scheirman
@subdigital
nsscreencast.com