mobile measurements: the mobile app / OS perspective

Why understanding mobile app performance is hard
cellular performance matters to some apps

• most demanding apps
  • multiplayer gaming
  • VoIP
  • video chat

• somewhat demanding
  • web search
  • app interfaces with web services

• reasonably irrelevant
  • network transfers that aren’t in the critical path
app development environment
Network Speed Test

Last Test on A-MSFTWLAN

Start

Network Delay: 76 ms
Download speed: 2.3 Mbps
Upload speed: 0.7 Mbps

Current Network
Connection: WiFi
Network Name: A-MSFTWLAN
Internet: Connected

History
A-MSFTWLAN (Wi-Fi)
08/30/12
Latency: 76ms
Download: 21Mbps
Upload: 0.5Mbps

26.March.2014 sharad.agarwal@microsoft.com
The Weather Channel
“If I had a nickel for every time Mitt Romney said something stupid I’d be in his tax bracket”

“Mitt Romney might vote for Obama as well”

“We recommend Mitt Romney for president”

“I would definitely trust Mitt Romney with my money.”
“Mitt Romney might vote for Obama as well”

“We recommend Mitt Romney for president”

“If I had a nickel for every time Mitt Romney said something stupid I’d be in his tax bracket”

“I would definitely trust Mitt Romney with my money.”
Mitt Romney might vote for Obama as well

“We recommend Mitt Romney for president”

“I would definitely trust Mitt Romney with my money.”

“If I had a nickel for every time Mitt Romney said something stupid I'd be in his tax bracket”
Hypothetical Synchronous Code
Hypothetical Synchronous Code

```c
ClickHandler()
{

}
```
Hypothetical Synchronous Code

```java
ClickHandler()
{
    tweets = HttpGet(url);
}
```
Hypothetical Synchronous Code

```c
ClickHandler()
{
    tweets = HttpGet(url);
    rating = ProcessTweets(tweets);
}

ProcessTweets(tweets)
{
    ...
}
```
Hypothetical Synchronous Code

```c
ClickHandler()
{

tweets = HttpGet(url);
    rating = ProcessTweets(tweets);
        display.Text = rating;

}

ProcessTweets(tweets)
{
    ...
}
```
Hypothetical Synchronous Code

```c
ClickHandler()
{

tweets = HttpGet(url);
rating = ProcessTweets(tweets);
display.Text = rating;
}

ProcessTweets(tweets)
{
    ...  
}
```

Hypothetical Synchronous Code

ClickHandler()
{
    tweets = HttpGet(url);
    rating = ProcessTweets(tweets);
    display.Text = rating;
}

ProcessTweets(tweets)
{
    ...
}
Asynchronous Code

Mitt Rating 53%

26.March.2014 sharad.agarwal@microsoft.com 20
Asynchronous Code

```csharp
ClickHandler()
{
    AsyncHttpGet(url, DownloadCallback);
}
DownloadCallback(tweets)
{
    rating = ProcessTweets(tweets);
    UIDispatch(DisplayRating, rating);
}
DisplayRating(rating)
{
    display.Text = rating;
}
ProcessTweets(tweets)
{
    ...
}
```
Asynchronous Code

```java
ClickHandler()
{
    AsyncHttpGet(url, DownloadCallback);
}
DownloadCallback(tweets)
{
    rating = ProcessTweets(tweets);
    UIDispatch(DisplayRating, rating);
}
DisplayRating(rating)
{
    display.Text = rating;
}
ProcessTweets(tweets)
{
    ...
}
```

@ 26.March.2014 sharad.agarwal@microsoft.com
Asynchronous Code

```c
ClickHandler()
{
    AsyncHttpGet(url, DownloadCallback);
}
DownloadCallback(tweets)
{
    rating = ProcessTweets(tweets);
    UIDispatch(DisplayRating, rating);
}
DisplayRating(rating)
{
    display.Text = rating;
}
ProcessTweets(tweets)
{
    ...
}
```
Asynchronous Code

```c
ClickHandler()
{
    AsyncHttpGet(url, DownloadCallback);
}
DownloadCallback(tweets)
{
    rating = ProcessTweets(tweets);
    UIDispatch(DisplayRating, rating);
}
DisplayRating(rating)
{
    display.Text = rating;
}
ProcessTweets(tweets)
{
    ...
}
```

26.March.2014 sharad.agarwal@microsoft.com
Asynchronous Code

```csharp
ClickHandler() {
    AsyncHttpGet(url, DownloadCallback);
}
DownloadCallback(tweets) {
    rating = ProcessTweets(tweets);
    UIDispatch(DisplayRating, rating);
}
DisplayRating(rating) {
    display.Text = rating;
}
ProcessTweets(tweets) {
    ...
}
```
Asynchronous Code

```csharp
ClickHandler()
{
    AsyncHttpGet(url, DownloadCallback);
}

DownloadCallback(tweets)
{
    rating = ProcessTweets(tweets);
    UIDispatch(DisplayRating, rating);
}

DisplayRating(rating)
{
    display.Text = rating;
}

ProcessTweets(tweets)
{
    ...
}
```
Apps are highly asynchronous

30 popular apps
167,000 transactions from user study
Apps are highly asynchronous

- On average, **19 asynchronous calls** per user transaction
- On average, **8 parallel threads** per user transaction
Apps are highly asynchronous

- On average, 19 asynchronous calls per user transaction
- On average, 8 parallel threads per user transaction

For each user transaction, what was the critical path, and did the network matter?
Background Thread

Background Thread

Background Thread

UI Thread

User Transaction

Mitt Rating

47%

26.March.2014 sharad.agarwal@microsoft.com
Background Thread

Background Thread

Background Thread

UI Thread

User Click

User Transaction

Tweets

Posts

Mitt Rating

User Transaction

47%

Mitt Rating

47%
Critical Path

Optimizing the critical path reduces the user perceived delay
Optimizing the critical path reduces the user perceived delay.
key points

• app pages designed in GUI
• async interfaces in code to UI elements
• async interfaces to network
• tendency to fetch many network objects at launch
• fetch-parse-fetch pattern
• critical path is hard to determine
• in which user transactions does network matter?
• of those, what caused network delay, if any?
cellular network complicates this even more

• radio power states
cellular network complicates this even more

- radio power states
- latency varies by load
cellular network complicates this even more

- radio power states
- latency varies by load
cellular network complicates this even more

- radio power states
- latency varies by load
- latency varies over space
why am I telling you all this?

• wireless BW, latency, RSSI, etc. data is useful for research
• relate it to user experience
  • this is hard because of how apps are built
  • need detailed & efficient app instrumentation
• our mobile OS throttles some network behavior
  • based on app performance need and cellular data limits
  • research systems to do this on a per app transfer basis, using detailed & lightweight instrumentation
• need detailed, predictive wireless performance info & how individual parts of apps are affected by net perf
for more details

• cellular performance variations in short timescales
  • Switchboard paper in ACM MobiSys 2011

• asynchronous nature of apps & perf measurement
  • ApplInsight paper in USENIX OSDI 2012

• network fetching behavior of apps
  • Procrastinator paper in ACM MobiSys 2014