Agent-Based Adaptive Mobile Computing in Games (Demonstration)

Damian Burke, Axel Heßler, Sahin Albayrak
TU Berlin, DAI-Labor
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
{damian.burke,axel.hessler,sahin.albayrak}@dai-labor.de

ABSTRACT

The demonstration is showing a mobile application (2-player game), utilizing context-awareness relying on agents. The agents are gathering and processing context-data retrieved from the device's built-in sensors, which will be distributed within the system. The application (game) adapts to certain changes in the environment (in the demo: connectivity, power supply) to preserve the mobile device's power level as well as decrease network utilization.

The presentation can be found at: http://goo.gl/ZFOnUO

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*

Keywords

Multi-agent system, mobile computing, context-awareness

1. INTRODUCTION

With increasing numbers of mobile devices, mobile computing gained more traction in the last few years. Current flagship devices are sold with quad-core or even octa-core CPUs, high-resolution displays, as well as increased physical size of displays – offering high performance but also high power requirements. With limited device size, the battery size is also restricted, manufacturers have to make a trade-off between the device's physical thickness and the battery's capacity. With mobile devices currently offering resolutions ranging between 720p and 1440p, often times they are used as mobile gaming-center. Especially running graphics-intense 3D games on these devices decreases the device's life-span on a typical battery-charge immensely. To counteract this, developers of games may reduce it's quality – for example lower the quality of textures, reduce the framerate, or simply render the game at a lower resolution. Often times games either offer no possibility to reduce these quality settings, or they are placed within a multi-leveled settings menu which the user would have to access each time these settings are changed. To adapt these settings automatically or in runtime requires context-awareness.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*
Copyright © 2015, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Agent systems have already found some traction in mobile development, for example with JADE for Android[1], but most of the approaches and implementations are focussed on turning the device itself into an agent, working in a multi-device agent system.

2. METHODS

The demonstration presented shows an example of a multi-agent system within a mobile device. Multiple agents are deployed on a mobile operating system, each autonomously capturing and processing certain information or executing tasks. In the demonstration, multiple agents are collecting information on the device's context as well as environment, processing these and passing them on to the application.

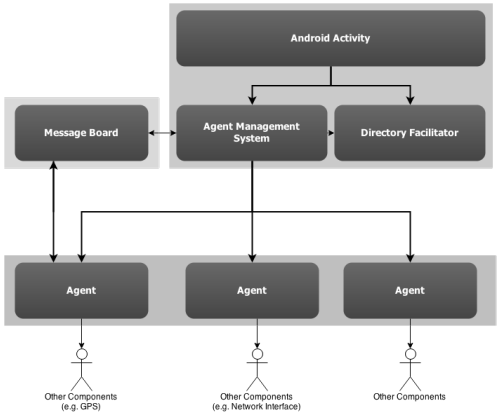

Figure 1: Agent Management Structure

Figure 1 depicts a simplified version of the used agent management structure. In the simplified version, the agent management system is centralized within the application itself.

2.1 Agent Management

Agents are stored within the local agent management system (AMS). In this demonstration, to allow both easy porting and testing, the AMS is located within the application itself and bound to its runtime. In an optimal setting, these agents were to be deployed centralized on the system, gathering and processing information in the background. In this demonstration, mainly two agents are utilized. One agent is monitoring the device's battery – i.e. it's power level as well as state (charging or discharging). This plays a significant

role on mobile devices – determining how much power the application may use and such how much processing time individual tasks may require. The demonstration utilizes these information to reduce the overall refresh rate of displayed content, as well as the calculations – the frames per second (FPS). Reducing the FPS in a game also reduced the user experience since fast movements may stutter – on the other hand it significantly reduces the processing time required and thus the power consumption. The other agent is observing the device's network interface; the network transmission speed and availability is important for many applications, and used in the demonstration to reduce or increase the transmitted information. In all circumstances, information to keep the application's state synchronized and consistent have to be transmitted. Non-essential information may be omitted to reduce the network workload. Combining these agents, it is possible to allow cross-referencing of the agent's states: Low network quality but a charging device might allow for compression of information sent instead of omitting them, to further increase the user experience.

2.2 Communication

An important factor is the communication between different agents – especially to allow agent cross-referencing of states between multiple agents, as well as cooperations. Two different ways of communication are implemented into the agent system; since this is focussing on a demonstration of the system, the exact protocols will be omitted. Agents are able to communicate direct or indirect. Direct communication happens by message-forwarding through the agent management system; indirect communication takes places by storing messages on the message board – these messages might be broadcasted to all other connected agents or directed into specific message queues.

2.3 Adaptation

To utilize the information provided by the agents, applications have to be adaptive. Implementing algorithms able to handle different sets of available information is an important factor, as well as adding control mechanisms into the application to switch between these algorithms. The demonstration adapts to information provided by two deployed agents: Upon receiving a new message, the state is checked – recognizing an important change in the state (e.g. connectivity change or power supply changed) will be treated by changing an internal flag as well as exchanging certain algorithms. Switching from high-quality networks to lower-quality ones will exchange the used position prediction algorithm (using a lower-degree polynomial), this reduces the precision but also requires less data. The removal of a power supply results in the renderer reducing the generated and calculated frames per second, thus requiring less power.

3. RESULTS

The execution of the demonstrated approach was evaluated in two settings. At first, in a simulated environment – i.e. one device with simulated networking and added delays. To gain more detailled information and value on the approach's usability in real-life scenarios, the multiplayer game shown in Figure 2 used as demonstration is also gathering statistical data – focussed on the impact on network utilization and load.

Figure 2: Game implemented with example usage of agent-based context-awareness to determine connection quality and battery / power state of the device.

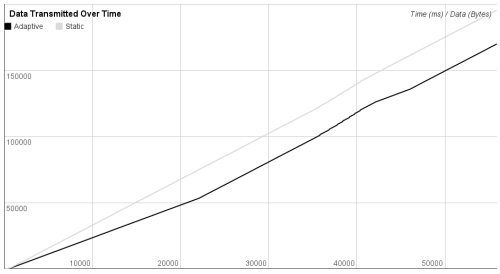

Figure 3: Network Utilization Over Time in executed sample application: The adaptive application sent roughly 13.08% less than the static application.

Figure 3 shows the network traffic produced by the demonstration application over the course of roughly 55 seconds. With a reduction of traffic by roughly 13% while keeping the user experience high as well as different states of applications consistent.

4. DISCUSSION

Realizing adaptive applications with agents has potential to improve the overall workload of mobile operating systems; even though processing power and battery capacities are rising, these resources are still limited and need to be preserved especially in mobile devices, which do not have access to external power sources or high-bandwidth connections at all times. Mobile devices often times have differ in the available hardware components, especially the sensors – smartphones may or may not have access to a variety of network interfaces (UMTS, LTE, ...) - having a modularized approach with autonomous components – the agents – allows the system to be applicable on all systems, with limitations to sensors and available context-data applying in run-time; the application utilizing these data thus is able to respond to certain triggers or information sets, depending on their availability.

REFERENCES

[1] M. Ughetti, T. Trucco, and D. Gotta. Development of agent-based, peer-to-peer mobile applications on android with jade. In *Mobile Ubiquitous Computing, Systems, Services and Technologies, 2008. UBICOMM'08. The Second International Conference on*, pages 287–294. IEEE, 2008.