SWAN-Fly : A flexible cloud-enabled framework for context-aware applications in smartphones

Roshan Bharath Das, Aart van Halteren, Henri Bal
Vrije Universiteit Amsterdam, The Netherlands
{r.bharathdas, a.t.van.halteren, h.e.bal}@vu.nl

Introduction

- Growth in no. of onboard sensors
- Interesting context-aware apps
  → Runtastic
  → Shazam
- Poor sensor abstraction and programming support
- Middleware solution - SWAN
- Limited resources on smartphones
- Need for a flexible cloud solution

Fig. 1: Big Data Analytics

Problem

- **Tight coupling** between the app and the cloud

Goal

- To help the application developers **to easily choose** the preferred cloud provider

Contribution

Server Connection Module

- Supports REST-based cloud solutions
- Lets developer choose the preferred cloud provider

Extended SWAN-Song language

- A domain specific language to build context expressions
- An expression - few lines of code
- Preference screen (Fig. 3) for easy tuning

SWAN-Fly Architecture

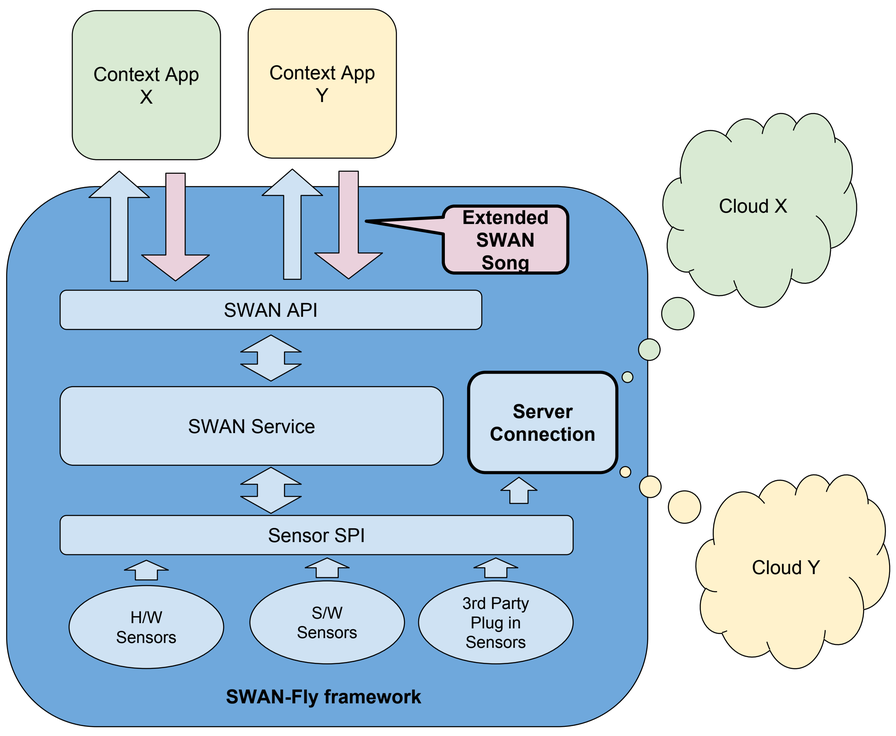

Fig. 2: **Architecture of SWAN-Fly**

Extended SWAN-Song Language

self@light:lux#endpoint$_1${MAX, 1000ms}

endpoint$_1$={
url = https://api.abc.com/update.json,
http_auth = NoAuth,
http_header = null,
http_body = api key : XXXXX,
http_body_type = formdata,
http_method = PUT
}

Application developers can register this expression to fetch light sensor data from the smartphone and send it to the endpoint$_1$

Mobile Screenshot

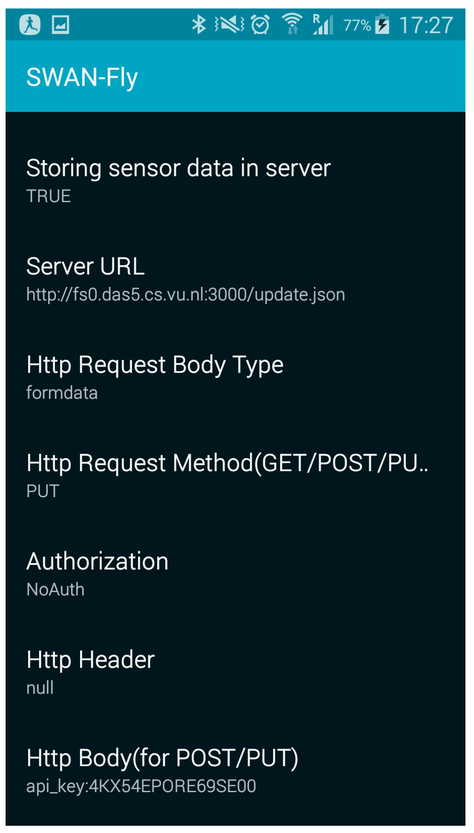

Fig. 3: **Server Configuration Parameters**

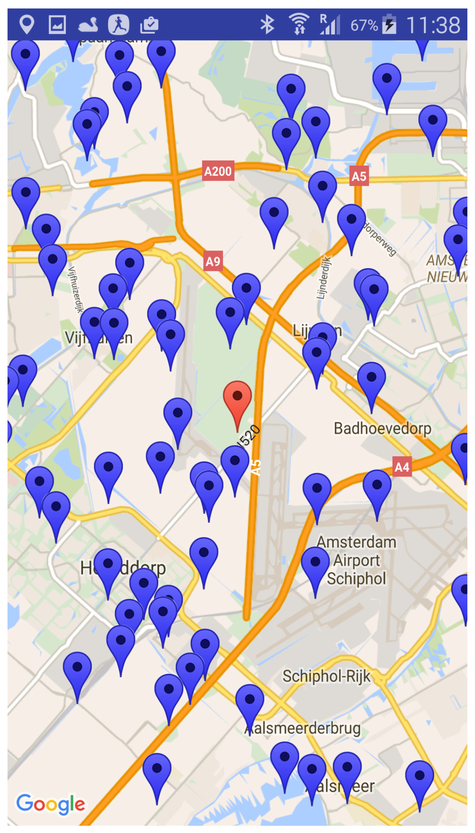

Fig. 4: **Crowd Monitoring Application**

Results

- **Ease of use**
  → Comparison of Crowd Monitoring app using SWAN v/s SWAN-Fly

Functionality	SWAN	SWAN-Fly
SWAN-Song expression	2	10
SWAN API	28	28
HttpUrlConnection	44	0
AsyncTask	15	0
Other	132	132
Total (LOC)	221	170

Fig. 5: **Lines of code that the application developers need to write**

- **Flexibility**
  → Tested with two different cloud solutions
  * **ThingSpeak** server
  * **Django** server