Cooperative Network of Mobile Agents to Remotely Process User Information Requests

Roberto Yus and Eduardo Mena
University of Zaragoza
Maria de Luna 1, 50018 Zaragoza, Spain
Email: {ryus,emena}@unizar.es

***Abstract*—In this paper we introduce a network of static and mobile agents which collaborate to process information requests by mobile users. The agents help users to define their information needs, process their requests by deploying themselves on devices near the information requested, and obtain results from users and their mobile devices, in a continuous way.**

***Keywords*-Software and Pervasive Agents, Information Exchanges in Multi-Agent Systems**

I. Introduction

We live in a world of connected devices (e.g., smartphones and tablets) equipped with multiple sensors able to measure temperature and location, or take pictures and videos, among others. Users might be interested in accessing information that can be provided by these network of devices. In this scenario, the traditional vision of agents that collaborate among themselves to obtain information for their users is very compelling. However, there is a lack of collaboration among these mobile devices that would be beneficial to obtain interesting information for users. Furthermore, software agents might not be enough for some tasks and in some situations might even need to collaborate with humans to achieve their goals (e.g., an agent could activate the camera of a smartphone to take a picture but it still needs the human to point the camera to a monument).

In this paper we present the network of agents designed for the SHERLOCK system [2], our approach to provide mobile users with interesting Location-Based Services (LBSs), which executes on mobile devices and leverages their communication mechanisms to exchange information among them in an ad hoc manner. The (static and mobile) agents process the information request, and deploy themselves through the (wired/wireless) network until reaching the most appropriate location to retrieve the data needed, send them back through the agent network, and answer the user in a continuous way. This approach assume that the highly-dynamic network infrastructure is unknown a priori, therefore mobile agents adapt themselves to such continuous changes, bringing the computation wherever needed and creating new agents when necessary to achieve their goals.

II. Request Processing Using Agents

A. Helping Users Defining their Requests

In SHERLOCK, the *User Request Manager (URM) agent*, residing on the user device, helps the user to specify her information needs and so to find and select the (location) service that she needs, by using a local ontology and a dynamically generated GUI. The result of this process (out of the scope of this work, see [2] for more details) is that the URM generates a formal query expressed in GeoSPARQL-DL (SPARQL extended with location and Description Logics constraints); see an example in Figure 1, for a user request of pictures of the July 4th celebration at Washington D.C. (specifically, pictures of the header of the parade and pictures showing the fireworks and the Lincoln Memorial). The URM, after consulting the ontology, specifies for each request whether other users should be considered or not as information sources, depending on the quality of answer (number of results, number of requirements not fulfilled including the required age of data) obtained from regular sources. For example, for our sample query, humans could be asked in case of obtaining less than 10 photos.

```
SELECT ?photo, ?timestamp
WHERE {
     Type(?photo, sherlock:Photo),
     PropertyValue(?photo, sherlock:target, ?parade),
     Type(?parade, sherlock:ParadeHeader),
     FILTER(geof:within(?photo,
              geof:buffer(?parade, 200, 'm'))),
     ...
} OR WHERE {
     Type(?photo, sherlock:Photo),
     PropertyValue(?photo, sherlock:target,
                    sherlock:Fireworks),
     PropertyValue(?photo, sherlock:target,
                    dbpedia:Lincoln_Memorial)
     FILTER(geof:within(?photo,
              geof:buffer(38.8885, -77.0523, 200, 'm'))),
     ...
}
```

Figure 1. Excerpt of an SPARQL-like sample query.

Afterward, the URM agent creates a *User Request Processor (URP) agent*, which will process the user request, and provides it with the formal query to be processed. URP firstly tries to obtain an answer locally by executing the query on the local ontology on the device. If the local infor-

mation is not enough according to query requirements, then a network of mobile agents is deployed to get information from other devices (see Figure 2). For this task we reuse and improved the approach presented in [1].

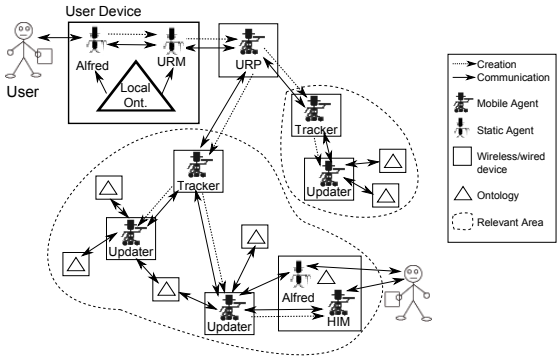

Figure 2. SHERLOCK's agent network.

B. Agent Network Deployment

Creating Trackers: The URP agent analyzes the query constraints to determine the number of agents to create: Each location constraint has associated 1) a geographic *reference* (a static point or a moving object in the scenario), 2) a geographic area related to that reference, and 3) the kind of *target objects* that must fulfill being inside that area. The URP agent creates a *Tracker agent* for each location constraint, to retrieve target objects inside their geographic areas. For the sample query in Figure 1, two Trackers are created, one will follow the header of the parade (the first location constraint reference), and the other will try to find a device around the centroid of the specified area near the fireworks (the second location constraint reference). Each Tracker agent will try to scan its associated geographic area but in parallel, due to its mobile nature, it will jump from moving device to moving device to keep itself as close as possible (physically) to the reference of such a geographic area, even staying on the reference object itself. In the running example, the first Tracker agent will try to move to a device of a member of the parade header to "automatically" follow the header (without moving to another device). This way, as opposed to the approach in [1], SHERLOCK is able to process requests in dynamic networks where the infrastructure is unknown and change anytime.

Creating Updaters: To retrieve information concerning its associated geographic area, each Tracker agent creates a dynamic network of *Updater agents*, which will move wherever needed to cover the whole geographic area associated to its Tracker agent, and obtain answers concerning its (partial) queries. Thus, each Updater will keep itself on the most appropriate device to provide its Tracker with the wanted information. The Tracker correlates information received from Updaters, analyzing communication delays of each one, in order to increase of decrease the number of Updaters to fit to the network status continuously.

Retrieving relevant information: Each Updater agent residing on a remote device executes the (sub)query assigned against the ontology in the SHERLOCK-enabled devices in current communication range with its host. As we said before, under certain circumstances the system can consider humans as information providers and they can be requested to obtain information needed by other users. In that case, first, the Updater agent sends a petition to the static agent in charge of the interaction with the user, *Alfred*, including a deadline attached (the maximum amount of time the Updater is willing to wait for a reply). Then Alfred checks the context of the user (which might show that she is busy) and her preferences regarding requests from others, whenever possible, Alfred will ask the user and communicate the Updater if she accepts; in that case, the Updater will send a *Human Interaction Manager (HIM) mobile agent* to the user device in order to manage such a request and then return the result to its corresponding Updater. In our running example the HIM agent will open the camera application and ask to the most appropriate user to move to the steps that lead up to the Lincoln Memorial and take a picture with the fireworks on the background.

C. Continuous Query Processing

Once Updaters obtain the information from devices around, they send it to their corresponding Trackers. Trackers correlate the information received, considering the most recent data when multiple Updaters retrieve related information, and return it to the URP agent, which performs a correlation of the information obtained from the different Trackers (e.g., in our running example to join the pictures obtained of the parade and the fireworks). The user might want to obtain a continuous flow of information (e.g., as long as the fireworks and the parade are taking place). Therefore, the URP maintains the network of agents as long as the user maintains the request active. In fact, concerning synchronization among all the agents, when the required refreshment rate falls below the requirements of the query, each Tracker can create new Updaters to balance the monitoring tasks. Also the URP could even create more than one Tracker for location constraints specially complex to process (e.g., with a single but huge associated geographic area).

Acknowledgments. Research work supported by the CICYT project TIN2013-46238-C4-4-R and DGA-FSE.

REFERENCES

[1] S. Ilarri, E. Mena, and A. Illarramendi. Location-dependent queries in mobile contexts: Distributed processing using mobile agents. *IEEE Trans. on Mobile Comp.*, 5(8):1029–1043, 2006.

[2] R. Yus, E. Mena, S. Ilarri, and A. Illarramendi. SHERLOCK: Semantic management of location-based services in wireless environments. *Pervasive and Mobile Comp.*, 15:87–99, 2014.