Sistemas de operación III (CI-4822)
Septiembre-Diciembre 2018
Proyecto Computación Voluntaria y Sistemas Multi-Agentes

El primer proyecto de esta materia está relacionado con los temas de Computación Ubicua y Sistemas Multi-Agentes. Uno de los enfoques más usados recientemente es la Computación Voluntaria (también llamada Computación *Peer2Peer o P2P*), cuya idea es poder compartir poder de cómputo ocioso y capacidad de almacenamiento entre los *peers* que integran el sistema, en un ambiente totalmente distribuido. El enfoque de sistemas multi-agentes (*MAS – Multi-Agents System*) ofrece diversas herramientas que apoyan el desarrollo de aplicaciones *P2P.*

Para este primer proyecto cada equipo deberá:

1. Realizar una revisión de las herramientas y *frameworks* para el desarrollo de *MAS* más usados en la actualidad y hacer una lista que indique la comunidad que lo soporta, qué lenguajes y sistemas de operación soporta, proyectos importantes que lo usen y la filosofía general que implementan (agentes, objetos distribuidos, *containers*, etc.).
2. Seleccionar una herramienta de desarrollo, instalarla y ejecutar correctamente un ejemplo muy básico.
3. Desarrollar un sistema sencillo en el que los *peers* puedan compartir archivos multimedia y poder de cómputo. Los archivos multimedia tienen un *peer* dueño (quien los publica por primera vez) y los demás *peers* podrán acceder a dichos archivos ya sea desde el dueño, desde cualquier otro *peer* que tenga una copia o simultáneamente de varios *peers*. Esto implica que cuando se registra un archivo para compartir o se hace una copia, un **Agente Directorio** debe estar al tanto. Cuando un *peer* solicita un archivo, ya sea por nombre exacto, por parte del nombre, por tipo de archivo (fotos, vídeos, pdf, la clasificación la determinan Uds.), el **Directorio** debe mostrarle al *peer* solicitante las copias disponibles en ese momento y si es posible, recomendando la mejor (porque está más cerca, porque tiene mejor conexión, etc.) . La transferencia debería realizarse de *peer* a *peer* (sin que pase por el Directorio). Para compartir poder de cómputo, un *peer* solicitará este recurso y el **Directorio** le mostrará los *peers* cuyo porcentaje de CPU esté por debajo de 40% (esto implica que eventualmente los *peers* que ofrecen poder de cómputo informan al **Directorio** su estado o el **Directorio** pregunta a todos los *peers* el estado cuando crea conveniente). El *peer* solicitante enviará un *script* de compilación y ejecución de su programa y el código fuente. El *peer* receptor, ejecutará el *script* y cuando finalice publicará en el **Directorio** un archivo con el resultado.

Los puntos 1 y 2 deberán entregarlos en el primer punto de control. El punto 3 deben entregarlo completo para el segundo punto de control.

Para el punto 1, aquí tienen información útil:

- SPADE: Smart Python Multi-Agent Development Environment
  SPADE [1][2] is a FIPA-compliant multi-agent platform written in the Python programming language. The platform offers a library that contains tools and classes to manage and to create new agents. The main communication protocol used by an agent to communicate inside and outside of the AP is the Jabber technology [3]. Each thread is the base for the creation of an abstract agent that is later used to create regular agents or management agents (AMS,DF). Each agent is composed of: a connection mechanism to the platform, a message dispatcher and a set of tasks [4]. The connection mechanism is the agent's communication interface with other agents. The message dispatcher redirects the messages to the correct destination, for example,

when a message is received, it is redirected to the correct task's message queue. On the other hand, if the agent needs to send a message, the message dispatcher redirects it to the correct receiver agent. Finally, the agent's behaviour is implemented as a set of tasks that can be run simultaneously.

- JADE: Java Agent Development Framework
  JADE is a Java-written and FIPA-compliant multi-agent platform and provides a set of APIs and a graphical user interface to develop MAS-based applications [5][6]. A JADE Agent Platform is built up of one or more containers that can be distributed over a network (i.e. the Agent Platform can be hosted by several devices). Each agent lives in a container, which is a Java process that provides the services required for an agent's life cycle. Each JADE platform should have a mandatory main container, which is the first container to be launched when the application is executed. Other agents can live in the main container, but the AMS, DF and ACC components shall run exclusively in this container. The intra-platform communication is managed by each container's message dispatcher, whereas the inter-platform communication is managed by the ACC by using the IIOP protocol. Each JADE agent is mapped into a single thread and it can execute several behaviours. Each behaviour is implemented as a task and it is scheduled according to a cooperative non-preemptive scheduling policy. JADE uses the concurrency model agent-per-thread instead of behaviour-per-thread in order to reduce the number of threads running in the agent platform. This model becomes important in particular for resources-constrained environments. Furthermore, the agent-per-thread model enables improved performance since a behaviour switching is faster than Java thread switching. Also, it eliminates synchronization issues such behaviours accessing the same resource. However, when a behaviour is blocked, the rest of the agent's behaviours are also blocked. If this is not a desired feature in the application, JADE provides a solution to implement behaviour-per-thread, i.e., behaviours that are executed in their own thread, therefore, if a behaviour is blocked, the rest of the agent's behaviours will not be blocked.

- Mobile-C: Multi-Agent Platform for Mobile C/C++
  Agents Mobile-C [7] is a MAS FIPA-compliant platform written in C/C++. This platform is specifically developed to support mobile agents that are software components capable to move between different execution environments. The implementation of this platform and its functionality are explained by its developers in [8] and [9]. The library is supported by Windows, Linux, Solaris, Mac OS X, QNX, HP-UX and Linux OS running on general-purpose or tiny and single-board computers. The major building block of a Mobile-C system is an agency whose core is the agent platform that contains the minimal software components to support the execution of a stationary or a mobile agent. Additionally to the FIPA-specified components, Mobile-C implements two extra components: the Agent Security Manager (ASM) that supervises security policies for the platform and the Agent Execution Engine (AEE) that serves as the interface with the mobile agents. The AP is a multi-threaded program that resides in a host computer running any of the abovementioned operating systems. Each component resides in its own thread and each agent (stationary or mobile) also has its own thread. An AEE thread is started per each mobile agent.

- BESA-ME Framework
  Flores et al. [10] present a software middleware in order to facilitate the construction of robotic control systems based on MAS techniques. The framework named BESA-ME (Behavior-Oriented, Event-Driven And Social-Based Agent Micro Edition) supports the development of

MAS-oriented applications for microcontrollers. Functional tests were run successfully in PIC18F8720 and PIC18F8620 running FreeRTOS. A BESA-ME agent is modeled by at least three components: the channel, one or more behaviours and the agent state. The Channel and the Behaviour components are implemented as RTOS tasks. The Channel component manages a message queue that blocks the task until an event is received. Once an event is received, the task assigns it to a respective treatment function and redirects the message to the corresponding behaviour. The behaviour is unblocked by the message and performs the action defined by the user. Also, the behavior is interconnected with the agent's state where it stores information about the agent, its environment and the global system. Given that two BESA-ME components are modeled as tasks, each agent can have two or more tasks, therefore, the agent is mapped to at least two threads.

- EmSBoT
  EmSBot (Embedded modular Software framework for networked robotic systems) [11][12] is a lightweight embedded C-written framework that provides an API to develop networked robotic systems. An EmsBoT system is composed by one or more independent nodes across the network. Each node contains processing power, memory, at least one communication channel and it may also contain sensors and actuators. The EmsBoT core layer provides the API for agents creating and message routing. The agents only interact between them by using messages. The message structure is fixed since it is beneficial for the determinism of the system by preventing dynamic memory allocation. Furthermore, an EmsBoT agent may have one or more threads: one mandatory thread and optional worker threads. An implementation is available for the ARM Cortex-A8 and the Cortex-M4 processors running µC/OS-III. The memory usage of the firmware is 13 KB of flash memory and 5KB of data memory when there are two agents. One extra agent will cost 1KB.

- ObjectAgent
  The ObjectAgent (OA) project is an agent-based software architecture for autonomous distributed systems that was developed and implemented in C++ by Princeton Satellite Systems [13]. A demonstration scenario was implemented by using a PowerPC 750 processor and Enea's OSE real-time operating system. The platform provides classes for agent creation, skills (behaviour) manipulation and agent communication. The communication and the synchronization among the agents is done by message passing. Each agent has one or more skills that are basic functions that trigger one or more actions that produce a message to be sent across the platform. Several skills can be grouped into a activities where each of them always runs in a separate thread. An agent can carry one or more activities, therefore, an agent is mapped into one or more threads.

Referencias:

[1] Gustavo Aranda and Javier Palanca. SPADE User's Manual (http://www.javierpalanca.com/spade/manual/).
[2] Miguel Escriva Javier Palanca, Gustavo Aranda. SPADE API Documentation (https://spade-mas.readthedocs.io/en/latest/).
[3] Matthew Wild Jerry Pasker Jonathan Siegle Edwin Mons Peter Saint-Andre, Kevin Smith and Jeremie Miller. jabber.org - the original XMPP instant messaging service.
[4] Miguel Escriv´a Gregori, Javier Palanca C ámara, and Gustavo Aranda Bada. A jabber-based multi-agent system platform. In Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS '06, pages 1282–1284, New York, NY, USA, 2006. ACM.
[5] Telecom Italia Lab. Jade Site Java Agent DEvelopment Framework (http://jade.tilab.com/).

[6] F. Bellifemine, G. Caire, A. Poggi, and G. Rimassa. JADE: A White Paper. EXP in search of innovation, 3(3):6–19, 2003 (http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf).
[7] Integration Engineering Laboratory. Mobile-C (http://www.dztall.com/CCR1/phpBB3/index.php)
[8] Yu-Cheng Chou, David Ko, and Harry H. Cheng. An embeddable mobile agent platform supporting runtime code mobility, interaction and coordination of mobile agents and host systems. Information and Software Technology , 52(2):185–196, February 2010.
[9] Bo Chen, Harry H. Cheng, and Joe Palen. Mobile-C: a mobile agent platform for mobile C/C++ agents. Software: Practice and Experience, 36(15):1711–1733, December 2006 (http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.4008&rep=rep1&type=pdf).
[10] David M. Flrez, Guillermo A. Rodrguez, Juan M. Ortiz, and Enrique González. Besa-me: Framework for robotic multiagent system design. In Proceedings of the 3rd International Workshop on Multi-Agent Robotic Systems - Volume 1: MARS, (ICINCO 2007), pages 64–73. INSTICC, ScitePress, 2007.
[11] L. Peng, F. Guan, L. Perneel, H. Fayyad-Kazan, and M. Timmerman. EmSBoT: A lightweight modular software framework for networked robotic systems. In 2016 3rd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA), pages 216–221, July 2016.
[12] Long Peng, Fei Guan, Luc Perneel, and Martin Timmerman. EmSBot: A modular framework supporting the development of swarm robotics applications. International Journal of Advanced Robotic Systems, 13(6):1729881416663662, 2016.
[13] D. M. Surka, M. C. Brito, and C. G. Harvey. The real-time ObjectAgent software architecture for distributed satellite systems. In 2001 IEEE Aerospace Conference Proceedings (Cat. No.01TH8542), volume 6, pages, 2731–2741 vol.6, 2001.