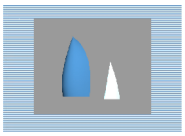


Smith's Point Analytics, LLC

Mobile Apps in the API Economy: Avoiding the Mobile Cliff

Authored by:
Peter Crocker

Principal Analyst
Smith's Point Analytics, LLC

This whitepaper was underwritten by:

moTwin Inc.

August 2013

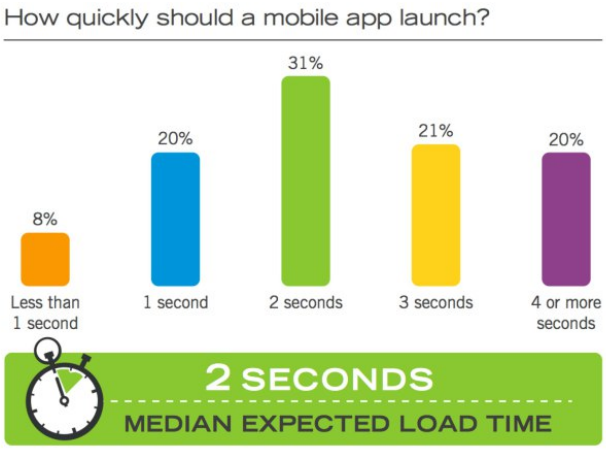
The growth of mobile technology has been nothing less than astounding, with an adoption rate faster than any technology in history. This explosive growth has lead to a fragmented ecosystem with the myriad of mobile devices in the market challenging developers to create compelling experiences across form factors and operating systems. Content formatted for a smartphone will not render correctly on a tablet or PC, vexing developers accustomed to a single ecosystem dominated by the Windows PC. The emergence of cloud services is also driving fragmentation on the back-end. The mobile device is maturing into "just another channel" and enterprises are scrambling to adapt. To capitalize on the opportunity to provide value added services to their customers through the mobile channel, enterprises need to expose their back-end systems to mobile developers. With increasing amounts of data and services available, developers are struggling to bringing together data from various sources such as internal/legacy enterprise systems, cloud services, and third parties to support their apps. Each of these sources provides data in different formats using different protocols, further complicating the issue.

This very fragmented world is driving demand for new computing architectures where the presentation layer is decoupled from the back-end data that powers a mobile app. Supporting developers with optimized data transfer from servers to devices and tools to manage complexity and fragmentation of both the client and back-end environments will be a key factor in the success of mobile strategies.

Fast and Engaging Apps Are Vital

In this hugely competitive mobile app market, the user experience is everything. Today's mobile user has limited patience for latency and expects app performance to be as good if not better than their experience on a wired PC. Research from Flurry Analytics shows that the median expected launch time for mobile apps is two seconds.

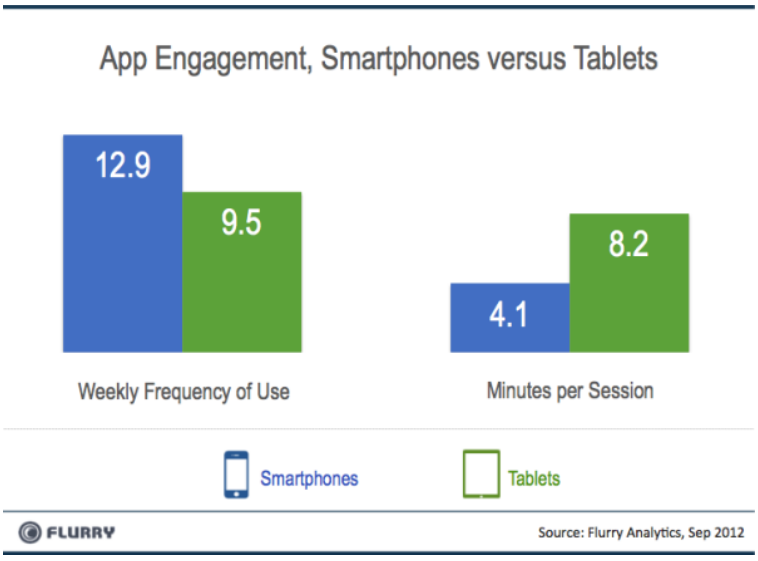
Figure 1 - Mobile App Launch Time Requirements



Source: Flurry Analytics

The demand for speed is also being driven by the short session times which are characteristic of usage patterns across mobile devices. Users of mobile apps are often multitasking and seek precise real-time data. Session times also differ across devices, making it essential for developers to tailor the app experience and performance to each device type. Developers can do this more effectively by creating front-ends tailored to the particular device while leveraging the same back-end for each experience.

Figure 2 – App Engagement Statistics



Source: Flurry Analytics

A number of factors can cause latency but the process of fetching data from a back-end system is a significant bottleneck slowing the performance of mobile apps. Today this issue is irrelevant to the majority of app developers as research conducted by mobile back end as a service (mBaaS) provider Kinvey found that only about one quarter of mobile apps are

connected to a back-end system. This figure is rapidly changing as an increasing number of savvy app developers are connecting to back-end data. Developers are leveraging back-end database to help drive revenues and according to Google, most successful apps are connected to some sort of server infrastructure. The ability for applications to access data in the cloud and dynamically provide context to the mobile experience provide significantly more value. In order to provide a rich high quality experience without latency, an optimized connection and data streams are paramount.

Mobile Cliff and Chatty APIs

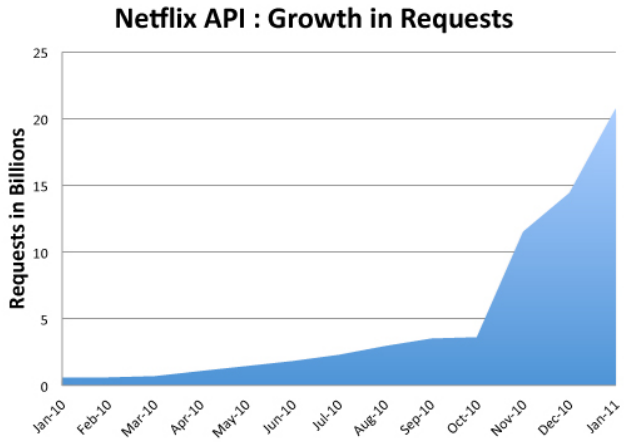
Developing for mobile devices is difficult and requires specialized skills that are hard to come by. The constraints inherent in mobile computing such as processing speed, power consumption, and network connectivity need to drive mobile architecture decisions. The fragmentation of mobile operating systems also makes it difficult for server developers to create applications for mobile clients. Likewise mobile developers who have built expertise in creating mobile clients do not possess the skills to build their own server stack to support their app. This "mobile cliff" is leading to numerous mobile project failures, particularly in the enterprise.

In an effort to leverage the mobile channel more effectively, enterprises are increasingly exposing data to mobile developers via API's. Traditional web service based on SOAP and designed for a world prior to ubiquitous mobile computing are being transformed to REST APIs that are better suited for mobile applications. REST APIs are lighter weight and easier to build but drawbacks to this architecture also exist.

REST APIs are designed to be very simple, easy to use, and granular. Mobile developers can get exactly the data that they need instead of making a call to a web service that serves entire data files containing data that is worthless to the application. Moreover, data providers don't need to worry about the myriad of mobile operating systems; form factors and screen sizes as REST APIs serve small bits of data in a uniform interface. While flexibility is very important, this fine granularity can lead to chatty applications and latency. Chatty applications are constantly making data requests and checking for updates. All these calls to the server via the API not only ties up bandwidth but also delays processes. Application processes are often dependent on each other so each call has to wait for the preceding call to return data to execute the next process. Chatty apps with many API calls also consume more battery power.

Netflix ran into this problem with their REST API a few years ago. As popularity of the API grew, API request exploded as detailed in the figure below.

Figure 3 – Growth of Request for Netflix API



Source: Netflix

Netflix needed a new solution to more effectively manage developer interaction with the Netflix API. This example is becoming more common as the API ecosystem grows.

Solutions

To continue to improve performance of mobile apps through reduced latency, mechanisms need to be implemented that enables a more efficient data transfer between mobile devices and servers.

One way that chattiness of APIs can be reduced is through orchestration and optimization. Resources exposed by REST APIs can be aggregated into a single API. This would enable a mobile developer to call on a single API to get all the necessary data instead of calling on each resource individually.

Caching strategies that stage data closer to the app so additional hops to the back-end systems can be reduced, can also be deployed. The performance of an application can also be enhanced by using updated and mobile optimized protocols.

Approaches to Reduce Chattiness and Cross the Mobile Cliff

API providers and app developers are approaching the challenges of optimizing data from a number of different angles including:

- Custom APIs for each use case
- Frameworks for server side code
- Middleware layer

In some situations, data providers can create custom APIs for a particular device type or use case. Unfortunately the expense of creating and managing custom APIs is significant, making this option only available to data provider's most important customers.

With data providers having little incentive to customize REST APIs for each app, solutions are emerging that implement a layer between the content gathering process of the API and the presentation layer on the device. This architecture allows the back-end data to remain simple and flat while data and content can be formatted for each endpoint. This layer can either be integrated with the server or as a standalone middleware layer.

Enterprises and some platform vendors have enabled developers to write custom code that resides within the API infrastructure. Orchestration platforms enable developers to create custom code that tailors APIs to serve only the data applicable to the app in one call, reducing chattiness. Netflix chose this approach and built their own solution while leading mBaaS providers have built this capability into their offering.

Middleware solutions can also improve performance for connected apps and ease development challenges. An independent middleware layer is able to abstract the complexity and fragmentation of client side operating systems and back-end data to vastly reduce the effort required to build and maintain mobile apps. Middleware can seamlessly connect to a myriad of back-end systems and cache data and stage it for deployment to the client, reducing a hop to the back-end system. By providing SDKs to mobile developers, middleware platforms can enable developers to easily create logic in the middleware layer to aggregate and orchestrate data and reduce chattiness. With data from multiple sources coming together in a single place, algorithms can also be implemented to add context to the available data, increasing its value.

The use of middleware also enables the use of optimized protocols that can speed data transfer. With middleware solutions able to install code on servers and clients, interoperability, a big benefit of open protocols, is less of a concern.

Protocols are Important Considerations

Protocol limitations can also cause data transfer bottlenecks. The HTTP protocol, which is the standard for web content, is aging and possesses a number of drawbacks:

- HTTP is only half duplex so messages can't be exchanged between server and client simultaneously. This increases latency as the server and client have to wait for a response to send additional data.
- Compression of HTTP data is optional and when data is compressed, only the content is compressed, not the header. The strategy is suboptimal as during the life cycle of the session, the compression dictionary is reset for each exchange.
- Data transfer can only be initiated by the client.

- Emerging protocols are improving performance of data transfer through full-duplex, multiplexing, full compression, and prioritization.

Figure 4 - Protocol Comparison

Protocol	Primary Benefits	Primary Drawbacks	Message or Streaming	Primary Application	Sponsor/ Owner
HTTP	• Open	• Half-duplex • Slow connection establishment ~ 200ms	• Streaming based	• General web content	• W3C Standard
SPDY	• Multiplexing • Header compression • Prioritization • Open	• Requires SSL • Not widely deployed	• Streaming based	• Optimized on per-domain basis	• Google
moTwin Protocol	• Multiplexing • Header compression • Binary serialization	• Closed	• Message based	• Designed for reliable high speed mobile data transfer	• moTwin, Inc.

Source: Smith's Point Analytics

SPDY is an upgrade to HTTP developed by Google that uses a TCP connection much more efficiently and is being used as the base for the HTTP 2.0 draft standard. The technology is able to compress headers and also enables full-duplex and multiplexing. Multiplexing enables a single server request to be divided into multiple requests to retrieve several objects. SPDY is also able to advise servers on the priority of the resources being requested and enables servers to initiate interaction with the client. One challenge that is hindering additional increases in performance is SPDY's reliance on SSL. While SSL provides encryption, its requirement adds some latency to the protocol. Guy Podjarny from Akamai did some benchmark testing of SPDY compared to HTTP and HTTPS in June 2012 with the results below. While SPDY is faster than secure HTTPS, the improvement is not significant.

Figure 5 - SPDY/HTTP Latency Comparison

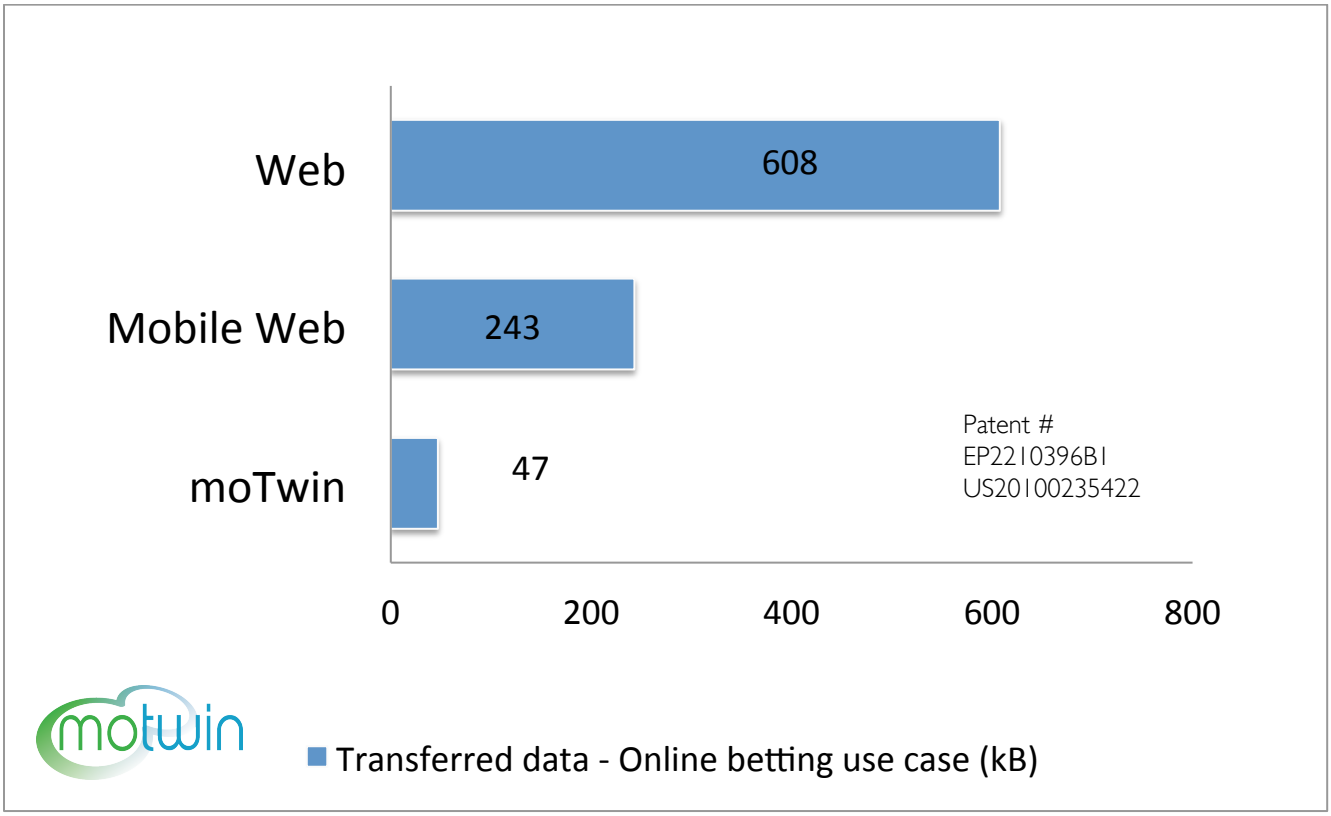
Network Speed (Down/Up Kbps, Latency ms)	SPDY vs. HTTPS	SPDY vs. HTTP
Cable (5,000/1,000, 28)	SPDY 6.7% faster	SPDY 4.3% slower
DSL (1,500/384, 50)	SPDY 4.4% faster	SPDY 0.7% slower
Low-Latency Mobile (780/330, 50)	SPDY 3% faster	SPDY 3.4% slower
High-Latency Mobile (780/330,200)	SPDY 3.7% faster	SPDY 4.8% slower

Source: Guy's Pod (Guy Podjarny)

Proprietary protocols such as moTwin's Protocol also provide features similar to SPDY including compression and multiplexing but unlike SPDY, which is a streaming based, the moTwin protocol is messaging based. The protocol was specifically designed for mobile so it

takes into consideration the network connectivity issues that are inherent in mobile. Because the protocol is message based, message can be re-transmitted if they fail to reach the client due to network failures. This feature is particularly important in enterprise mobility applications where reliable delivery of data is much more critical as financial transactions are often dependent on it. The performance of the moTwin protocol also clocks at a much greater speed than traditional web protocol such as HTTP. The moTwin protocol can also run over TCP or Web Sockets protocol.

Figure 6- moTwin Protocol vs. Web Performance Comparison



Source: moTwin, Inc.

Web Sockets is another important protocol as it is part of the HTML5 specification that features full-duplex capability that enables data to pass between server and client simultaneously. The protocol sets up an open connection that enables data to stream between the client and server. With this free flow of data between client and server, Web Sockets is well suited for live content and real-time games.

Key Take Aways

In order to create great mobile apps, developers need strong user experiences on the front end supported by strong back-ends. The complexity of the mobile ecosystem is not going away and mobile developers need tools that are flexible enough so developers can choose the appropriate technology and combination of tools for the job.

In a one size fits all HTTP/REST world, chatty apps are tying up bandwidth and increasing latency. Customization of data flows is key to managing this problem and middleware can be an important piece of tooling to help developers bridge the gaps in their skill sets.

Using the right protocols can also be an important factor for accelerating application performance. Proprietary protocols can fill gaps in the mobile ecosystem that are not addressed by open protocols.

The architecture of the mobile app has changed and the decoupling of the front-end and back-end and the exposure of enterprise data is enabling mobile developers to do things they have never been able to do before. The ability to access almost any data anywhere will spawn a new phase of innovation where the mobile device becomes another channel for enterprises to provide information and services.

About the author

Peter Crocker is the founder and principal analyst at Smith's Point Analytics LLC, a full service market research and consulting firm focused on the mobile and wireless industry. Peter has a decade of experience in the mobile industry. Prior to founding Smith's Point Analytics LLC in 2009, Peter was a Senior Analyst with VDC Research. Prior to Peter's work as an analyst he participated in starting and growing software ventures in the mobile space.