MoSCA: Service Composition in Mobile Environments

Lucia Del Prete*
Dept. of Computer Science
University College London
Gower Street, London, WC1E 6BT, UK
L.DelPrete@cs.ucl.ac.uk

Licia Capra
Dept. of Computer Science
University College London
Gower Street, London, WC1E 6BT, UK
L.Capra@cs.ucl.ac.uk

ABSTRACT

We present MoSCA, a run-time framework for the discovery and composition of services in mobile environments. MoSCA combines information about users' historical mobility patterns, together with composition semantics, to maximise the chances of successfully consumed compound services.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures; C.2.4 [**Distributed Systems**]: Distributed Applications

General Terms

Design, Reliability

Keywords

Service Composition, Mobility, Middleware

1. SCENARIO

In the last few years, two major trends have been observed: first, the enormous evolution (and market penetration) of mobile technology. Mobile phones have seen their computing capabilities increase according to Moore's law; they have been enriched with additional functionalities (e.g., cameras, MP3 players, and GPS receivers), and integrated with a variety of wireless network technologies of increasing bandwidth (e.g., Bluetooth 2, Zigbee, WiFi and WiMax), thus enabling the on-the-fly creation of networks of devices in proximity. Second, the Internet has seen a proliferation of blogs and personal content spaces, revealing a transformation of users from traditional consumers to active producers of content. It will not be long before these two trends will converge, thus creating an integrated environment where, besides traditional services delivered by powerful server machines accessible via wide area networks, new services and content will be offered to users by users on the go. These fine grained services, attached to people and the environment, will need to be composed to deliver more sophisticated functionalities to the end user.

Let us consider a user Alice, who owns a next generation mobile phone on which she has installed the Smart Media Player application. This application streams music and video for free from other devices, mocking the functionalities of radio and TV channels; advertisements are injected from time to time. Both content and adverts are selected based on what is currently available in the environment, taking into consideration Alice's profile. This simple scenario describes a variety of mobile services and introduces a variety of compositions semantics. For example, the media content selector and the advertising service both require to collect Alice's profile and context first; as such they need to be composed *sequentially* (in sequence) to an eventual context-aware user profiling service. Depending on the actual context and user preferences, advertising may be shown or played, either *in parallel* to the content selection and reproduction service, or *subsequent* to it. In order to enable the selection of one or the other strategy, a *choice* composition semantics will be needed. The Smart Media Player updates the list of the next-to-come songs or videos at a regular basis, so that the overall composition *loop* is (re)started.

As this scenario shows, pervasive services are often compound, provided by aggregating more basic functionalities according to a variety of semantics. Some of these services will be local to the client's device, while others will be available from a combination of stationary and mobile providers. In order to enable mobile users to seamlessly and successfully consume compound services, they must be given the impression of being interacting with a monolithic, local service. We propose to do so by means of MoSCA, a run-time service composition framework that takes care of finding and composing those services needed by the composition.

2. MOSCA FRAMEWORK

MoSCA (Figure 1) consists of four main modules: the *Service Manager*, the *Service Analyser*, the *Service Discoverer* and the *Service Coordinator*, that exchange objects of two types: the *Service*, used as a service descriptor, and the *Token*, storing information related to an ongoing request.

The *Service Manager* component is the access point to the composition framework. It provides an interface to request services, abstracting all the complexity of dealing with composite services in mobile environment. Upon receiving a request for service s, the Service Manager creates a Service

*The author kindly acknowledges support from the MiNEMA ESF Scientific Programme.

Middleware'08 Companion, December 1-5, 2008, Leuven, Belgium.

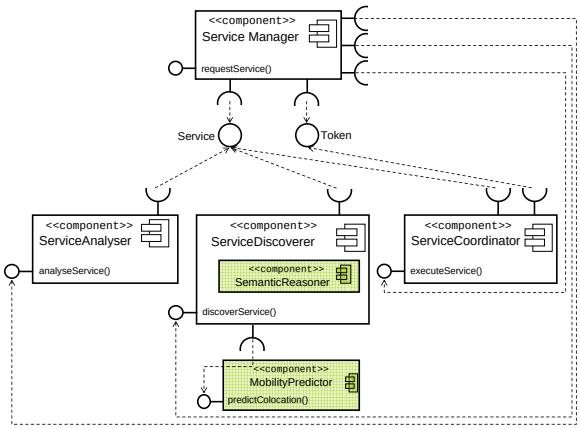

Figure 1: MoSCA Core Components

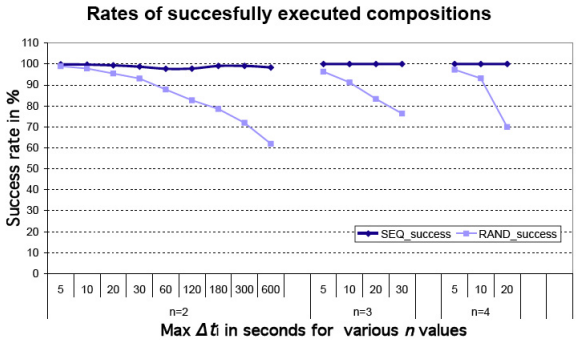

Figure 2: Successful Composition Rates

object and passes it to the *Service Analyser*, whose goal is to 'understand' the request: the Service Analyser decomposes s into component services $s_1, s_2 \ldots, s_n$, and returns the same Service object back to the Service manager, now enriched with the service (de)composition semantics (e.g., $s = s_1$ `seq` s_2 `seq` $\ldots$ `seq` s_n). The Service Manager then passes the annotated Service object to the *Service Discoverer* component; this component selects the set of providers $p_1, \ldots, p_m$, among those available in the current environment, that will be able to *collectively* deliver service s and that will maximise the chances of successful service completion. Once a binding with these providers has been formed, the Service Manager creates a Token object, storing the service request input parameters, and passes it, together with the annotated Service object, to the *Service Coordinator* component which is in charge of executing the request.

In order to give users a truly pervasive experience, MoSCA executes only those compositions that have a high chance of completing successfully. To do so, MoSCA Service Discoverer relies on two key components: the *Mobility Predictor* and the *Semantic Reasoner*. The former estimates for how long a given provider will remain colocated with the client's device, based on historical co-location patterns; the latter then uses these predictions, together with the specific composition semantics, to determine if a composition can be attempted (even before considering other QoS parameters [2]). The basic observation underpinning the *Mobility Predictor* component is that people show a high degree of regularity in their activities. Based on this observation, we have defined a simple yet effective prediction mechanism that aims at learning human behavioural patterns from past activities: for every day of the week d, and for every hour h within a day, a device i logs the duration of its encounters with any other device j. We use the symbol $\delta_{i,j}(d,h)$ to refer to the historical colocations between devices i and j in the specific time slot (d,h). To reduce overhead, only records about *familiar strangers*, that is, hosts we have been encountering with at least a certain frequency, are kept. Given two service instances i and j, and the current time t which falls in slot (d,h), the predicted duration of colocation is then computed as follows:

$$\sigma_{i,j}(t) = \begin{cases} 0 \\ \text{if } \mathrm{avg}(\delta_{i,j}(d,h)) \leq \alpha \cdot \mathrm{stddev}(\delta_{i,j}(d,h)) \\ \\ \mathrm{avg}(\delta_{i,j}(d,h)) - \alpha \cdot \mathrm{stddev}\ (\delta_{i,j}(d,h)) \\ \text{otherwise} \end{cases}$$

The basic idea behind the *Semantic reasoner* is that not all services are indeed needed for the whole duration of the composition. For example, if two services s_1 and s_2 are sequentially composed, s_1 will only be needed initially, and not for the whole duration s_1`seq`s_2. Based on the composition semantics, the Semantic Reasoner computes, for each component service s_i, the amount of time Δt_i that s_i is requested to be available; it then proceeds with the composition only if all providers p_i required by the composite service s have a remaining colocation time greater than their minimum colocation requirement Δt_i.

Does it work? In Figure 2 we report the percentage of the successfully executed compositions, among those initiated, when using MoSCA as opposed to a random service provider selection technique. The experiment focused on sequential composition, while varying Δt_i and number of component services n. The experiment was conducted using the MIT Reality Mining data set of human movement. As shown, MoSCA successfully executes more than 97.5% of the initiated compositions, with pick differences of 35% with respect to random compositions. The percentage of compositions mistakingly not started is often lower then 1% and never higher than 8%.

Is it easy to use? The programming effort associated to using MoSCA is very small. To invoke a composite service, just three steps are required: (i) creation of a *CompositeService* s; (ii) creation of a *Token* object *token* to be passed to the actual input request, and (iii) invocation of the *ServiceManger* to execute service s as described by *token*.

```
CompositeService s = new CompositeService(taxonomyID,
                                          serviceID);
Token token = new Token(''postcode'', ''WC1E 6BT'');
ServiceManager.requestService(s, token);
```

3. STATUS & FUTURE WORK

Our goal is to provide a framework that runs seamlessly on mobile devices and that enables the interaction with composite services as if they were delivered by a unique provider, reachable and available for the duration of the service. To do so, it is necessary, not only to react to changes in the environment, but more importantly to predict them. So far

we have studied historical colocation information to make reliable predictions [1]. Selecting providers based on their predicted colocation will not protect users against malicious ones. We are now studying the integration of distributed trust models within MoSCA, to dynamically reason about the trustworthiness of a composition.

4. REFERENCES

[1] L. D. Prete and L. Capra. Reliable Discovery and Selection of Composite Services in Mobile Environments. In *Proc. of 12th IEEE EDOC*, 2008.

[2] J. Liu and V. Issarny. QoS-Aware Service Location in Mobile Ad-Hoc Networks. In *Proc. of IEEE MDM Conference*, 2004.