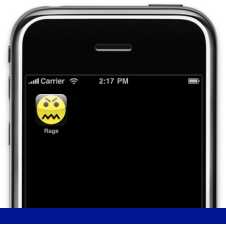


Rage App!




Neil Bell, Michael Hsueh, Nicholas Jalbert, Barret Rhoden

Library Support for Integrating Mobile Devices with Cloud Services.

Introduction

- The future of computing will be mobile devices working in conjunction with the cloud.
- Our goal is to develop an easy-to-use, flexible framework for sending key-value pairs from a mobile device to a back-end in the cloud.
- We identified a desirable set of framework characteristics:
- **Flexible**—easy extension of messages to accommodate the growing number of mobile sensors.
- **Dynamic**—network aware and able to change or delay messages based on quality of connection.
- **Robust**—seamlessly handles multiple data formats.

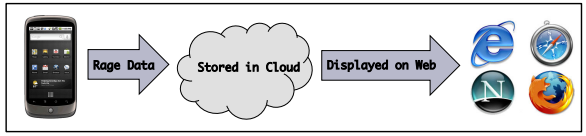
A Key-Value Framework

- Cooperative components in both the mobile client and the back-end web server.
  - Encapsulates connectivity and serialization behind a simple key-value interface on the client side.
  - The websites sees the message as a simple XML message.
- Allows for **mobile-aware intelligence** about when to send data:
  - If there is no connection, save the message until there is a connection.
  - For slow connection and a large multimedia message, delay message send.
- **Client Side API Example:**

```
msg = new MobKVMessage(URL);
msg->addKeyValue(String, String);
msg->addKeyValue(String, Blob);
msg->send();
```

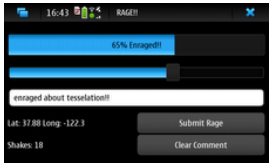
Rage App

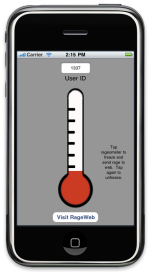
- For proof of concept, we built the **Rage App** using our framework prototype.
- Rage App allows users to express their anger at a particular location for a particular reason.
  - Accelerometers detect rage amount, GPS detects location.
- Use our framework to upload key-value pairs of sensor readings and other information into the cloud.



Implementation

- Built Rage App for Nokia N900 and Apple iPhone.
  - Implemented our communication library in both C++ and Objective C.
  - Rage App captures accelerometer and GPS data and sends it to the cloud.
- Nokia N900 provides a standard Linux platform with the QT UI and application framework.
  - Familiar model for Linux programmers.
  - Very portable—execute on desktop and phone.
- Apple requires all applications use their proprietary SDK.
  - Objective C is object oriented so our communication library implemented as a set of classes.
  - Interoperability is (by design) discouraged.





Cloud

- Back-end implemented in PHP, accepts serialized data using POST submissions from mobile device.
- Uses SimpleXML Library to parse XML key-value pairs into PHP array.
- Application-specific customization needs only a single line of code modified.
- Database uses a column-per-key format. Requires one-time setup for implementation.
  - Customizable for any schema.

Back-end mashup with Google Maps

```
<incident>
  <uid />
  <timestamp />
  <rage />
  <comment />
  <lat />
  <long />
</incident>
```

Serialized XML message format

Future Work/Conclusion

- Extend support to additional data interchange formats (e.g., JSON, YAML).
- Connectivity awareness and deferral of transfers over slow connections.
- Built-in framework support for “standard” sensors like microphone, camera.
- Integration with additional cloud services like social networking and route calculation.