NSURLProtocol

Timur Islamgulov

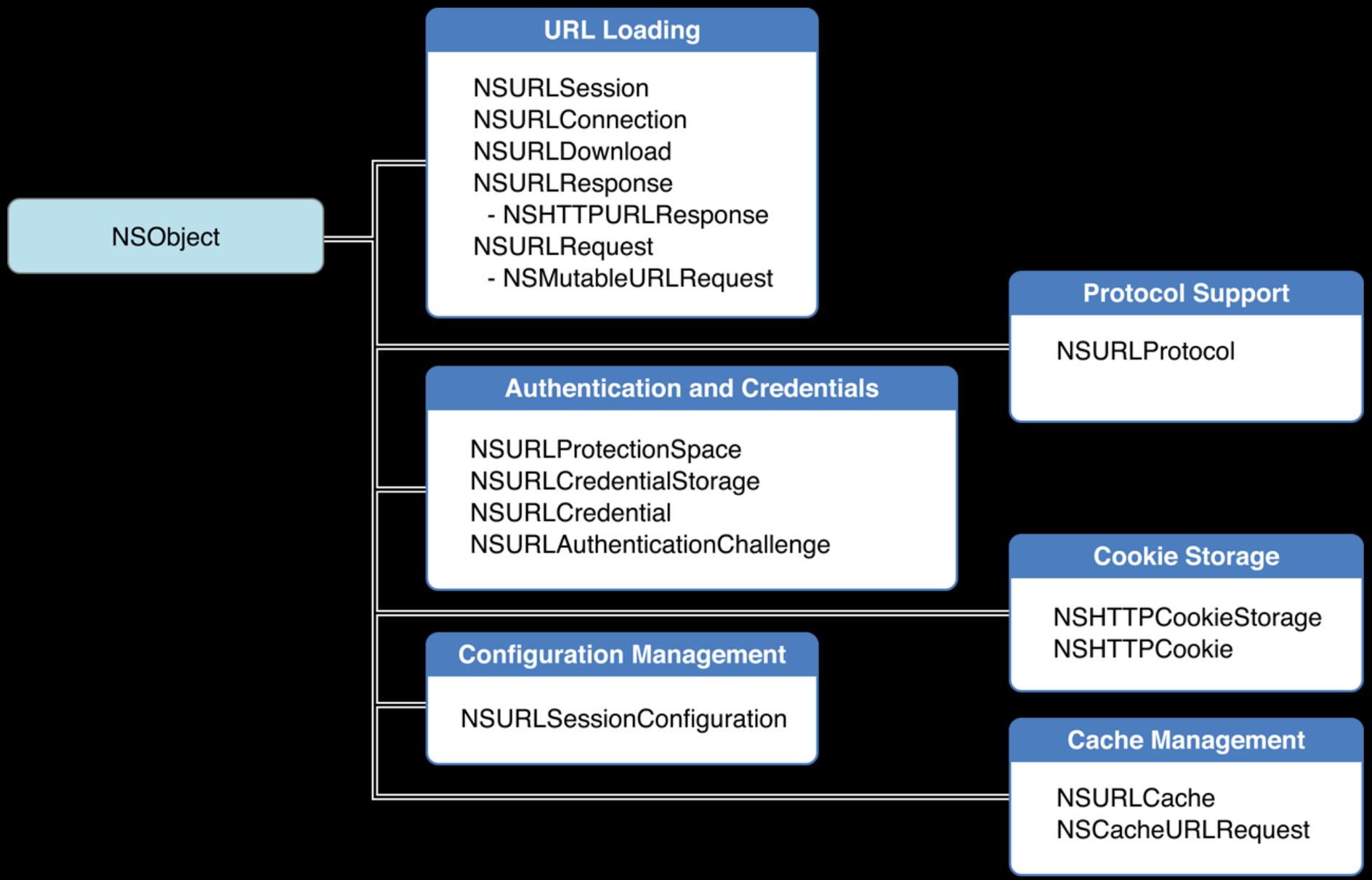
NSURLProtocol

- Overview
- Caching
- UIWebView and NSURLProtocol
- Tips & tricks

Overview

Overview

URL Loading System



Overview

NSURLProtocol

- It's an abstract class that allows subclasses to define the URL loading behavior of new or existing schemes.

- Custom protocols can configure the behavior of NSURLConnection, and NSURLSession-based networking facilities

- Apple-sanctioned man-in-the-middle attack.

Overview

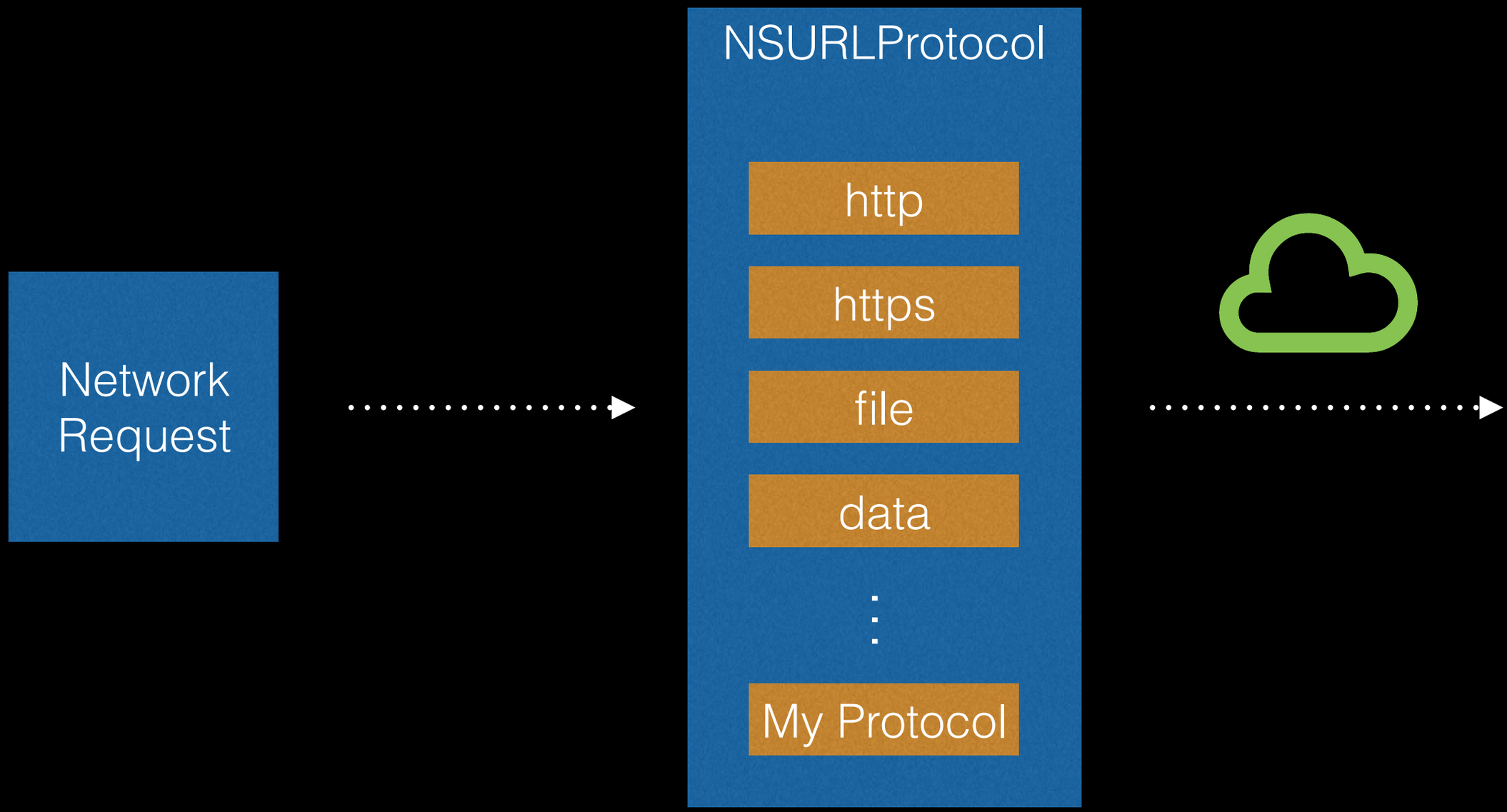
Use cases

- Logging network requests
- Modifying HTTP headers and parameters
- Providing locally stored data
- Mocking and stubbing HTTP responses for testing
- Using your own network protocol
- Intercepting, filtering, redirecting requests
- Implementing HTTP Live Streaming

Overview

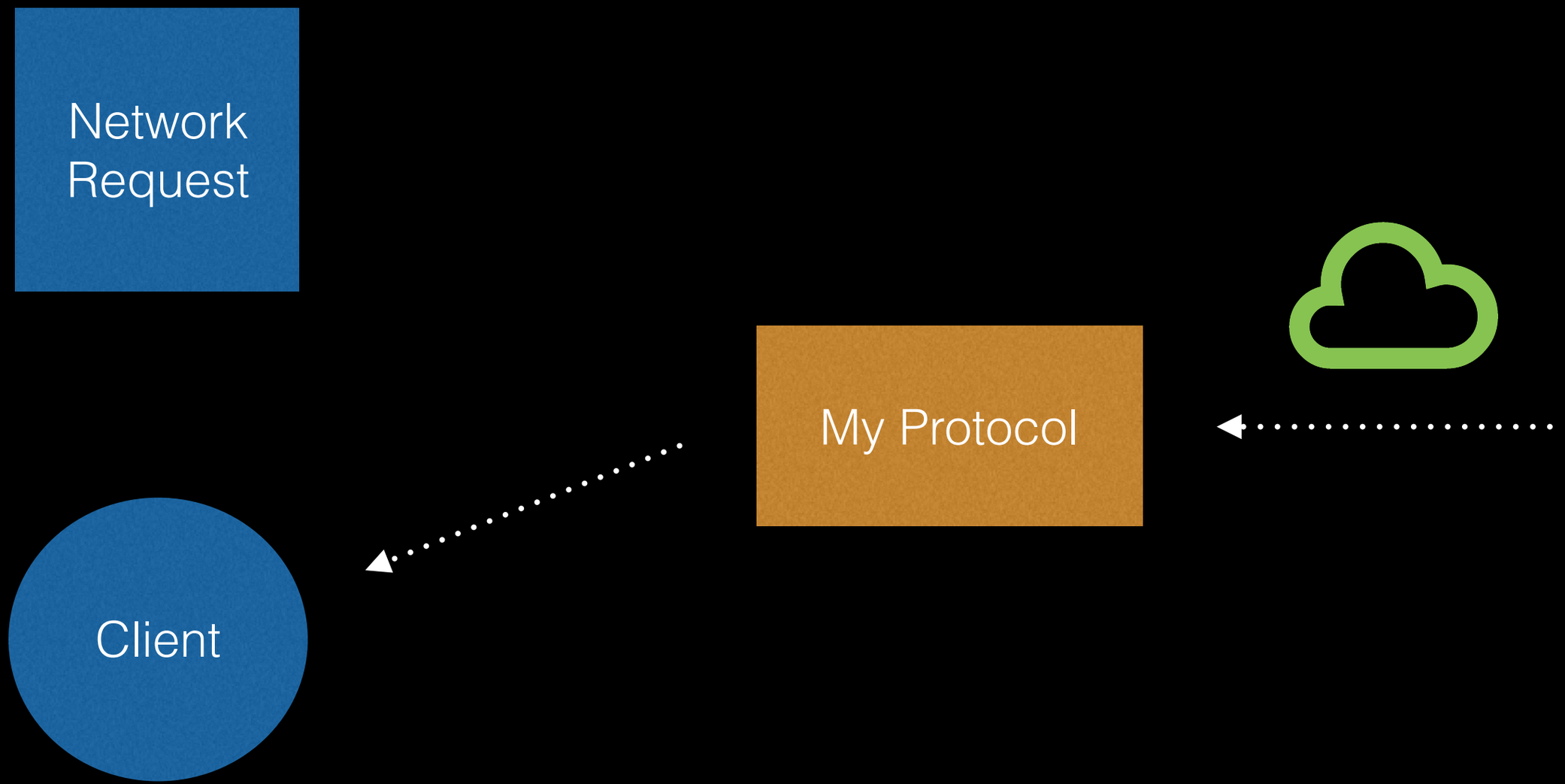
Overview

Implementation



Overview

Implementation



Overview

Implementation

```
@interface NSURLProtocol : NSObject

@property (readonly, retain) id <NSURLProtocolClient> client;

+ (BOOL)canInitWithRequest:(NSURLRequest *)request;

+ (NSURLRequest *)canonicalRequestForRequest:(NSURLRequest *)request;

- (void)startLoading;

- (void)stopLoading;

+ (id)propertyForKey:(NSString *)key
           inRequest:(NSURLRequest *)request;

+ (void)setProperty:(id)value
             forKey:(NSString *)key
          inRequest:(NSMutableURLRequest *)request;

. . .
```

Overview

Implementation

```
@protocol NSURLProtocolClient <NSObject>

- (void)URLProtocol:(NSURLProtocol *)protocol
 didReceiveResponse:(NSURLResponse *)response
 cacheStoragePolicy:(NSURLCacheStoragePolicy)policy;

- (void)URLProtocol:(NSURLProtocol *)protocol
        didLoadData:(NSData *)data;

- (void)URLProtocolDidFinishLoading:(NSURLProtocol *)protocol;

- (void)URLProtocol:(NSURLProtocol *)protocol
   didFailWithError:(NSError *)error;

- (void)URLProtocol:(NSURLProtocol *)protocol
wasRedirectedToRequest:(NSURLRequest *)request
   redirectResponse:(NSURLResponse *)redirectResponse;


. . .
```

Overview

Implementation

An essential responsibility for a protocol implementor is creating a NSURLResponse for each request it processes successfully.

Overview

Implementation

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    // Override point for customization after application launch.

    [NSURLProtocol registerClass:[TICachingProtocol class]];

    . . .

    return YES;
}

    . . .

  NSURLSessionConfiguration *configuration =
  [NSURLSessionConfiguration defaultSessionConfiguration];
  configuration.protocolClasses = @[[TICachingProtocol class]];
  NSURLSession *session = [NSURLSession sessionWithConfiguration:configuration];
```

Caching

Caching

- NSURLCache
- NSURLRequestCachePolicy

Caching

NSURLRequestCachePolicy

- NSURLRequestUseProtocolCachePolicy
- NSURLRequestReloadIgnoringLocalCacheData
- NSURLRequestReloadIgnoringLocalAndRemoteCacheData
- NSURLRequestReturnCacheDataElseLoad
- NSURLRequestReturnCacheDataDontLoad
- NSURLRequestReloadRevalidatingCacheData

Caching

NSURLRequestCachePolicy

- NSURLRequestUseProtocolCachePolicy
- NSURLRequestReloadIgnoringLocalCacheData
- NSURLRequestReloadIgnoringLocalAndRemoteCacheData
- NSURLRequestReturnCacheDataElseLoad
- NSURLRequestReturnCacheDataDontLoad
- NSURLRequestReloadRevalidatingCacheData

Caching

Request Cache Headers

- If-Modified-Since
- If-None-Match

Caching

Response Cache Headers

- Cache-Control
- Last-Modified
- Etag

Caching

Controlling Caching Programmatically

```
@protocol NSURLConnectionDataDelegate <NSURLConnectionDelegate>
@optional

- (NSCachedURLResponse *)connection:(NSURLConnection *)connection
                  willCacheResponse:(NSCachedURLResponse *)cachedResponse;

. . .

@end


@protocol NSURLSessionDataDelegate <NSURLSessionTaskDelegate>
@optional

- (void)URLSession:(NSURLSession *)session
          dataTask:(NSURLSessionDataTask *)dataTask
 willCacheResponse:(NSCachedURLResponse *)proposedResponse
completionHandler:
(void (^)(NSCachedURLResponse *cachedResponse))completionHandler;

. . .

@end
```

Caching

Rules

- The request is for an HTTP or HTTPS URL (or your own custom networking protocol that supports caching).
- The request was successful (with a status code in the 200–299 range).
- The provided response came from the server, rather than out of the cache.
- The NSURLRequest object's cache policy allows caching.
- The cache-related headers in the server's response (if present) allow caching.
- The response size is small enough to reasonably fit within the cache. (For example, if you provide a disk cache, the response must be no larger than about 5% of the disk cache size.)

Caching

Controlling Caching Programmatically

Open Radar

Community bug reports

NSURLRequestUseProtocolCachePolicy default expiration handling not documented

Originator:	daniel		
Number:	rdar://11541549	Date Originated:	5/26/2012
Status:	Open	Resolved:	No
Product:	Documentation	Product Version:	N/A
Classification:	Other bug	Reproducible:	N/A

```
/* Invoke the completion routine with a valid NSCachedURLResponse to
 * allow the resulting data to be cached, or pass nil to prevent
 * caching. Note that there is no guarantee that caching will be
 * attempted for a given resource, and you should not rely on this
 * message to receive the resource data.
 */
```

UIWebView
and
NSURLProtocol

UIWebView and NSURLProtocol

Implementing custom caching

UIWebView and NSURLProtocol

Workaround

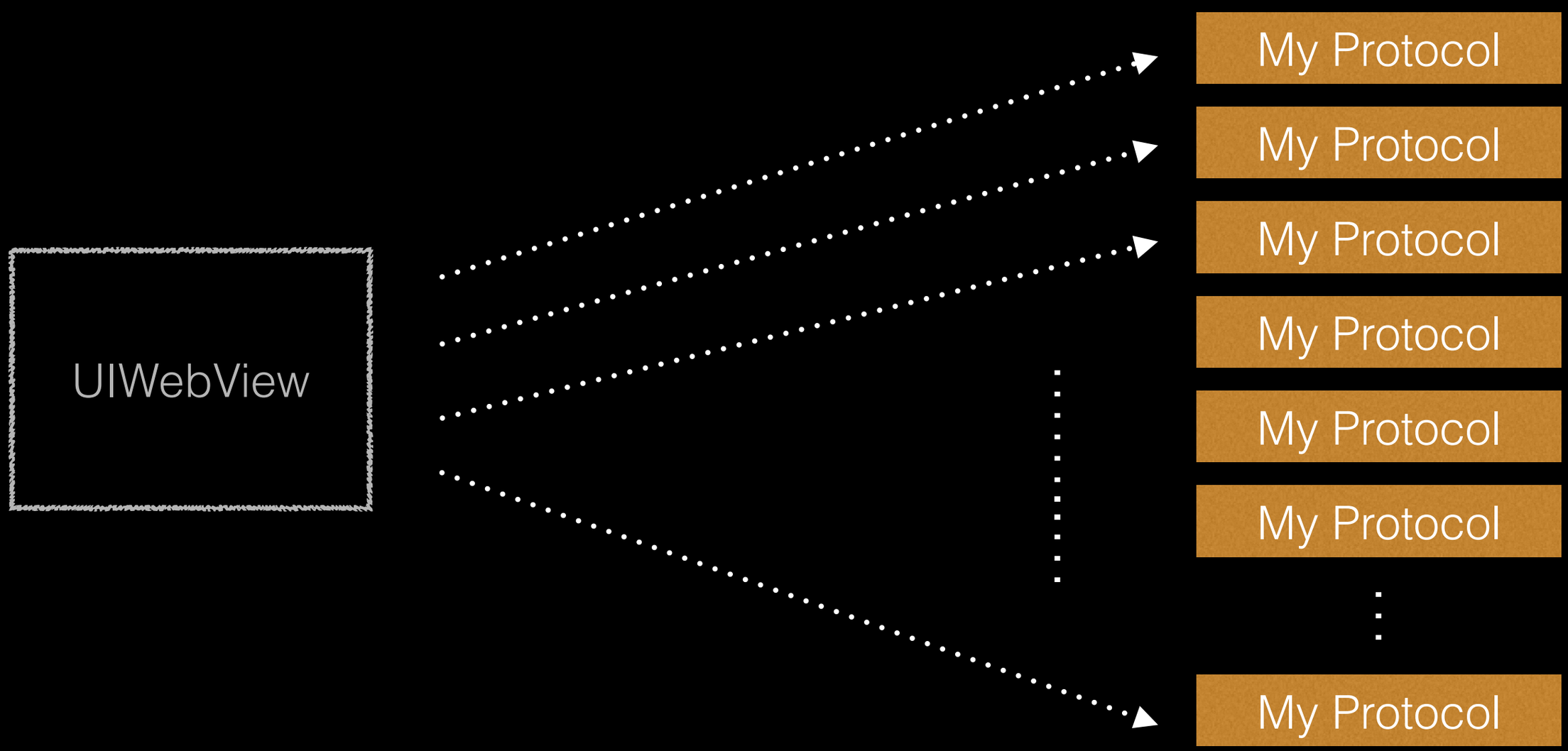
```
//  TIWebViewController.m


- (void)setupUserAgent {
    NSDictionary *dictionary = @{@"UserAgent" : self.userAgentIdentifier};
    [[NSUserDefaults standardUserDefaults] registerDefaults:dictionary];
    [self.webView loadRequest:
        [NSURLRequest requestWithURL:[NSURL URLWithString:@""]]];
}
```

UIWebView and NSURLProtocol

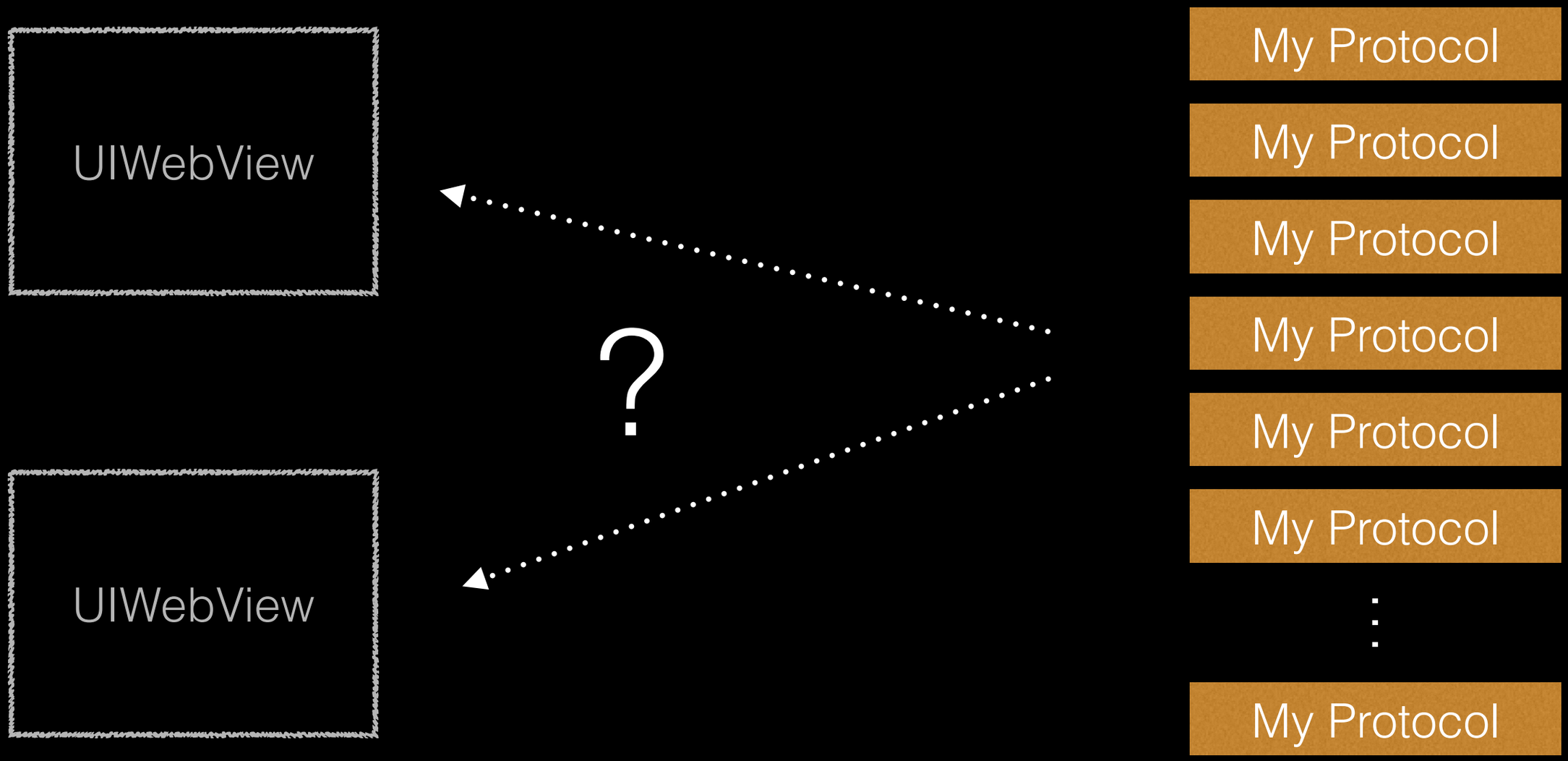
Filtering requests

```
//  TICachingProtocol.m


+ (BOOL)canInitWithRequest:(NSURLRequest *)request
{
    NSString *userAgent = [request valueForHTTPHeaderField:@"User-Agent"];
    BOOL canInit = ([[[request URL] scheme] isEqualToString:@"http"] ||
                    [[[request URL] scheme] isEqualToString:@"https"]) &&
                    ([userAgent isEqualToString:@"iPhone UA1"] ||
                    [userAgent isEqualToString:@"iPhone UA2"]) &&
                    ![NSURLProtocol propertyForKey:TICachingProtocolHandlerKey
                                         inRequest:request];

    if (canInit) {
        return YES;
    }
    return NO;
}
```

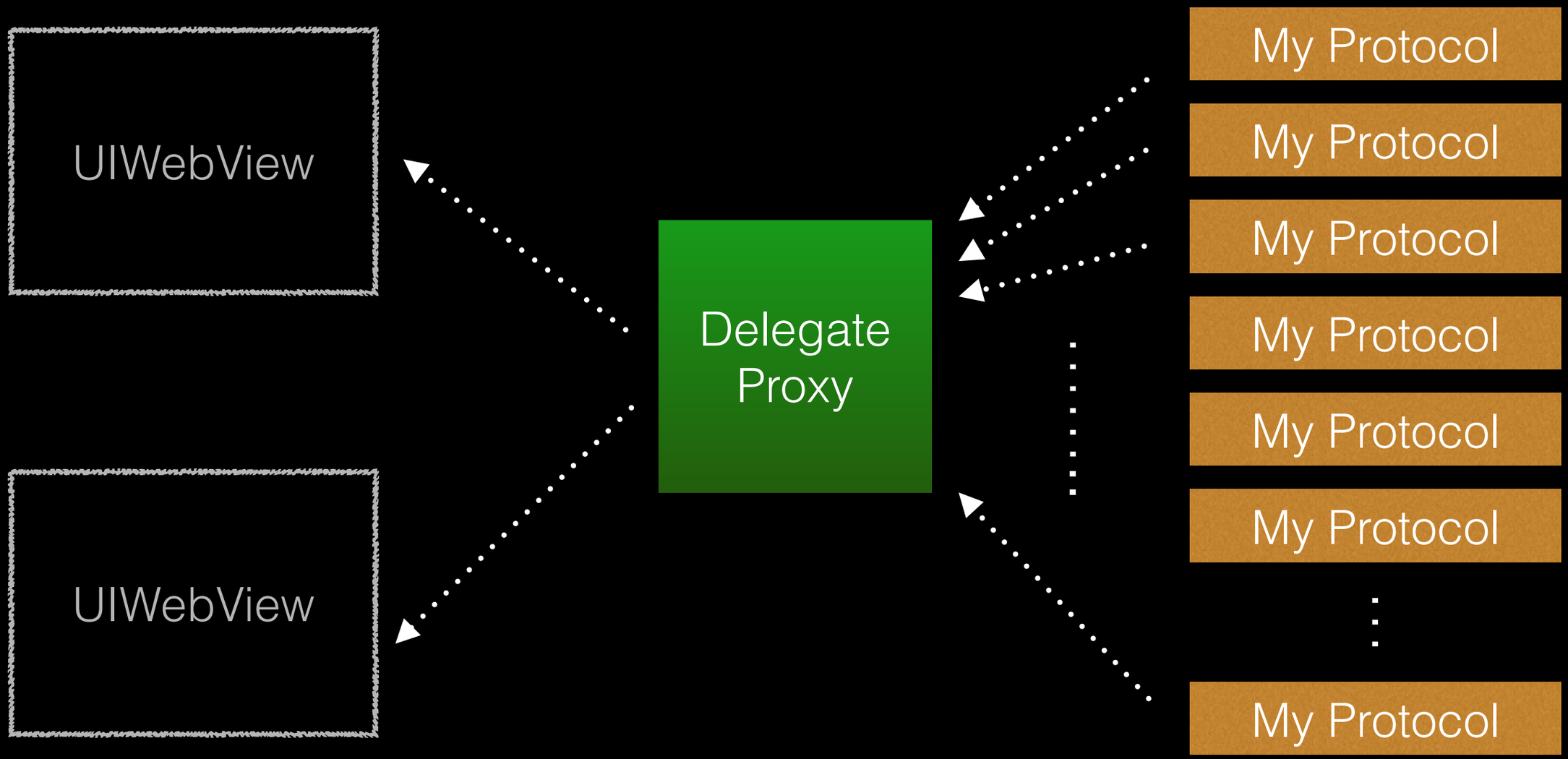
UIWebView and NSURLProtocol



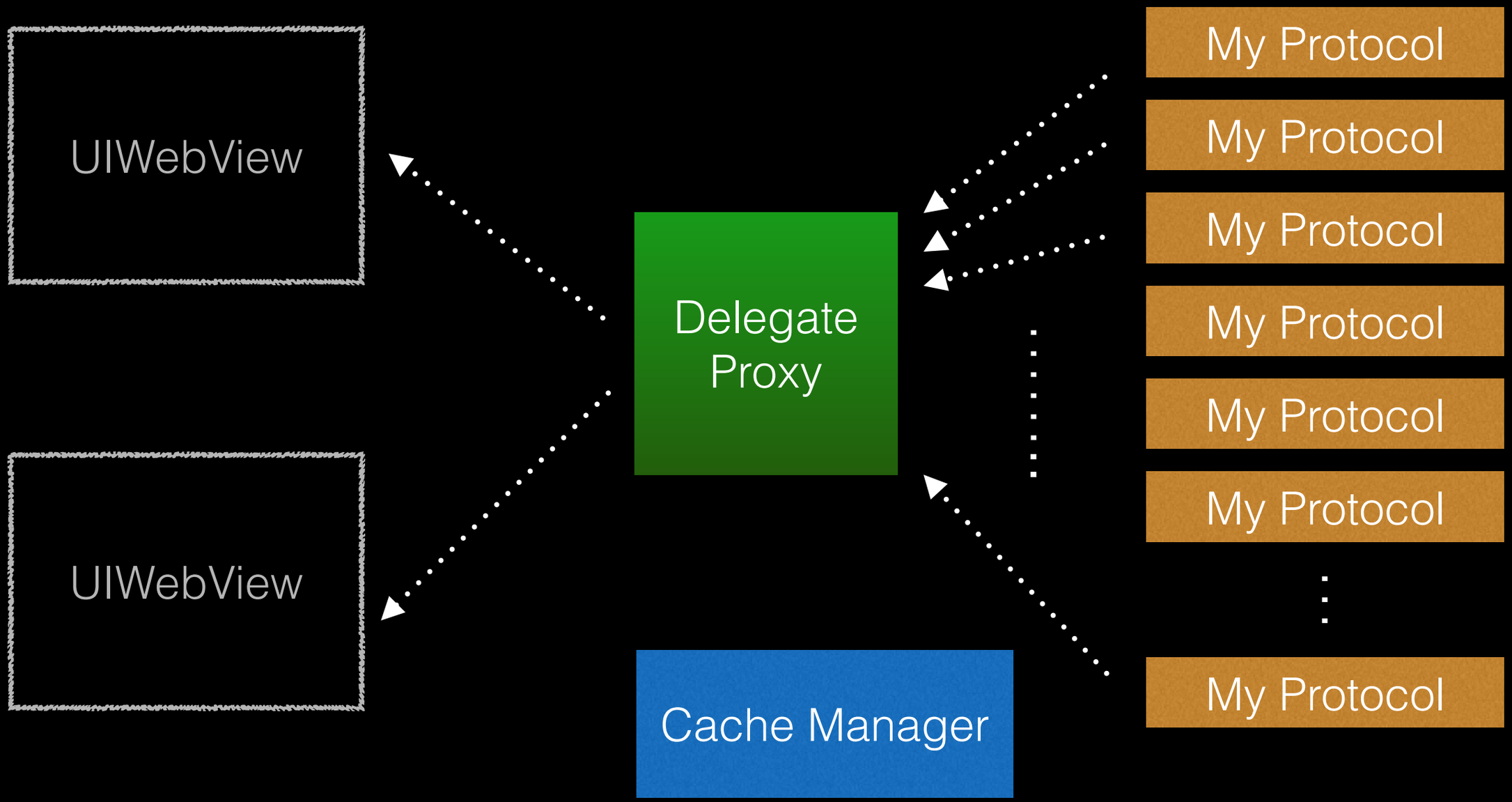
UIWebView and NSURLProtocol



UIWebView and NSURLProtocol



UIWebView and NSURLProtocol



UIWebView and NSURLProtocol

Demo

Tips & tricks

Tips & tricks

NSURLProtocolClient

- iOS 7 _NSCFURLProtocolBridgeWithTrampoline
- iOS 8 __NSCFURLProtocolClient_NS

Tips & tricks

Apple loves NSURLProtocol

- _NSCFWikipediaProtocol
- NSAboutURLProtocol
- _NSCFTranslatedFileURLProtocol
- _NSURLDataProtocol

Tips & tricks

HTTP Live Streaming

NSURLProtocol

Timur Islamgulov

timislamgulov@yahoo.com