Mobile Software Development Framework: Mobile-Cloud Services (Push/Track)

10/18/2012
Y. Richard Yang

Outline
- Admin
- Mobile cloud service
  - Push notification service
  - Storage service
  - Track service
  - Split service

Admin.
- HW3 posted

Recap: Accessing Data in Cloud
- A typical design pattern is that a device updates/receives data in the cloud
  - Cloud as a rendezvous point
- Challenge: How do you keep data on a device fresh?

Recap: Solution Space
- Mobile poll
- Cloud push
  - Each app push
  - Shared (infrastructure) push

Shared Push Service
- A single persistent connection from device to a cloud push service provider
- Multiple application providers push to the service provider
- Service provider pushes to a device using the persistent connection
- Two examples
  - Apple Push Notification Service (APNS)
  - Google Cloud Messaging (GCM)
Design Requirements of a Shared Push Service

- **Security/Authorization**
  - Do not allow arbitrary app to push to a device

- **Scalability**
  - A large scale system may have millions of clients

- **Fault tolerance**
  - Client/device, push servers all can fail

- **Generality**
  - Can be used by diverse applications

Design Point: Authorization

Design 1: App does not know registered devices. Broadcast to all.

Design 2: App query registered devices. Multicast

Design 3: Device notifies registration ID to its server;

Design Point: What to Push?

- **Option 1**: Just push signal (data available) to devices and then devices fetch from app servers

- **Option 2**: push app data

Soft State Design

- State at the third party is soft state if the entity who sets up the state does not refresh it, the state will be pruned at the 3rd party

Apple Push Notification Service

- iOS device maintains a persistent TCP connection to an Apple Push Notification Server (APNS)

- A push notification from a provider to a client application

- Multi-providers to multiple devices
### APNS Authorization: Device Token

- Device token contains information that enables APNs to locate the device.
- Client app needs to provide the token to its app provider.
- Device token should be requested and passed to providers every time your application launches.

### Apple Push Notification Data

- Each push notification carries a payload.
  - 256 bytes maximum.
  - Best effort delivery.
- App provider provides a JSON dictionary object, which contains another dictionary identified by the key `aps`.
- `aps` specifies the following actions:
  - An alert message to display to the user.
  - A number to badge the application icon with.
  - A sound to play.

### APNS Example: Server

1. $devicetoken = "F05571f4b6b6e4132447e6a36b02132f438527f6470c46c68d0ff6b877f";
2. // Put your private key's passphrase here:
3. $passphrase = 'PushChat';
4. // Put your alert message here:
5. $message = "CS454: my first push notification!";
6. $ctx = stream_context_create();
7. stream_context_set_option($ctx, 'ssl', 'local_cert', '/path/to/cert.pem');
8. stream_context_set_option($ctx, 'ssl', 'passphrase', $passphrase);
9. // Open a connection to the APNS server
10. $fp = stream_socket_client( //gateway.sandbox.push.apple.com:2195', $err, $errstr, 60, STREAM_CLIENT_CONNECT | STREAM_CLIENT_PERSISTENT, $ctx);
11. if ($fp) {
12.    exit("Failed to connect: $err $errstr", PHP_EOL);
13. } else {
14.    echo 'Connected to APNS'. PHP_EOL;
15.}

### APNS Example: Client

1. - [UIApplication sharedApplication].registerForRemoteNotificationTypes = [UIRemoteNotificationTypeBadge UIRemoteNotificationTypeSound UIRemoteNotificationTypeAlert];
2. if (!$result) {
3.    return YES;
4. } else {
5.    // Send it to the server
6.    $msg = chr(0)
7.    // Build the binary notification
8.    $payload = json_encode($body)
9.    // Encode the payload as JSON
10.   $payload = pack('n', strlen($payload)) . $payload;
11.   // Create the payload body
12.   $message = 'My token is: %1$s', deviceToken;
13.   $message = echo $message, $message;
14.   // Open a stream to connect
15.   $fp = stream_socket_client( //gateway.sandbox.push.apple.com:2195', $err, $errstr, 60, STREAM_CLIENT_CONNECT | STREAM_CLIENT_PERSISTENT, $ctx);
16.   if ($fp) {
17.      exit("Failed to connect: $err $errstr", PHP_EOL);
18.   } else {
19.      echo 'Connected to APNS'. PHP_EOL;
20.   }

### APNS Example: Server (cont')

16. // Create the payload body
17. $body = array(
18.    'aps' => array,
19.    'alert' => $message,
20.    'sound' => 'default'
21.);
22. $payload = json_encode($body); // Encode the payload as JSON
23. // Build the binary notification
24. $payload = json_encode($body); // Build the binary notification
25. $payload = pack('n', strlen($payload)) . $payload;
26. // Send it to the server
27. $result = fwrite($fp, $msg, strlen($msg));
28. if (!$result) {
29.    exit("Failed to connect: $err $errstr", PHP_EOL);
30. } else {
31.    echo 'Message successfully delivered'. PHP_EOL;
32. }
33. // Close the connection to the server
34. fclose($fp);

### Google Cloud Messaging

- Very similar to APNS.

GCM Flow: App Developer Registration
- App developer registers a project at Google
  - Open API console: https://code.google.com/apis/console/
- After Create project
    - Project ID
    - Sender ID

GCM Flow: Device App Registration
- Enable cloud to device messaging in your app
  - Add permissions in Manifest
  - App (on device) registers with Google to get registration ID
- GCMIntentService
  - // called after GCM library finishes registration
  - onRegistered(Context context, String regId);
  - onUnregistered(Context context, String regId);
  - // called after your server sends a msg to GCM, and
  - // GCM sends to this device
  - onMessage(Context context, Intent intent);
  - onError(Context context, String errorId);
  - onRecoverableError(Context context, String errorId)

Device App Handle Events
- The GCMBroadcastReceiver (defined in GCM library) handles the broadcast messages, and calls methods defined in GCMIntentService, if you define this service

GCM Flow: Summary
- Enabling cloud to device messaging
  - App (on device) registers with Google to get registration ID
  - App sends registration ID to its App Server
- Per message
  - App Server sends (authenticated) message to Google
  - Google sends message to device, which sends to app
- Disabling cloud to device messaging
  - App can unregister ID, e.g., when user no longer wants push
**Additional Details**


**Discussion: Mobile Cloud Services**

- We have discussed push notification service. What other services can you think of?

**Example Mobile Cloud Services**

- Push notification service
- Location based service
  - Track service (supporting location based services)
- Storage and sync
  - Syncing and storage service (iCloud)
- Proxy service (Kindle Split Browser)
- Recognition services
  - Speech to text/text to speech service
  - Natural language processing service (open Siri API for 3rd party applications in the future)

**Outline**

- Admin
- Mobile cloud service
  - Push notification service
  - Track service
    - StarTrack Next Generation: A Scalable Infrastructure for Track-Based Applications, by Maya Haridasan, Iqbal Mohamed, Doug Terry, Chandramohan A. Thakkath, and Li Zhang, in OSDI 2010.

**Location-Based Applications**

- Many phones already have the ability to determine their own location
  - GPS, cell tower triangulation, or proximity to WiFi hotspots
- Many mobile applications use location information

**A Common Abstraction: Track**

Time-ordered sequence of location readings

![Map](image)
**Application: Personalized Driving Directions**

Goal: Find directions to new gym

![Map of directions](image)

**A Taxonomy of Applications**

<table>
<thead>
<tr>
<th>Current location</th>
<th>Personal</th>
<th>Social</th>
</tr>
</thead>
<tbody>
<tr>
<td>Driving directions, Nearby restaurants</td>
<td>Friend finder, Crowd scenes</td>
<td></td>
</tr>
</tbody>
</table>

| Past locations | Personal travel journal, Geocoded photos | Post-it notes, Recommendations |

| Tracks | Personalized Driving Directions, Track-Based Search | Ride sharing, Discovery, Urban sensing |

**Class of applications enabled by StarTrack**

---

**StarTrack System**

- Insertion
- Retrieval
- Manipulation
- Comparison

- ST Client
- Location Manager
- Application

<table>
<thead>
<tr>
<th>Inserción Application</th>
<th>Location Manager</th>
<th>ST Client</th>
</tr>
</thead>
</table>

**System Challenges**

1. Handling error-prone tracks
2. Flexible programming interface
3. Efficient implementation of operations on tracks
4. Scalability and fault tolerance

---

**Challenges of Using Raw Tracks**

Advantages of Canonicalization:
- More efficient retrieval and comparison operations
- Enables StarTrack to maintain a list of non-duplicate tracks

---

**StarTrack API: Track Collections**

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC MakeCollection(GroupCriteria criteria, bool removeDuplicates)</td>
</tr>
<tr>
<td>TC JoinTrackCollections([TC ICs]), bool removeDuplicates)</td>
</tr>
<tr>
<td>TC SortTracks(TC IC, SortAttribute attr)</td>
</tr>
<tr>
<td>TC TakeTracks(TC IC, int count)</td>
</tr>
<tr>
<td>TC GetSimilarTracks(TC IC, Track refTrack, float simThreshold)</td>
</tr>
<tr>
<td>TC GetPassByTracks(TC IC, Area) areas)</td>
</tr>
<tr>
<td>TC GetCommonSegments(TC IC, float freqThreshold)</td>
</tr>
</tbody>
</table>

Pre-filter tracks ➔ Manipulate tracks ➔ Fetch tracks
**API Usage: Ride-Sharing Application**

```java
// get user's most popular track in the morning
TC myTC = MakeCollection("name = Maya", [0800 1000], true);
TC myPopTC = SortTracks(myTC, FREQ);
Track track = GetTracks(myPopTC, 0, 1);

// find tracks of all fellow employees
TC msTC = MakeCollection("name.Employer = MS", [0800 1000], true);

// pick tracks from the community most similar to user's popular track
TC similarTC = GetSimilarTracks(msTC, track, 0.8);
Track[] similarTracks = GetTracks(similarTC, 0, 20);

// Verify if each track is frequently traveled by its respective owner
User[] result = FindOwnersOfFrequentTracks(similarTracks);
```

**Track Similarity**

![Track Similarity Diagram](image)

**Summary**

- The Track abstraction is simple but quite interesting
- Think about abstractions in your project