UNIVERSIDAD NACIONAL DE COLOMBIA

Construcción un sub-sistema de software que permita generar movilidad de agentes artificiales de acuerdo con el rol que ellos desempeñan sobre el Sistema TLÖN cuyos recursos se comparten a través de una red Ad hoc.

Germán Darío Álvarez Rodríguez

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Sistemas e Industrial
Bogotá, Colombia
2017

Construcción un sub-sistema de software que permita generar movilidad de agentes artificiales de acuerdo con el rol que ellos desempeñan sobre el Sistema TLÖN cuyos recursos se comparten a través de una red Ad hoc.

Germán Darío Álvarez Rodríguez

Tesis o trabajo de grado presentado como requisito parcial para optar al título de:
MAGISTER EN INGENIERIA - TELECOMUNICACIONES

Director:
Ph.D., Jorge Eduardo Ortiz Triviño

Línea de Investigación:
Computación Aplicada y Sistemas Inteligentes
Grupo de Investigación:
TLON

Universidad Nacional de Colombia
Facultad de Ingeniería, Departamento de Sistemas e Industrial
Bogotá, Colombia
2017

Si descartas lo imposible, inmutable e intangible. Lo que queda es la verdad, por improbable que parezca - Sir Arthur Conan Doyle

Para todas aquellas personas que me acompañaron y me apoyaron, que entendieron que todo aquello que se hace en busca de la verdad, siempre estará más allá del bien y del mal... Que algunos sólo queremos respuestas y vemos nuestra vida como un camino para obtenerlas.

Para mis padres, mi hermana y mi novia.

Resumen

Durante el desarrollo del trabajo se presentará el diseño, construcción e implementación de un sub-sistema de movilidad que permitirá generar la movilidad en agentes de software los cuales son desplegados sobre un conjunto de nodos conectados entre sí mediante una red Ad-Hoc.

El sub-sistema de movilidad que acompaña la creación de los agentes permitirá que los mismos puedan moverse sobre un ambiente virtual y sobre una red Ad-Hoc, es decir, de nodo a nodo.
El objetivo de este trabajo es tomar ventaja de la movilidad y consigo mitigar las limitaciones en recursos que poseen las redes Ad-Hoc. En efecto la flexibilidad que permite una programación de agentes móviles permitirá la continuidad de tareas y procesos para los cuales estos son creados, puesto que las redes Ad-Hoc son caracterizadas por su dinamismos y arquitectura cambiante, sin lugar a duda el permitir que las tareas puedan ser iniciadas y terminadas en nodos diferentes es de gran utilidad para soportar la ejecución de los sistemas.

Palabras clave: Agente Móvil, Ad-Hoc, Sistemas Distribuidos, Agentes Inteligentes.

Abstract

During development of this work we'll introduce the design, construction and implementation of a software Mobile Agent on a multi-agent system in which will be deployed across a set of connected nodes by a MANET (Mobile Ad-Hoc Network.

The agent will provide a discovery and location service for processing and autonomy resources such as: CPU, RAM, and battery on participant nodes. Allowing the system to update their own resources information and establish which services can provide to the final user.

The aim of this work is to take advantage of the well known benefits provided by Mobile Agents and bring them to a system whose needs and constraints require it, for this particular case an Ad-Hoc network.

Keywords: Mobile Agent, Ad-Hoc, Distributed Systems, Intelligent Agents

Contenido

Introducción

En general cuando se piensa en comunicar dos dispositivos o servicios, lo primero en lo que se piensa es un esquema cliente-servidor. Y esto se debe a que es el esquema más utilizado y acogido en diferentes sistemas. Sin embargo para estructuras distribuidas aunque este es totalmente funcional, existen alternativas que pueden dar un mejor ajuste a las necesidades de las mismas.

Los agentes móviles representan un nuevo paradigma para el mantenimiento e implementación de sistemas distribuidos [15], siendo estos una porción de código independiente de plataforma con características especificas que permite moverse, actuar y ejecutarse a través de la red[34]. Los agentes no pueden verse como un cuerpo independiente y aislado, todo lo contrario, estos son situados dentro de un ambiente el cual es el encargado de brindar los recursos necesarios para que el mismo pueda ejecutar sus funciones [9]. Un agente puede convivir junto a otros en el mismo nodo u mismo ambiente, y es precisamente esta interacción lo que forma un Sistema multi-agente.

La principal razón y beneficio que otorgan los agentes móviles se refleja en la reducción de carga sobre la red [16], y esto es gracias su características de ejecución asíncrona. Puesto que los mismos son orientados a metas no requieren una conexión permanente u envío constate de información sobre la red, esto permite desplegar un Agente Móvil con una tarea asignada sobre la red y esperar que el mismo la termine. [25].

La siguiente pregunta que surge es que tipo de sistema puede dar uso de los beneficios de un Agente Móvil, y es allí donde se hace una introducción a las redes Ad-Hoc. Las redes Ad-Hoc son una colección de nodos participantes que se conectan entre sí a través de enlaces inalámbricos, al no existir un ente central cada nodo debe cumplir la función de enrutador y nodo al mismo tiempo [25]. Este tipo de redes provee un gran número de beneficios como su rápido despliegue al no necesitar una infraestructura previa, claro esta que con estos beneficios también vienen algunos inconvenientes o retos a superar, entre los cuales se encuentran: Anchos de banda limitados, autonomía de los dispositivos, estabilidad de la red y auto-organización[36].

Dicho esto la programación de Agentes Móviles sobre redes Ad-Hoc se torna promisoria en el objetivo de mitigar los retos que estas presentan y obtener sus beneficios, en el desarrollo de este trabajo se demostrará la creación de un sub-sistema que permitirá generar la movilidad de agentes artificiales de software, su integración con el ambiente de ejecución y como en el mismo estos agentes pueden ejecutar tareas moviéndose a través de la red. Para el desarrollo de la solución se utiliza el lenguaje de programación Python, el cual por su flexibilidad, plataforma independiente, integración y alta compatibilidad con dispositivos embebidos como Raspberry PI es elegido para la construcción del ambiente, el sistema multi-agente y desde luego el sub-sistema que permite la movilidad de los Agentes que lo componen.

Finalmente se pretende que los Agentes que sean creados para el sistema multi-agente posean la capacidad de movilidad, vista esta como el conjunto de protocolos, arquitectura, puertos , servicios e interacciones que permitirá a los mismos realizar sus acciones en diferentes espacios y migrarse a estos a través de la red.

CAPÍTULO 1

ANTECEDENTES Y JUSTIFICACION

Generalmente los sistemas de red implican un software en el cliente y del lado del servidor, los cuales son dependientes entre sí para un esquema completo de funcionamiento. Por esta razón la programación de agentes móviles se torna cada vez más atractiva para la implementación y mantenimiento de sistemas distribuidos, puesto que brinda una alternativa diferente al esquema mencionado inicialmente de cliente-servidor.[15]

Durante el desarrollo del documento se describirá detalladamente como los Agentes Móviles han surgido como una opción promisoria para la construcción de aplicaciones sobre sistemas distribuidos y sistemas con baja capacidad de recursos, sus ventajas y como los mismos pueden coordinarse formando sistemas complejos y eficientes para la resolución de tareas y cumplimiento de metas. El concepto de Agente Móvil será primordial ya que el sub-sistema que se pretende desarrollar será la esencia de estos Agentes, otro aspecto relevante dentro del proyecto es la forma en la que se obtienen los recursos para que estos Agentes desempeñen sus tareas los cuales serán mediante dispositivos comunicados a través de redes Ad-Hoc.

Cada aspecto del proyecto será descrito y especificado, tal como su función e importancia dentro del mismo. En las siguientes secciones se dará inicio a la definición del lenguaje de programación, Agente Móvil, sus componentes y características con el objetivo de clarificar como la movilidad es un factor de influencia sobre este tipo de Agentes.

1.1. Python: Las raices del sub-sistema de movilidad

Actualmente existen dos versiones de Python: Python 2 y su más reciente versión Python 3, con el pasar del tiempo nuevas tecnologías y nuevos ideas emergen, y sobre los lenguajes de programación no existe esta excepción. Normalmente esta mejora continua se hace de forma incremental y es difícil de notar, sin embargo Python ha tenido un gran avance entre su versión base Python 2 y su más reciente Python 3, desde luego esto puede generar que

algunos programas escritos en Python 3 requieran de cambios para ser ejecutados en Python 2, sin embargo el proyecto aquí presente tiene como objetivo la versión de Python 3, no sólo por ser la más reciente y la que representa el futuro de Python sino por una serie de ventajas y posibilidades futuras que la hacen que la misma sea la más adecuada. [23]
////Meter aca el agente primero.

CAPÍTULO 2

Sistema Multi-Agente (Multi-Agent System - MAS)

Un Sistema Multi-Agente (MAS) por sus siglas en inglés, es un conjunto de agentes individuales capaces de comunicarse, cooperar e interactuar entre sí principalmente a través del envío de mensajes, y es precisamente esta habilidad lo que permite a estos ser auto-organizados, siendo este un enfoque común cuando son desplegados sobre sistemas dinámicos en el tiempo. Para formar el Sistema Multi-Agente es necesario clarificar el papel que juega el Agente (como ente individual), el Ambiente , la comunicación entre los mismos y finalmente como la coordinación de estos elementos da vida a un Sistema completo.

En el siguiente capitulo será descrito el concepto de Agente Móvil, sus principios, su ambiente y sus características principales en cuanto a fortalezas y debilidades en mayor detalle, entre los mismos se mencionarán los modelos de comportamiento para un Agente que consisten en una secuencia de percepción, razonamiento, procesamiento y realización de acciones discretas.[5]Por ahora sólo se usará una visión de lo que es un Agente y Ambiente con el fin de lograr posicionarlos dentro del sistema multi-agente que los contiene.

Existe diferentes tipos de agentes pero inicialmente los agentes desde el momento en que son creados basan sus acciones en percepciones que reciben del ambiente en que existen, y es precisamente este intercambio con el ambiente o con agentes que pertenecen al sistema la que permite la colaboración y procesamiento orientados al cumplimiento de metas. [4]

En la siguiente figura se observa como el agente basado en sus percepciones provenientes del ambiente ejecuta sus acciones, al igual como puede llegar a modificar el estado de su ambiente.

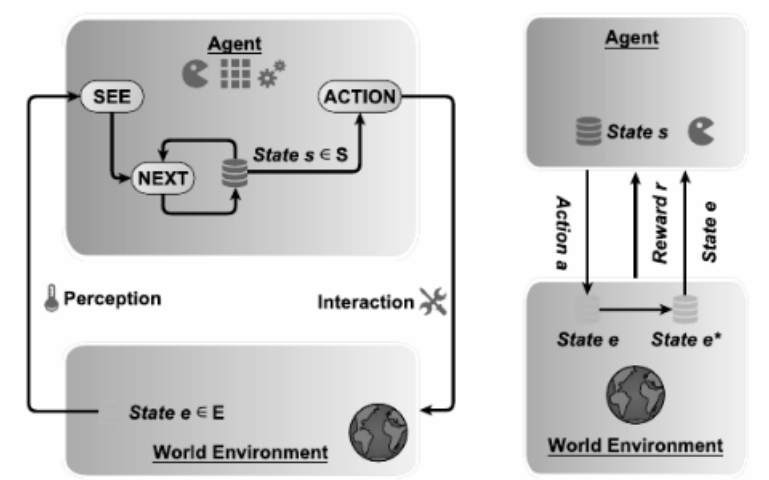

Figura 2-1.: Agente interactuando con su ambiente [4]

Este conjunto de elementos, su interacción y ejecución con un propósito es la muestra de un Sistema Multi-Agente. Ahora que el sistema ha sido definido sólo es necesario determinar los recursos computacionales que soporten el mismo, y es allí donde la redes Ad-Hoc juegan un papel determinante al ser la base de estos recursos que son expuestos al Sistema lo que permitirá al mismos operar y de esta forma a cada Agente cumplir su propia meta.

2.1. Sincronización del Sistema

El tiempo, así como en nuestra vida cotidiana el tiempo es de gran importancia en los sistemas computacionales es aún mayor, y el MAS no es una excepción a este, todos los componentes deben estar sincronizados con el mismo tiempo, ya que en base a este se determina el estado del sistema como tal, si la información que se tiene es aceptable en un intervalo determinado o si por otra parte es necesario desplegar agentes que brinden información actual.

El sistema operativo que soporta el ssub-sistema de movilidad sera sincronizado a través del protocolo NTP (Network Time Protocol), el cual permite que cada nodo participante dentro del sistema distribuido tenga en mismo tiempo de referencia. Este tiempo será usado para determinar la vida de los Agentes y otros factores de medida.

El tiempo de referencia utilizado es POSIX time, UNIX time el cual es definido como el numero de segundos que han pasado desde 00:00:00 Coordinated Universal Time (UTC), Jueves, 1 Enero 1970. [19]. El timestamp será usado al momento de la creación de un agente, en los mensajes de supervivencia enviados por el mismo a su ambiente y para cada información que este obtenga de la ejecución de sus tareas.

2.2. Protocolos de comunicación para la Movilidad

El sub-sistema de movilidad, cuya principal característica es la capacidad de moverse a través de la red, [22] se apoya en protocolos de conexión dependiendo de las acciones han sido establecidas sobre este.

El núcleo de este proyecto es basado en la capacidad del sub-sistema de movilidad en dar uso de los protocolos de Internet para acceder y comunicarse en red. Una de las razones por la que el lenguaje Python ha sido seleccionado para el desarrollo del proyecto es precisamente que el mismo soporta estos protocolos dentro de su librería nativa.

2.2.1. El protocolo TCP/IP

Cuando se construye una aplicación sobre IP, existen varias preguntas a las cuales el diseñador se enfrenta, preguntas como: ¿La aplicación deberá enfocarse en rapidez, agilidad y facilidad, o deberá tomar en cuenta la fiabilidad, trazabilidad y confiabilidad de la comunicación?. A continuación se describirán los tres mayores enfoques al construir una aplicación sobre IP:

- La mayoría de las aplicaciones del hoy día son construidas sobre TCP, el cual ofrece paquetes ordenados y confiables en un flujo de datos sobre aplicaciones de IP.
- Unos pocos protocolos que usualmente usan un método corto de solicitud-respuesta, y clientes simples que se permiten repetir la solicitud en caso de que esta se pierda eligen UDP.
- Y Finalmente protocolos muy especializados evitan estas dos opciones y deciden el crear un nuevo IP-based protocol el cual define una forma totalmente nueva para conversar a través de una red IP.

Para nuestro caso podría decirse que el sub-sistema ha sido basado en las dos primeras opciones, la razón por la cual elegir algunos servicios sobre TCP y otros servicios sobre UDP es precisamente la naturaleza de los mismos, por ejemplo, mientras que en el servicio de envío del código del agente debe ser garantizada la integridad del mismo y de sus datos no es tan importante que sea garantizado lo mismo para la solicitud previa de recepción sobre el nodos destino, es así como esta por su parte debe ser ágil y rápida. En este orden de ideas el sub-sistema utiliza ambos enfoques los cuales son determinados por la naturaleza del servicio o solicitud que es necesario en cada momento.[18]

TCP/IP esta especificado en los documentos llamados Request for Comment (RFCs) los cuales son publicados por la Internet Engineering Task Froce(IETF). Dentro de los RFCs

existen un gran número especificaciones para el protocolo TCP/IP, para un ejemplo especifico IPv4 esta documentado en el RFC 791.

2.2.2. Dirección IP

Las direcciones IP para el caso de IPv4 son un número de 32-bits el cual es normalmente escrito en formato decimal para la facilidad del usuario final, ¿Para que asignar este número a un dispositivo? bien, las direcciones IP básicamente desempeñan dos papeles fundamentales dentro de un red los cuales son mencionados a continuación: [11]

- Estas brindan a cada dispositivo una identificación única a nivel de dirección dentro de la red.
- Estas ayudan a enrutar el tráfico entres las redes.

Acorde a la primera característica, cada dispositivo dentro de la red debe poseer una única dirección y no dos dispositivos pueden compartir la misma dirección en la misma red.

2.2.3. Asignación de direcciones IP

Existen dos formas principales para la asignación de las direcciones IP: **Estática y Dinámica**. La asignación dinámica es configurada por medio el protocolo Dynaic Host Configuration Protocol(DHCP). Sin embargo para el alcance de este proyecto será usado el método de asignación estática, esto con el fin de poder analizar y entender el funcionamiento del módulo de una forma más clara y concisa. [11]

En los siguientes párrafos cuyo objetivo es dar una ligera visión de los servicios por los que un Agente que haga uso del sub-sistema de movilidad deberá reconocer y usar se describe la naturaleza del mismo y el protocolo de comunicación elegido para su desarrollo con base a está misma.

2.2.4. Registro del Agente en el Ambiente

Una vez ejecutado el código del Agente, su primera acción sera la búsqueda del ambiente sobre el cual el mismo debe registrar su creación, por esta razón el ambiente debe existir previamente y estar asequible al Agente, si el servicio de Ambiente cuyo conocimiento es heredado a el Agente al momento de su definición no esta ejecutándose, el Agente no podrá registrarse y por ende no podrá ser creado. Puesto que este proceso es la primera acción inherente a la creación del Agente, el mismo debe ser ágil, rápido y no necesariamente confiable puesto que si falla simplemente el sistema deberá intentar el despliegue del agente en un tiempo posterior , razón cual el protocolo de comunicación implementado es UDP.

El puerto y servicio para usar es heredado a el Agente de la super clase inicial, en esta es definidos los comportamientos, puertos , servicios y características con los que mínimo debe contar cada uno de los Agentes dentro del MAS.

2.2.5. Envío de estado

El estado del Agente representa su memoria, representa la información que este ha obtenido en el transcurso de su creación hasta su destrucción. Por esta razón es importante que en el proceso de migración o dicho de otra forma cuando el Agente se mueva a través de la red sea garantizado la integridad de esta información, de lo contrario el Agente perdería el trabajo realizado y debería empezar su tarea nuevamente para alcanzar el objetivo por el fue creado.

El protocolo para envío del estado ha sido designado como TCP, este agregar tcp info por su característica y orientado a conexión nos brinda una mayor fiabilidad sobre la información que se esta enviando,

2.2.6. Envío de código del Agente

El código del Agente es el cuerpo del mismo, la sentencia de lineas que permiten que este pueda ejecutarse en cada uno de los nodos por los que el mismo viaja, este código es leído y ejecutado por el Interprete de Python al momento de llamar el mismo. Dicho esto si el código estuviese corrupto o dañado,el Interprete no podría ejecutar el mismo y el Agente no podría cargar su estado o iniciar su tarea. Por esta razón una vez mas el protocolo designado para esta acción ha sido TCP.

2.2.7. Servicios y puertos del sub-sistema

Rangos y número de puertos

Cuando vamos a definir los puertos para cada aplicación es importante tener claro los rangos que han sido definidos para ambos protocolos (UPD, TCP), existen tres tipos de rangos dados por la The Internet Assigned Numbers Authority (IANA) los cuales son:

1. "Well-Known Ports"(0-1023) : Estos son reservados para los más importantes y usados protocolos. En muchos sistemas Unix-like los usuarios no pueden usar estos puertos, así se evita conflictos y problemas con los protocolos designados para estos.

2. Registered Ports"(1024-49151): Estos no son usualmente tratados como puertos especiales por los sistemas operativos, siendo así cualquier usuario puede escribir su programa y tomar el puerto 5432 y pretender ser una base de datos PostgreSQL pero estos pueden ser registrados por la IANA para protocolos específicos.

3. Los puertos restantes (49152-65535) están libres para cualquier uso, es así como los sistemas operativos modernos usan este conjunto para generar puertos aleatorios cuando al cliente no le preocupa el puerto que se le es asignado. [18]

CAPÍTULO 3

La Arquitectura de un Agente Móvil

3.1. Un Agente que hereda su esencia

3.1.1. Clases y herencia

Como un lenguaje orientado a objetos, Python soporta una totalidad de características como la herencia,polimorfismo y encapsulamiento. Para el caso de este proyecto el concepto de herencia es fundamental en cuanto se observará en los siguientes párrafos.

Las clases y la herencia permite mejorar y extender la funcionalidad del sub-sistema a través del tiempo, brindando flexibilidad en un ambiente en que los requerimientos pueden cambiar.[26]

Recordemos que en nuestros escenarios esta presente un Agente Móvil, pero ¿Qué es un Agente móvil?, en una definición corta que podría ser aceptada es la de un Agente que posee movilidad,en todo caso nos llevaría a definir por separado lo que es un Agente y lo que la Movilidad. Sin embargo con el fin de no prolongar esta definición y enfocarnos en lo concerniente a este capitulo la abstracción que se hace de un Agente es todo lo que esta en incluido en el módulo Agents.py, es decir, cualquier Agente que nace o "hereda"de las clases que constituyen ese modulo puede considerarse entonces un Agente como tal dentro del contexto de este proyecto.

Es aquí donde se observa como la literatura y la definición conceptual es representada por una definición y utilidad del lenguaje de programación, efectivamente los Agentes son creados a partir de la clase Agente del modulo Agents.py lo cual es afortunado ya que las característica de el Agente base siempre estarán presentes en todos aquellos que aunque puedan tener una función, atributos o formas diferentes siempre en su interior serán Agentes del sistema al conservar su esencia.

3.2. Agente Móvil

Ventajas como un uso efectivo de recursos, reducción del tráfico y carga sobre la red e interacción en tiempo real con su ambiente son características óptimas para las redes Ad-Hoc. [?] Para su desarrollo es necesario contar con una plataforma denominada Mobile Agent Platform (MAP) o ambiente por otros autores la cual es necesaria para que los agentes puedan operar, un ambiente apropiado provee para la ejecución de los agentes, seguridad, persistencia, comunicación entre agentes y mensajería [25]. [10] [37][6], En un típico Sistema Multi-Agente (MAS) puede existir múltiples agencias ejecutándose sobre los diferentes nodos, [22] siendo estás una capa de abstracción distribuida que provee los conceptos y mecanismos para movilidad y comunicación entre los agentes, brindando capacidad de migración de agentes. [17].

En la siguiente figura se observa como el código del agente puede ser migrado y ejecutado en otro nodo apoyándose en la Plataforma o ambiente existente en cada uno de ellos.

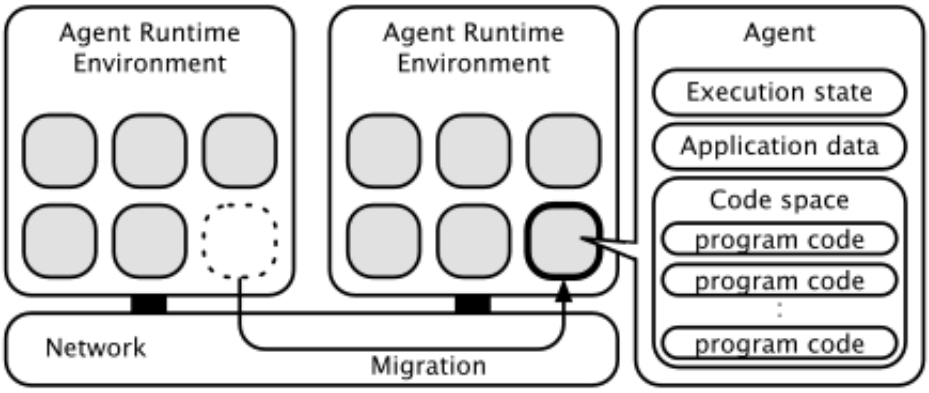

Figura 3-1.: Visión de un sistema de Agentes Móviles.[15]

3.3. Identificador del Agente

El identificador del Agente o como es llamado a través del desarrollo del trabajo AgentId es un numero único asignado a cada agente al momento de su creación, el cual es calculado y registrado dentro del ambiente al momento de creación y proceso de registro de cada Agente perteneciente al sistema.

El AgentId es generado como Universally Unique IDentifier (UUID), los UUIDs son secuencias de (128 bits) en formato de 16 octetos relativamente corto referente a otras alternaitvas.

Una ve las ventajas de UUID es que no requiere un proceso central de registro y puede ser calculado fácilmente por los diferentes lenguajes , entre ellos claro esta, Python.

La versión especifica usada el el proyecto es la versión 4, para números aleatorios o pseudo-aleatorios. Referente a la seguridad no debe asumirse que los UUIDS son difíciles de adivinar por lo cual es importante no usar los mismos en estos procesos [21]. Como el objetivo de el proyecto es el uso de los mismos como identificador único, es una solución apropiada para el sistema multi-agente. A continuación se muestra la forma y resultado para generar un UUID en Python:

```
>>> from uuid import uuid4
>>> uuid4()
UUID('d3bdac13-83ec-4ea4-8ab9-420c0b7763cb')
```

3.4. Estructura de un Agente Móvil

El principio de un Agente Móvil o Agente en general fue definido con base a una Máquina de estados finitos (FSM), las máquinas de estados finitos son usadas en aplicaciones de computación y ciertamente en más cosas a nuestro alrededor de las que imaginamos. Se basa en un concepto simple pero eficaz de mantener estados internos los cuales recibirán entradas que permitirán el cambio de estado y a su vez la generación de una salida, como su nombre lo indica este número de estados deben ser finitos y solo se podrá estar en cada uno de ellos un tiempo a la vez. La definición de Máquina de estados finitos (FSM) según Ralph Grimaldi se enuncia a continuación:

> "Una máquina de estados finitos M se define como una Quíntupla:
>
> $$\mathrm{M} = (\mathrm{S},\Upsilon,\mathrm{O},\nu,\omega)$$
>
> Donde S = el conjunto de los estados internos para M, Υ = el alfabeto de entrada para M, O el alfabeto de salida para M, ν es el siguiente estado de la función, y finalmente ω = la función de salida. - Ralph Grimaldi [13]

Este principio será la esencia de cada Agente, así este definirá su comportamiento basado en las percepciones del mismo. Sin embargo además de esto los Agentes están constituidos de: un estado (estado de ejecución del agente), código (instrucción a ejecutar) y data (Información de las variables del agente) [10] también es identificado por características tales como: Inteligencia, comunicación, autonomía, sociabilidad y movilidad, las cuales son detalladas a continuación: [9] [3]

3.4.1. Sociabilidad

Así como en las sociedades humanas, la comunicación es un factor que habilita la cooperación y recolección de información sobre el ambiente más allá que los intereses locales o individuales, un agente cooperativo y con habilidad social puede obtener información tanto de sus sensores individuales como de los demás agentes y el mismo ambiente sobre el que este se ejecuta[9]. En efecto un Agente que reconoce de su entorno y siente los Agentes en en el mismo, es un Agente comunicativo que puede valerse de eso para beneficio de sus objetivos.

3.4.2. Movilidad

Capacidad de migrarse a través de la red para desempeñar tareas específicas siendo esta primordial para el funcionamiento de un Agente Móvil.[22] En este orden de ideas si un agente decide bajo su definición y programación moverse hacia otro nodo, este deberá seguir el siguiente flujo con el fin de mantener sus propiedades y funciones[**?**].

3.4.3. Comunicación

Es la propiedad de intercambiar información con otros agentes dentro de su ambiente de ejecución es denominada comunicación.[3]

3.4.4. Inteligencia

Una de las particularidades y atractivos sobre el paradigma de los agentes móviles es su inteligencia, sea esta misma definida en un contexto como la capacidad de adaptarse y/o cambiar respecto a su ambiente basado en la información que el mismo le provee. [3]

3.4.5. Autonomía

Es la capacidad de los mismos de controlar sus acciones, estrategias y comportamiento sin la necesidad de la interacción humana, [22], Un agente puede interactuar con otros agentes, utilizar los recursos disponibles para completar su tarea y aun así preservar su autonomía. [37]

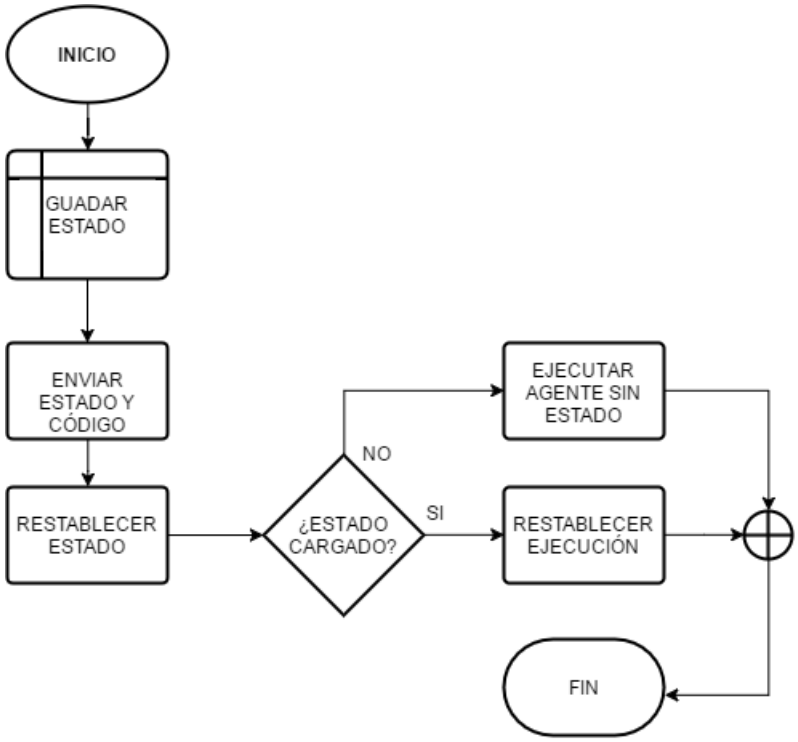

Figura 3-2.: Diagrama de flujo - Agente Móvil construido con base en [**?**]

Cuando un agente se mueve de un nodo a otro lleva consigo la información del nodo origen, de esta forma mantiene la información actualizada y a su vez se actualiza a sí mismo. Con el fin de que esto sea posible la información debe llevar un tiempo de referencia así los nodos sabrán el estado del sistema respecto al tiempo, esto para el caso de los agentes que monitorean recursos o sincronizan información [8]

Los agentes móviles van a través de la red en busca de información, ejecutando una tarea, relacionándose con otros agentes que a su vez se mueven o permanecen dentro del mismo nodo, todo esto es posible ya que múltiples agentes pueden ser ejecutados bajo el mismo nodo.[14][9] Puesto que la característica de movilidad se convierte en la acción principal de todo agente móvil, surgen a su lado factores relevantes como patrón de movilidad, tiempo de ejecución, y destrucción del mismo.[**?**]

Uno de los retos al desplegar agentes móviles sobre ambientes inalámbricos como lo son las MANET es el patrón de migración para estos, ya que el mismo afectará directamente el desempeño de la red. Existen grandes consecuencias al escoger este patrón de migración el cual decidirá el siguiente nodo que el Agente visitará. [8]

Un agente de acuerdo a su patrón de movilidad puede ser dividido principalmente en los siguientes tres tipos:

- Itinerario estático orden estático
- Itinerario estático orden dinámico
- Itinerario dinámico orden dinámico

Sea el itinerario el conjunto de nodos que el agente deberá visitar una vez sea desplegado en la red. [10]

También existe el modelo Free roaming mobile agent (FroMA), donde el agente viaja a través de los nodos de la red sin una ruta definida, recorriendo los mismos de forma aleatoria. [10].Para el caso de sistemas cuyas condiciones de red son impredecibles generando rutas poco confiables este patrón de migración puede ser una buena opción a considerar, incluso algunas veces puede mostrar mejores resultados que patrones considerados más "inteligentes".[8] Sin embargo el siguiente nodo a recorrer puede ser decidido acorde a las características que el ambiente actual de ejecución ofrezca, haciendo uso de las propiedades inherentes del agente, tales como inteligencia y autonomía.

Los Agentes Móviles son implementados ampliamente en aplicaciones como comercio electrónico, manufactura, administración de red, sistemas de control en tiempo real, control de recurso y automatización de entornos [31][22]. Sin embargo, cuando se piensa en un Agente Móvil no puede pensarse en el mismo como un ente aislado, estos no pueden ser vistos como sistemas sin cuerpo, deben ser situados sobre un ambiente y es precisamente junto a este ambiente que los mismos son capaces de percibir, actuar, [9] ser creados, ejecutados y finalmente destruidos si es el caso. Por esta razón durante los siguientes párrafos se definirá un término inherente a todo Agente el cual es: El Ambiente.

CAPÍTULO 4

El Ambiente y su papel en la Movilidad

4.1. Ambiente

Los conceptos de Agente y Ambiente han sido considerados inseparables desde el principio de investigación de Agentes, "Los Agentes no puede ser considerados independientes se su ambiente en el cual existen y a través del cual interactúan". Algunas definiciones apuntan que el ambiente puede considerase como todo aquello fuera de los límites del Agente como tal y su estado cambia por la influencia de todos los agentes que se ejecutan con este, claro está, limitado por las restricciones que el mismo ambiente puede llegar a imponer.

Sin embargo recientemente el Ambiente ha sido considerado como la primera clase de abstracción con sus propias responsabilidades fuera del Agente. Una de las principales confusiones al momento de identificar el ambiente es confundir el .ªmbienteçomo una abstracción lógica relativa a los Agentes y sus modelos con la infraestructura necesaria para el despliegue de Sistemas Multi-Agentes, la confusión entre la abstracción teórica y lógica con la física. [33]

En el contexto de los sistemas de información, una primera clase de abstracción es un pieza de software que provee el mecanismo, la interfaz para que los agentes puedan acceder a recursos y servicios que no están asignados a ellos en sí, regulando la interacción entre la comunicación por parte de los Agentes[33]. Existen servicios que corresponden sólo al Ambiente, por ejemplo, una registro general de todos los Agentes, su identidad, su estado y funciones básicas del mismo serán almacenadas por el Ambiente y brindadas por el Agente. Es así como cada uno se encarga de su labor y se complementan entre sí.

El papel del Ambiente en los Sistemas Multi-Agente ha sido enfocado hacia la comunicación del Agente y como un contenedor del mismo. Sin embargo el Ambiente posee responsabilidades tales como:

- Estructuración: Al ser un espacio común para los Agentes, este podrá definir aspectos de organización como grupos, papeles, ubicaciones entre otros.
- Administrar recursos y servicios: El Ambiente actúa como un contenedor, es una entidad de control para el acceso a los servicios y recursos, en el cual el mismo puede asignar tareas o solicitudes a los Agentes que lo habitan.
- Comunicación: El Ambiente define la forma de comunicación y los mensajes a través de los cuales los Agentes se entienden y comunican unos a otros.
- Regir el Sistema Multi-Agente: El Ambiente puede definir las reglas y leyes sobre las cuales se rige el Sistema Multi-Agente.[**?**]

El ambiente es lo que el Agente conoce lo rige, en el caso computacional este ambiente puede sentirse como un entorno vitual.

4.1.1. Virtualización

En computación frecuentemente se refiere a virtualización como la abstracción de un componente físico en un objeto lógico lo que permite correr simultáneamente varios sistemas operativos bajo el mismo hardware manteniendo cada objeto lógico o máquina virtual aisladas una de otra.[29]

Al permitir abstraer y compartir recursos entre las diferentes máquinas virtuales [35] la virtualización genera una reducción en costos y aprovechamiento de recursos tanto para el usuario final como para el proveedor del los mismos.[29] [7] En efecto la virtualización proveerá esta base computacional a los Agentes en un nivel lógico, lo que permitirá que los mismos desde su perspectiva detecten sólo los recursos que han sido expuestos.

4.1.2. Nubes Móviles

La palabra nube usada recientemente en ambientes como Cloud Computing, Cloud Storage, y Cloud en general es una abstracción para un sistema que consiste en recursos distribuidos interconectados entre sí los cuales a su vez son compartidos sobre la nube para un propósito como la prestación de servicios. [12]

Una nube móvil es un conjunto cooperativo de nodos dinámicos, conectados compartiendo recursos. Donde un nodo puede ser cualquier dispositivo con una interfaz inalámbrica, tales como celulares, tabletas, vehículos y computadores en general.[12]

Dependiendo de la situación la nube móvil puede servir como una plataforma flexible para exponer un conjunto de recursos, y estos recursos son de hecho los recursos de los nodos,

tales como físicos, de conectividad y aplicaciones entre otros. En la siguiente figura se observa como este conjunto de recursos confirman lo que se denomina una Nube Móvil:

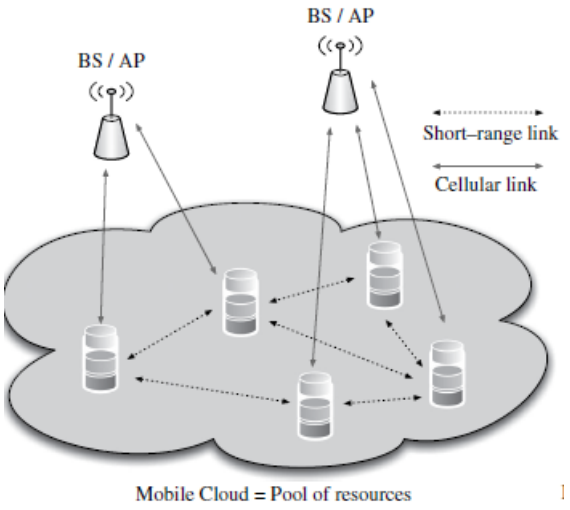

Figura 4-1.: Nube Móvil = Conjunto de recursos [12]

Otro concepto que se menciona es el de sistema distribuido, el cual consiste en el uso de más de un recurso para crear un sistema. [27]

Hoy día la computación de alto desempeño y las súper computadoras son amplia mente usadas en áreas de la ciencia y la Industria. Sin embargo estas computadoras requieren grandes esfuerzos y costos asociados, es por ello que los sistemas distribuidos han sido desplegados en aplicaciones como procesamiento de vídeo entre otras.

Los sistemas distribuidos permiten tener un poder de computación de alto desempeño apoyado en múltiples recursos de menores características. [24] Una de las principales necesidades para la construcción de un sistema de alto desempeño y características de computación es precisamente lograr ejecutar tareas de grandes requisitos que un computador u dispositivo normal no podría satisfacer, pero que desde luego un sistema distribuido de estos sí lo logrará. [30]

Esto genera una introducción a lo que será la base conceptual del sistema de cómputo TLÖN, un sistema distribuido donde sus nodos comparten recursos bajo un modelo definido y cooperativo con el propósito de brindar servicios a través de aplicaciones a los usuarios finales. A continuación se realiza una introducción detallada sobre la arquitectura y modelo del sistema.

CAPÍTULO 5

Movilidad vista como una habilidad del Agente

El propósito general del trabajo aquí presente es generar la movilidad de Agentes, desde luego la movilidad por sí sola aunque un concepto sencillo es fundamentada en una serie de procesos y funciones inherentes a la misma.

A continuación se presentará una descripción del diseño del sub-sistema de Movilidad, como este es integrado por los Agentes y finalmente los requisitos, funciones y características que el mismos necesita para su correcto funcionamiento.

5.1. Guardar estado de ejecución

El poder preservar y restaurar el estado de ejecución para un Agente antes de ser migrado es aquello que permitirá al mismo dar continuidad a sus tareas. Es así que si esto no fuese posible el Agente deberá iniciar desde el principio y perderá el progreso que ha sido alcanzado durante su ejecución en el nodo actual.

Por esta razón el paso inicial para la movilidad parte en una serializción a través de la herramienta del lenguaje Python: **Pickle**

El módulo pickle implementa un fundamental pero poderoso algoritmo para serizalición y de-serialización de un estructura de Objeto en Python. Pickle es el proceso donde un objeto de Python es convertido en flujo de bytes y un-pickle la operación inversa.[1]

El modulo pickle esta diseñado para convertir objetos de Python en una secuencia binaria de bytes, entre los cuales podremos convertir listas, tuplas, funciones y clases definidas por el usuario. Convertir datos a string o bytes para almacenamiento o trasmisión sobre la red es

una operación común en la computación. El nombre genérico que este recibe es serialización Pickle es una forma de representación en Python sin embargo no es la única; Por ejemplo el JavaScript Object Notation (JSON) es otra de estas usada a través de gran cantidad de lenguajes especialmente en la web.[2]

Pickle por si mismo no es una solución de administración o persistencia de datos, solamente convierte los objetos en secuencias de bytes,[20] una de las principales ventajas y por la que se ha decido incluir pickle como herramienta en el desarrollo del proyecto es precisamente el paradigma de la memoria en una Agente. Durante el contexto del proyecto: la memoria y el estado hace parte del Agente y no puede descartarse en la movilidad, todo lo contrario debe ser una de las piezas en consideración. Pickle nos permite resolver este dilema, salvar el estado de ejecución y restaurarlo en otro lugar y tiempo siempre y claro podamos acceder al archivo .pickle que ha sido generado.

Durante el proceso de migración este estado almacenado es creado en un archivo temporal, que representa la información del Agente, sus variables,estructura y lo que caracteriza a este agente como único. Una vez este archivo sea enviado a través de la red a la ubicación destino del Agente será eliminado.

5.2. Envío y serialización

Igualmente el guardar y restaurar son pasos que están divididos por el envío de la información, una vez completado el paso de guardar estado, se da cabida a el envió de este estado al destino. En efecto el envió debe ser realizado a través de la red, por lo que se da uso de protocolos orientados a conexión para garantizar la correcta migración de la información.

5.3. Cargar y restauración de ejecución

Siempre que los pasos anteriores hayan sido exitosos, el cargar el estado y la restauración de la ejecución del agente deberán darse para que pueda darse por exitoso el proceso, el ambiente es de vital importante en la recepción del Agente, es el encargado de ofrecer la plataforma computacional y de servicios necesaria para que los agentes que lleguen puedan continuar la ejecución de sus tareas y así el cumplimiento de sus metas.

5.4. Finalización y destrucción

Un Agente debe tener una finalidad, una meta por cumplir, un propósito por el cual este fue creado. En efecto es muy común que este propósito tenga un inicio y un fin, por esta razón

el Agente debe ser capaz de entender cuando su propósito ha sido culminado, entregar la información que ha obtenido y ser destruido para mantener el ciclo.

CAPÍTULO 6

Funciones de una Movilidad

En principio la movilidad puede verse como un proceso básico de cambio de espacio-tiempo, sin embargo existe un gran número de posibles incógnitas cuando la palabra "moverse.[en]tra en juego.

En Python, las funciones proveen una forma organizada de código que puede ser re-usado. En esencia esto permite usar las funciones pre definidas en el sistema o definir nuestras propias funciones. Las funciones permiten recibir variables, procesarlas y retornar el resultado del procesamiento o acción que se ha requerido.

Las funciones del sub-sistema como lo son la función de itinerario y rol no van muy lejos de la definición de las funciones del lenguaje de programación. Se ha generado una acción la cual obedece a una serie de variables y parámetros propios del Agente que determinarán las forma en que el mismo realiza sus acción en lo referente a movilidad, cabe resaltar que un Agente puede tener una o múltiples acciones y propósitos y la movilidad acá planteada no es más que un medio, no es más que un vehículo que sirve al mismo para finalmente lograr su objetivo o meta inicial para el cual fue creado.[26]

6.1. Roles del Agente

Dentro del sistema es necesario identificar los roles o papeles que poseen los Agentes al momento de su creación. La importancia de los mismos se basa en que son los roles quienes otorgan al Agente de un propósito, este se comportará y tendrá características inherentes a sí dependiendo del rol que desempeñe.

Para este trabajo han sido definidos dos roles iniciales, los cuales darán al Agente un comportamiento particular y su propia forma de usar la movilidad desarrollada. Los roles definidos son:

6.1.1. Collector

Un Agente colector, será encargado de viajar a través de su ambiente y los nodos que lo conforman con el objetivo de recopilar información de cada destino por el cual el mismo se desplace. Este deberá obtener en su punto final toda la información ejecutándose de forma temporal en cada paso dentro de su itinerario previamente planeado.

6.1.2. Worm

Un Agente worm es diferenciado esencialmente al usar un esquema de replica dentro del proceso de movilidad, es decir, el Agente se ejecutará en cada uno los destinos que se encuentran en su itinerario, pero vez de usar una forma temporal el mismo al terminar el proceso de movilidad dejará una copia persistente de su ejecución por cada nodo que ha recorrido.

6.2. Itinerario

El itinerario de un Agente son los puntos definidos por los cuales el mismo debe moverse, la siguiente pregunta es el orden que el mismo deberá recorrerlo. Por ello ha sido determinados dos formas iniciales de recorrer el itinerario las cuales son explicadas a continuació:

6.2.1. Estático

Dentro de las entradas registradas en el itinerario inicial el Agente deberá moverse en el orden estricto al definido, es decir, el primer destino que aparece deberá se el primero en visitar y el último aquel donde terminará su proceso de movilidad.

6.2.2. Aleatorio

Dentro de las entradas registradas en el itinerario inicial el Agente deberá moverse a través de las mismas de forma aleatoria, es decir, no existe prioridad y su decisión del siguiente nodo carece de importancia para el mismo.

6.3. Lectura de Itinerario

Aunque el rol puede ser cambiado como atributo de un Agente durante su ejecución, es el itinerario el cual cambia cada vez que el mismo se desplazada de un lugar a otro. Para lograr resolver tanto la necesidad de leer un itinerario inicial como el de ir actualizado el mismo se ha definido el mismo dentro de un archivo, el cual el Agente escribirá y leerá cuantas veces

el mismo lo requiera.

Una increíble cantidad de información esta disponible en archivos de texto. Información desde trafico, socio-económica , literatura y configuraciones, sin embargo estos archivos también pueden ser usados para representar información o simplemente guardar los resultados de procesos ejecutados. Existen una variedad de partes del sub-sistema que usa los archivos tanto como entrada como salidas, un caso particular de salida es el Módulo Resources el cual deberá captura información del sistema y escribir la misma en un archivo en formato JSON una vez haya sido completada su tarea. A continuación se muestra la forma en la que podemos leer información de un archivo como la lista de nodos los cuales formarán la ruta que deberá seguir la movilidad dentro del sistema en un caso de uso. [23]

6.3.1. Leyendo Información desde un archivo

```
with open('tracer') as file_object:
    contents = file_object.read()
    print(contents)
```

De forma similar es posible actualizar o crear un nuevo archivo, durante el proceso se ejecución el Agente leerá su itinerario y en la movilidad llevará consigo el mismo para una vez termine la misma el Agente dará como visto el nodo que acaba de dejar hasta terminar con toda la lista del itinerario inicial.

CAPÍTULO 7

Los recursos como una meta a través de su Movilidad

7.1. Recursos Computacionales

La forma en que se obtiene información sobre los recursos de almacenamiento, el procesamiento y las configuraciones lógicas en una computadora depende del sistema operativo que esté ejecutando. Por esta razón en el desarrollo de este documento se mostrará la construcción de un módulo de software que permite la extracción de esta información y que tiene el potencial de trabajar en varios sistemas operativos, permitiendo así el acceso a través de una capa de programación creada bajo lenguaje de programación Python.[7]

Finalmente el módulo de software se ejecutará bajo una distribución de Linux que permitirá verificar el funcionamiento del mismo y generar un informe sobre los recursos e información del sistema actual.

7.2. Diseño del módulo de recursos

Los módulos son fundamentales para la mayoría de los ambientes de programación donde esta no te torna de una forma trivial, estos permiten a los programas dividirse en pequeñas partes y de esta forma a su vez permitir la re utilización de código a través de diferentes proyectos. En Python, los módulos son simples archivos cuya terminación es .py y localizados en algún lugar del sistema donde el lenguaje pueda ubicar. Dentro del proyecto esto no es diferente, el mismo sub-sistema ha sido divido en módulos los cuales en su interior desempeñan una función principal; tal como el caso del modulo Transport.py el cual administra y define todo lo referente a la comunicación y transporte del modulo Agents.py que define las propiedades inherentes al Agente.

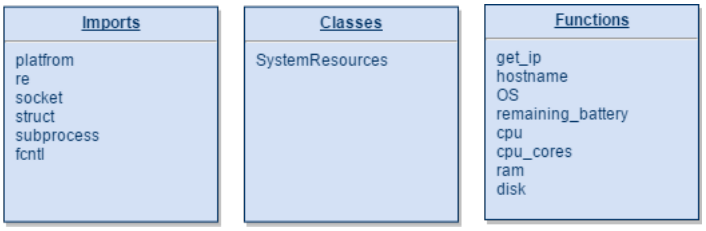

Figura 7-1.: Módulos, clases y funciones para obtener información sobre recursos del sistema

En general podría decirse que el sub-sistema de movilidad es un paquete compuesto de módulos internos que se complementan entre sí brindando la habilidad de moverse y cumplir sus metas a los Agentes que han sido creados para beneficiarse el mismo.[20]

Para el diseño se propone generar una clase denominada SistemResources y dentro de las mismas funciones indicadas utilizando únicamente módulos nativos de Python, de modo que eliminemos posibles dependencias en el momento de ejecución del módulo en otro entorno. El código de Resources.py puede finalmente ser utilizado y migrado a diferentes distribuciones e importado por aplicaciones externas para llamar a sus funciones. La versión de Python3 se utilizará para crear el módulo.

La siguiente figura muestra un diagrama de los módulos, clases y funciones que se encuentran en Resources.py. Algunos de éstos serán explicados e implementados en los siguientes párrafos, sin embargo la mayoría serán probados en trabajos futuros.

7.3. Recursos del sistema y configuración de red

7.3.1. Sistema Operativo

Python ofrece la posibilidad de identificar el sistema operativo en el que se está ejecutando el intérprete, mediante el módulo **platform** que incluye información adicional sobre el sistema, entre otros:

- architecture
- collections
- dist
- machine
- processor

- python version
- system
- uname
- version

La función generada para devolver el tipo de sistema operativo en el que se ejecuta el módulo es muy útil ya que cada sistema proporciona características específicas que pueden generar problemas en el momento de la ejecución de cualquier aplicación desarrollada. La siguiente es la definición de la función OS:

```
@staticmethod
def OS():
    """Return system platform Linux, Windows, MacOS"""
    return platform.system()
```

Todo el módulo cumple con PEP8 de Python y docstrings estarán presentes para explicar el uso de cada función.

7.3.2. RAM

Las utilidades en las diferentes distribuciones de Linux como gratuitas nos permiten determinar la información relevante de la RAM (Random Access Memory) del sistema, sin embargo en aquellas que la aplicación no está presente podemos determinarla por el archivo /proc/-memeinfo, usando Expresiones regulares A través de Python es posible leer y presentar de forma sencilla las características de la memoria como el valor total, utilizado y libre.

7.3.3. IP Address

En el día de hoy donde la conexión y las redes han adquirido gran importancia, cada sistema, PC, smarthphone, debe ser identificado en la red a través de una dirección IP (Internet Protocol), por lo que saber en qué red somos y si tenemos Acceso o salida a Internet son las principales acciones a identificar en un dispositivo con interfaces de red.

La función definida a continuación mira el sistema operativo diferente y devuelve la dirección IP del host que ejecuta la acción basada en los módulos (socket, struct) de Python.

```
@staticmethod
def get_ip(ifname='eth0'):
        """Get the IP number of the main interface"""
```

```
        ip = socket.gethostbyname(socket.gethostname())
    if ip.startswith('127') and platform.system() == 'Linux':
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        ip = socket.inet_ntoa(fcntl.ioctl(
            s.fileno(),
            0x8915,
            struct.pack(b'256s', ifname[:15].encode('utf-8'))
        )[20:24])
    elif ip == None: ip = "Could not get IP Address"
    return ip
```

7.3.4. Software y requerimientos

Dentro de los requerimientos necesarios de software ha sido posible reducir los mismos a los módulos y librerías nativas de Python, es decir, el sub-sistema ha sido diseñado usando los módulos nativos, esto es de gran utilidad ya que puede ser ejecutado sobre cualquier sistema que posea el lenguaje Python en su versión 3 previamente instalado. A continuación se denota una lista de los requisitos de entorno para poder dar uso de todas las funcionalidades del sub-sistema de movilidad acá descrito :

1. Python3
2. Linux OS (recomendado)
3. Interfaz de red
4. Paquete mobile-agent

CAPÍTULO 8

Usando la movilidad de un Agente en una red Ad-Hoc como sistema distribuido

Uno de los escenarios sobre los que se plantea el sub-sistema de movilidad es una red Ad-Hoc creada a partir de dispositivos embebidos que para este caso particular será la Raspberry Pi 3 Model B, la cual en esta versión posee un adaptador inalámbrico integrado. Es así que con un conjunto de tres nodos conectados entre sí por una red Ad-Hoc permitirán evaluar la movilidad de Agentes sobre una implementación real.

Para el contexto de este proyecto ha sido adoptada y trabajada la definición de computación distribuida o sistema distribuido como:

> Çomputación distribuida es el uso simultaneo de uno o más computadores para resolver un problema"[28]

Una aclaración se hace pertinente ya que uno de los objetivos es desplegar el sistema de movilidad sobre un sistema distribuido, donde bajo nuestra definición del mismo los dispositivos o nodos de Raspberry PI son los computadores y el problema a resolver es la comunicación y el permitir la movilidad del Agente sobre la red que estos han creado.

La computación distribuida sobre múltiples computadores es una estrategia cuando se usan sistemas que son capaces de hablarse entre sí a través de la red. La razón principal para construir una sistema distribuido es poder dividir un problema tan grande que un sólo computador no puede manejarlo todo por si sólo, pero al dividir el mismo múltiples computadores pueden comunicarse para resolver cada una de sus partes.

Uno de los ejemplos familiares son las películas animadas de Pixar o DreamWorks, procesar las animaciones de 3D, a 30 cuadros por segundos en una película de dos horas es una gran carga, por lo cual los estudios requieren dividir en granjas de computadoras para que

computadores individuales procesen una sólo parte de la película logrando con ello el objetivo final en un tiempo razonable.

8.1. Programación Asincrónica

Existe un estilo de programación llamado asincrónico o programación no bloqueante, básicamente la diferencia inicial entre el método de programación sincrónica y asíncrona es la utilización de recursos y el bloqueo de los mismos para ejecutar una tarea. Supongamos que existen cuatro tareas a ejecutarse las figuras a continuación muestran las mismas de forma sincrónica y asincrónica.[28]

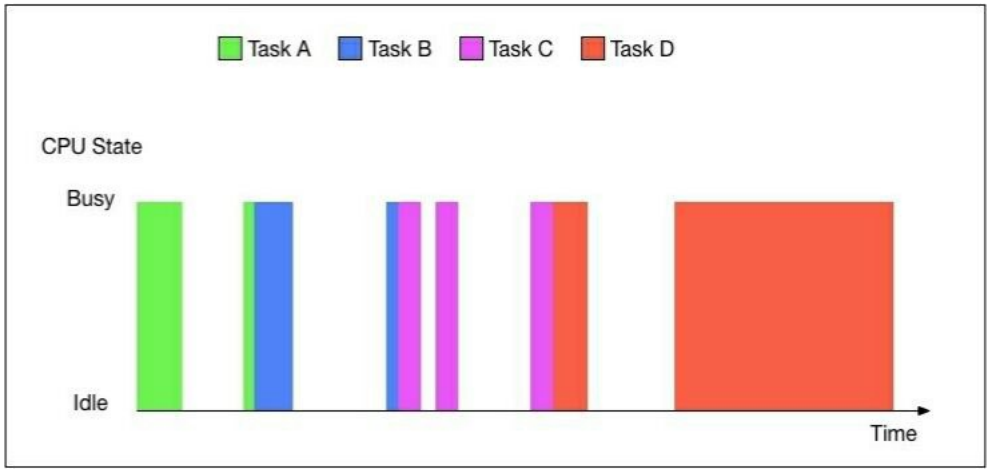

Figura 8-1.: Distribución programación bloqueante

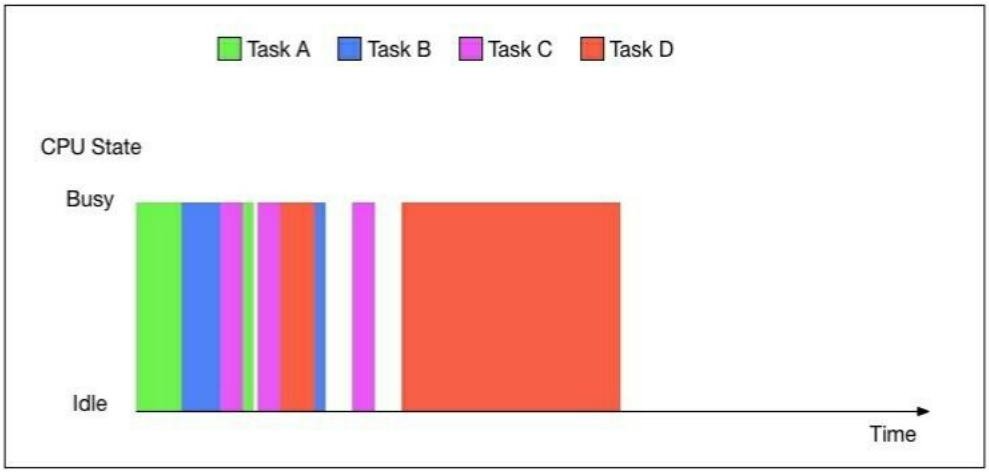

Figura 8-2.: Red Ad-Hoc para movilidad nodo a nodo

La progrmación asincronica no se encuentra dentro del alcance de este proyecto pero ha sido mencioanda pues la versión de Python 3 que se usa para el mismo ya soporta este tipo de programación nativamente, lo que significa que en un trabajo futuro partes del sub-sistema o de todo el sistema puede usar la misma migrando o desarrolando nuevos módulos lo que lleva a una mejora de desempeño considerable.

Mientras que en la forma sinicronica los recursos de CPU por ejemplo son liberados hasta que la tarea es completada , lo que generan mayor uso de tiempo en los mismos, en la segunda figura se observa como en vez de bloquear y reservar la CPU hasta que cada tarea este completa todas ellas VOLUNTARIAMENTE liberan la CPU cuando no la necesitan(porque estan esperando por data).

Aunque aun en la segunda figura se observan intervalos en los que los recursos no son utilizados hay una considerable mejora en tiempo para la ejecución de las tareas. Por ello, es importante tener en la visión de un proyecto como este la posibilidad de una programación asincrónica que aunque no este en el alcance a corto plazo, puede ser planteada como una mejora de desempeño para trabajos futuros.

8.2. Arquitectura de la red Ad-Hoc

En la siguiente figura se muestra el esquema y direccionamiento de red con el que son conectados los dispositivos en mención:

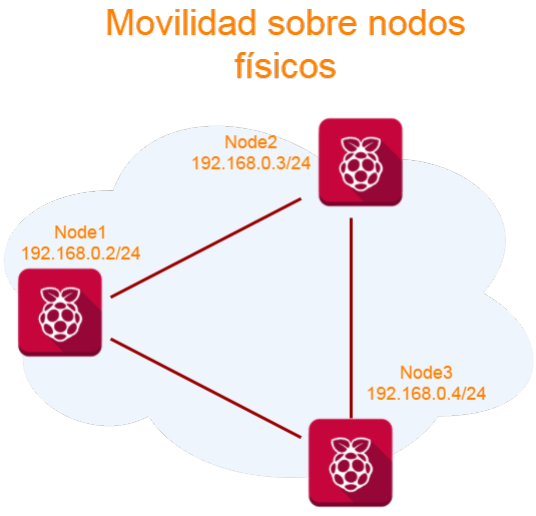

Figura 8-3.: Red Ad-Hoc para movilidad nodo a nodo

Para este escenario ha sido necesario la configuración de tres nodos con el fin de permitir

una movilidad completa y un itinerario mayor que sólo una red con dos nodos. La razón principal de este escenario es observar el funcionamiento de la movilidad en un ambiente físico real, permitiendo la expansión del mimo a otros nodos heterogéneos que cumplan con los requisitos y desde luego a mayor cantidad de los mismos. Sin embargo el despliegue en redes de mayor complejidad será planteada como trabajo futuro del aquí presente.

8.2.1. Sistema Operativo de los nodos

Para estos dispositivos ha sido instalado como sistema operativo Raspbian basado en Debian JESSIE con kernel 4.4, una de las distribuciones recomendadas por el fabricante del dispositivo Raspberry PI.

8.2.2. Agregando el sub-sistema de Movilidad a cada uno de los nodos

Una vez creada la red es necesario añadir a cada uno de los nodos participantes el paquete mobile-agent el cual contiene los módulos necesarios para el funcionamiento tales como Ambiente, Agente, Recursos entre otros. Como ha sido mencionado para que un Agente pueda existir debe estar sobre un ambiente es por esta razón que el primero módulo en ejecución deberá ser el de ambiente. El envio de este sub-sistema se realiza accediendo por SSH (secure shell) directamente a cada uno de los dispositivos.
///cite
Los nodos designados han sido configurados para la prueba de la movilidad, con fines de verificación ha sido creado un Agente (MobileAgent.py) cuya meta será moverse por cada uno de los nodos recolectando información del sistema (CPU, RAM, IP, Almacenamiento) y finalmente generar un reporte con la misma; En la siguiente figura se observa la configuración y estado de cada uno de los nodos:
////// hacer la figura de rasppbian

8.3. Ejecución del Agente Móvil y sub-sistema de Movilidad

Una vez configurado nuestro sistema distribuido por medio de las red Ad-Hoc, El primer paso deberá ser iniciar el ambiente en cada nodo, planear su itinerario y desplegar el Agente para que el mismo pueda cumplir su objetivo. A continuación se observa como cada nodo corre el ambiente y sólo el MobileAgent.py está presente en el nodo número 192.168.0.2(El

primer nodo del itinerario), el itinerario del Agente para este caso será estático y recorrerá los siguientes nodos en orden aleatorio.

8.3.1. Itinerario

1. 192.168.0.2 (Node1)
2. 192.168.0.3 (Node2)
3. 192.168.0.4 (Node3)

El Agente será creado en Node1 y terminará en Node3 con la creación del reporte de recursos que el mismo debe generar. Una vez ejecutado el Agente el reporte en formato JSON generado es el que se muestra a continuación:


```
{
    "node172.17.0.5": {
        "OS": "Linux",
        "host": "4074d48bb7c6",
        "disk": {
            "%use": "9%",
            "total": "1.8T",
            "free": "1.6T",
            "used": "150G"
        },
        "ram": {
            "total": 31783.93359375,
            "free": 3645.625
        },
        "ip": "172.17.0.5"
    },
    "node172.17.0.3": {
        "OS": "Linux",
        "host": "4a9211aae28d",
        "disk": {
            "%use": "9%",
            "total": "1.8T",
            "free": "1.6T",
            "used": "150G"
        },
        "ram": {
            "total": 31783.93359375,
```

```
            "free": 3646.140625
        },
        "ip": "172.17.0.3"
    },
    "node172.17.0.4": {
        "OS": "Linux",
        "host": "377512b44e4c",
        "disk": {
            "%use": "9%",
            "total": "1.8T",
            "free": "1.6T",
            "used": "150G"
        },
        "ram": {
            "total": 31783.93359375,
            "free": 3645.26953125
        },
        "ip": "172.17.0.4"
    }
```


JSON: es un estandar de representación de objetos simples, como listas o diccionarios en forma texto, aunque originalmente fue desarrollado para JavaScript, por ello su nombre JavaScript Object Notation(JSON) esto es independiente de lenguaje, liviano , flexible y capaz de manejar gran cantidad de datos. Es así que se hace ideal para intercambiar información sobre protocolos como HTTP y un gran número de web APIs las cuales usan este como su principal formato de datos. [11]

8.4. Análisis de resultados y funcionamiento

Una vez el Agente ha llegado a su destino como nodo final de su itinerario el mismo ha generado el reporte por el cual fue creado, es decir ha cumplido su meta. Por ello el Agente detiene su ejecución y puede dar por terminada su tarea.

Es interesante como en este escenario se evidencia cada una de las características técnicas y conceptuales de la movilidad, el Agente que ha sido creado en Node1 puede desplazarse a través de la red y sobre su ambiente por Node2 y Node3 respectivamente desarrollando una función determinada por cada nodo que este recorre.

De igual forma la meta que se ha propuesto para este Agente de prueba es pertinente en el caso que nos permite ver como el Agente posee memoria, como el mismo lleva consigo no

sólo su código fuente sino además los datos y lo que ha aprendido durante su recorrido. Esto es fundamental para que Agente pueda guardar la información que ha aprendido y además de ellos controlar su itinerario y progreso.

CAPÍTULO 9

Usando la movilidad de un Agente sobre un Sistema virtualizado a través de Docker

9.1. Arquitectura del sistema virtualizado

El sub-sistema de Movilidad y los módulos que lo componen han sido diseñados para trabajar tanto un ambiente físico como el caso anterior como sobre uno virtual el cual será desarrollado en los siguientes párrafos.

Para la virtualización de los nodos ha sido configurado a través de Docker, el cual es un software de virtualización con contenedores que comparten el mismo kernel y que permiten configuran el ambiente y requisitos dentro de cada uno, para el caso aquí mencionado debe contener el lenguaje de programación Python.

Python3 por otra parte es considerado el futuro, constantemente esta obtiene nuevas y mejoras que nunca llegarán a Python2, por esta y mucas razones del lenguaje se ha decido realizar el proyecto sobre versiones de Python3 en general.[32]

En la siguiente figura se observa el esquema de conexión de los contenedores virtuales por donde el Agente usando su módulo de movilidad se desplazará:

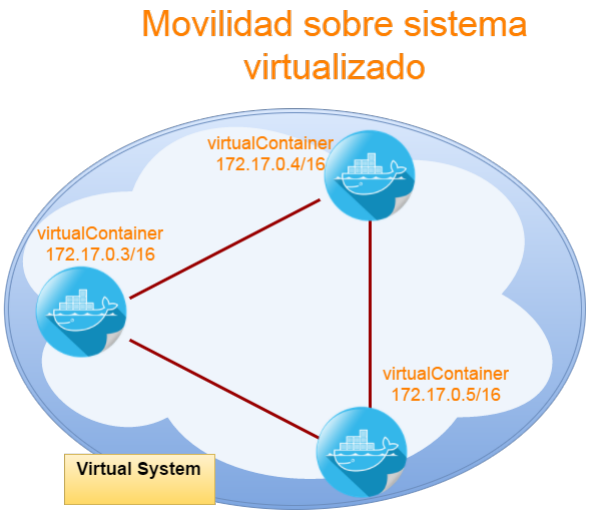

Figura 9-1.: Movilidad sobre un sistema virtualizado

La diferencia principal es que para este sistema sólo necesitamos un nodo físico, es así que este virtualizará los contenedores creando desde la perspectiva del Agente como si se moviera a través de sistemas y nodos físicos, cuando realmente el mismo esta desplazándose sobre un sistema virtualizado gracias a Docker. Esto nos permite intuir que la movilidad de un Agente no necesariamente siempre se ejecuta sobre dispositivos físicos, también es posible sobre dispositivos virtuales sin cambiar en sí lo que la movilidad representa.

En la ejecución del mismo la movilidad funciona sin complicación alguna, sólo que esta vez ha sido reemplazado el itinerario el cual es:

9.1.1. Itineario

1. 172.17.0.3
2. 172.17.0.4
3. 172.17.0.5

9.1.2. Creación de un contenedor

Dockerfile

El Dockerfile es el archivo que define las características de la imagen que se desea construir, imagen base para la creación de cada uno de los contenedores del sistema virtual. Para este caso a sido usada como base una imagen con sistema operativo Ubuntu con características sencillas que incluyen dentro de sí el sub-sistema de movilidad generado en el desarrollo de este trabajo.

```
FROM ubuntu

# Update Software repository
RUN apt-get update
RUN apt-get install -y python3
RUN apt-get install -y net-tools
ADD mobile-agent /
```

Contenedores

Un contenedor es una forma ligera, ejecutable de una pieza de software que posee librerías, herramientas del sistema, configuraciones y todo lo que necesite para ser ejecutado. Los contenedores son una forma de aislamiento para el software que corre dentro de ellos de su alrededor, es la forma de virtualización que ofrece docker para sus usuarios y servicios.
/// citar docker website y el libro de docker también y el dockerbook
Para este escenario han sido creados tres contenedores, los cuales poseen la misma imagen base de tal forma que poseen similitudes entre ellos, sin embargo las direcciones de red, identificadores y nombres los difieren.

En las siguientes figuras se muestra el despliegue de el Agente Móvil sobre el sistema virtualizado de contenedores y a su vez el reporte final que este arroja al completar la ruta de su itinerario.

9.2. Verificando las funciones del sub-sistema de movilidad acorde a su rol e itinerario

En secciones anteriores se ha mencionado los roles definidos y el comportamiento de movilidad que cada Agente adquiere acorde este rol. En el reporte extraído del sistema virtualizado se logra identificarse como el orden de los datos no corresponden estrictamente a el orden de itinerario dado inicialmente, esto se debe a que el rol que ha sido asignado al Agente ha sido çollector", lo que le da habilidad de recorrer aleatoria mente su itinerario.

9.3. Movilidad: Un sub-sistema del Sistema TLÖN

La razón por la que siempre se ha referido a la movilidad como un sub-sistema es porque conceptual mente ha sido diseñado para hacer parte de un sistema más grande, para este caso particular se ha tomado como base la definición y arquitectura del sistema TLÖN, sin embargo el sub-sistema aquí descrito puede ser expandido, modificado para su uso en sistema

con diferentes arquitecturas propósitos.

CAPÍTULO 10

Conclusiones y trabajo futuro

10.1. Conclusiones

Python puede definirse como un popular lenguaje de alto nivel, para propósito general, interpretado y dinámico. La filosofía de Python esta enfocada en la claridad y fácil entendimiento. Ademas de ello la sintaxis del mismo permite expresar conceptos con gran calidad y fácil entendimiento en pocas lineas comparados con otros lenguajes populares como Java y C++.//citar python nonsense
La creación del sub-sistema de movilidad ha incluido en una serie de módulos, funciones y decisiones que han sido tomadas siempre con lineamientos base del comportamiento de los Agentes y la posible influencia de su naturaleza, es decir, se ha inspirado las habilidades del mismo con la intención de cubrir sus necesidades de movilidad en un escenario real.

Sobre le trabajo desarrollado puede destacarse la modular, flexibilidad y arquitectura del proyecto, como cada una de los componentes juega un papel dentro del sub-sistema y como se integran entre sí, en el caso del Agente y su Ambiente se logra observar la comunicación y complemento de los mismos. Ademas de ello la ventaja que ofrece un despliegue del sub-sistema como único requerimiento del lenguaje Python el cual esta presente por defecto en la mayoría de sistemas operativos que usan el kernel Linux o Unix-like.

Es importante observar el nivel de abstracción que ofrece el la movilidad para el sistema Multi-Agente, como ha sido probado en los escenarios el sub-sistema es capaz de funcionar bajo un esquema de recursos físicos, un esquema de virtualizar contenedores como lo es Docker o incluso un sistema de virtualización completa como el caso de Oracle VirtualBox aunque este ultimo no fue incluido dentro del alcance del proyecto. Gracias al uso de librerías nativas, de protocolos de comunicación y siguiendo con las recomendaciones y buenas practicas ha sido generado un proyecto escalable, modular y que puede integrarse a sistemas más complejos.

Entre las funciones que permiten planear y ejecutar el itinerario y el modo de llevarlo a cabo ha sido tomado en consideración el rol del Agente dentro del sistema como tal, lo cual sin lugar a duda determinará su comportamiento en lo que a movilidad se refiere, estas funciones reciben características del Agente como lo es un rol, sin embargo cada vez que la función es llamada requiere este parámetro, esto representa una ventaja en el tiempo puesto que al ser el rol un atributo de Agente puede ser cambiado en un momento dado y el módulo movilidad que provee esta función entenderá que ahora que el Agente posee otro rol , seguramente requiere un esquema de movilidad o itinerario sea acorde a su nueva definición.

El objetivo o meta es una de las primordiales características de un Agente, para el desarrollo de los escenarios la meta del Agente utilizado ha sido de gran ayuda al demostrar un caso de uso real, el obtener los recursos del sistema, la cual fue la meta de este Agente de prueba es posible de lograr si se instalara en cada nodo o contenedor un Agente estático que envíe la información a un punto de concentración de información, pero en los escenarios de este proyecto usando el paradigma de Agentes Móviles fue posible el mismo resultado a través de un solo Agente, y sobre un sistema distribuido el cual era una de las razones del proyecto, dentro del trabajo futuro sera incluida una comparación de el alcance de esta meta con Agentes sin movilidad comparados con Agentes Móviles.

Finalmente en los escenarios de prueba tanto el que incluye el despliegue sobre un esquema físico como el que lo hace sobre una red y esquema virtual ha sido posible usar y observar la interacción y funcionalidad de cada uno de los componentes y módulos del sistema. Aunque el escenario cambio de un ambiente físico a un ambiente virtual el sub-sistema no requirió cambio alguno, gracias a que el diseño del mismo no lo limita a un sistema, fabricante o dispositivo físico.

10.2. Trabajo Futuro

Como trabajo futuro se plantea el despliegue del sub-sistema de movilidad y la incursión de nuevos Agentes Móviles sobre una red con mayor tamaño, con un incremento en los nodos participantes. De esta forma puede verse la movilidad para diferentes funciones fuera de la recolección de recursos presentada en este trabajo, también puede verse como habilidad para configuración, control y mantenimiento.

Recordemos que uno de las ventajas que han sido mencionadas sobre la Movilidad de agentes ha sido precisamente el ahorro y desempeño sobre redes limitadas que estos ofrecen referente a anchos de banda y reducción de trafico. Podría plantearse un escenario en el que se cuantifique el desempeño de la movilidad para una tarea en un modelo de Agentes Móviles cooperativos contra un modelo de Agentes sin movilidad.

Para dar por terminado se sugiere añadir al sub-sistema de movilidad la función de mover o transferir no sólo aquello que hace parte del Agente , sino además objetos externos al mismo, es decir, que un Agente podría pensar en viajar y llevar junto a este un Agente estático que se encuentre a su alcance sin modificar o hacer cambios en las características de este acompañante.

APÉNDICE A

Anexo: Nombrar el anexo A de acuerdo con su contenido

Los Anexos son documentos o elementos que complementan el cuerpo de la tesis o trabajo de investigación y que se relacionan, directa o indirectamente, con la investigación, tales como acetatos, cd, normas, etc.

Bibliografía

[1] *11.1. pickle — Python object serialization — Python v3.1.5 documentation*

[2] *JSON official*

[3] Abdel-Halim, Islam T. ; Fahmy, Hossam Mahmoud A. ; Bahaa-Eldin, Ayman M.: Agent-based trusted on-demand routing protocol for mobile ad-hoc networks. En: *Wireless Networks* (2015), Nr. 2, p. 467. – ISSN 1022–0038

[4] Bosse, Stefan: Intelligent microchip networks: an agent-on-chip synthesis framework for the design of smart and robust sensor networks. En: Riesgo, Teresa (Ed.) ; Conti, Massimo (Ed.): *Proceedings of the SPIE 2013, Microtechnologie Conference, Session EMT 102 VLSI Circuits and Systems, 24-26 April 2013, Alpexpo/Grenoble, France* Vol. 8764, 2013, p. 87640R

[5] Bosse, Stefan: Agent-based Solutions for Industrial Environments composed of Autonomous Mobile Agents, Modular Agent Platforms, and Tuple Spaces. En: *Proceedings of 2nd International Electronic Conference on Sensors and Applications*. Basel, Switzerland : MDPI, nov 2015, p. S5001

[6] Cao, Jiannong ; Das, Sajal K.: *MOBILE AGENTS IN NETWORKING AND DISTRIBUTED COMPUTING*. 2012

[7] Cassell, Laura ; Gauld, Alan: *Python Projects*. 1. Wrox, 12 2014. – ISBN 9781118908662

[8] Cicirello, V.A. ; Mroczkowski, A. ; Regli, W.: Designing decentralized software for a wireless network environment: evaluating patterns of mobility for a mobile agent swarm. En: *IEEE 2nd Symposium on Multi-Agent Security and Survivability, 2005.*, IEEE, 2005. – ISBN 0–7803–9447–X, p. 49–57

[9] Dean, Justin W. ; Macker, Joseph P. ; Chao, William: A study of Multiagent System operation within dynamic ad hoc networks. En: *MILCOM 2008 - 2008 IEEE Military Communications Conference*, IEEE, nov 2008. – ISBN 978–1–4244–2676–8, p. 1–7

[10] Dhanalakshmi, K ; Kadhar Nawaz, G M.: Fault-Tolerant Scheme for Matrix Hop Mobile Agent (MHMA) System. En: *Arabian Journal for Science & Engineering (Springer Science & Business Media B.V.)* 38 (2013), Nr. 12, p. 3339–3347. – ISSN 13198025

[11] Dr. M. O. Faruque Sarker, Sam W.: *Learning Python Network Programming*. Packt Publishing - ebooks Account, 2015. – ISBN 1784396001,9781784396008

[12] Frank H. P. Fitzek, Marcos D. K.: *Mobile Clouds: Exploiting Distributed Resources in Wireless, Mobile and Social Networks*. 1. Wiley, 2014. – ISBN 0470973897,9780470973899

[13] Grimaldi, Ralph P.: *Discrete and combinatorial mathematics : an applied introduction*. 5th ed. Pearson Addison Wesley, 2004. – ISBN 0201726343,9780201726343

[14] Higashino, M. ; Hayakawa, T. ; Takahashi, K. ; Kawamura, T. ; Sugahara, K.: Management of Streaming Multimedia Content Using Mobile Agent Technology on Pure P2P-based Distributed E-learning System. En: *2013 IEEE 27th International Conference on Advanced Information Networking and Applications (AINA)*, IEEE, mar 2013. – ISBN 978–1–4673–5550–6, p. 1041–1047

[15] Higashino, Masayuki ; Takahashi, Kenichi ; Kawamura, Takao ; Sugahara, Kazunori: Mobile Agent Migration Based on Code Caching. En: *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, IEEE, mar 2012. – ISBN 978–1–4673–0867–0, p. 651–656

[16] Horlait, Eric: *Mobile Agents for Telecommunication Applications*. Kogan Page Science, 2003 (Innovative technology series. Information systems and networks). – ISBN 9781903996287,1–9039–9628–7

[17] Jha, Sudan: Implementation of the Mobile Agent System using Resource Allocation Problem. En: *International Journal of Multidisciplinary Approach & Studies* 2 (2015), Nr. 5, p. 44–56. – ISSN 2348537X

[18] John Goerzen, Brandon R.: *Foundations of Python 3 Network Programming, Second Edition*. 2. 2010. – ISBN 1430230037,9781430230038

[19] K., Thomson: *UNIX programmer's manual*. 1971

[20] Laura Cassell, Alan G.: *Python Projects*. 1. Wrox, 2014. – ISBN 111890866X,9781118908662

[21] Leach, Paul J. ; Mealling, Michael ; Salz, Rich: A Universally Unique IDentifier (UUID) URN Namespace.

[22] Malik, Najmus S. ; Ko, David ; Cheng, Harry H.: A secure migration process for mobile agents. En: *Software: Practice and Experience* 41 (2011), jan, Nr. 1, p. 87–101. – ISSN 00380644

[23] Matthes, Eric: *Python Crash Course: A Hands-On, Project-Based Introduction to Programming*. No Starch Press, 2015. – ISBN 1593276036,9781593276034

[24] Mishchenko, Polina V.: Distributed computing system for solving applied tasks. En: *2015 16th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices*, IEEE, jun 2015. – ISBN 978–1–4673–6718–9, p. 154–157

[25] N. S. Nithya ; K. Duraiswamy. *Control, Automation, Communication and Energy Conservation, 2009. INCACEC 2009. 2009 International Conference on*. 2009

[26] O'Connor, TJ: *Violent Python A Cookbook for Hackers, Forensic Analysts, Penetration Testers and Security Engineers*. Syngress, 2012. – ISBN 978–1597499576

[27] Pierfederici, Francesco: *Distributed Computing with Python*. Packt Publishing, 2016

[28] Pierfederici, Francesco: *Distributed Computing with Python*. Packt Publishing, 2016

[29] Portnoy, Matthew: *Virtualization Essentials*. 1. Sybex, 2012. – ISBN 1118176715,9781118176719

[30] Ramji, Tangudu: Adaptive resource allocation and its scheduling for good tradeoff between power consumption and latency in OFDMA based wireless distributed computing system. En: *2015 International Conference on Computation of Power, Energy, Information and Communication (ICCPEIC)*, IEEE, apr 2015. – ISBN 978–1–4673–6525–3, p. 0496–0501

[31] Singh, Rajwinder ; Dave, Mayank: Antecedence Graph Approach to Checkpointing for Fault Tolerance in Mobile Agent Systems. En: *IEEE Transactions on Computers* 62 (2013), feb, Nr. 2, p. 247–258. – ISSN 0018–9340

[32] Slatkin, Brett: *Effective Python: 59 Specific Ways to Write Better Python (Effective Software Development Series)*. 1. Addison-Wesley Professional, 3 2015. – ISBN 9780134034287

[33] Terziyan, Joonas K. ; Vagan ; Alkhateeb, Faisal (Ed.): *Multi-Agent Systems - Modeling, Interactions, Simulations and Case Studies*. InTech, apr 2011. – ISBN 978–953–307–176–3

[34] Vigilson Prem, M ; Swamynathan, S: Mobile Agents for Information Retrieval: Detection and Recovery From Failures. En: *Arabian Journal for Science & Engineering (Springer Science & Business Media B.V.)* 39 (2014), Nr. 4, p. 2817–2829. – ISSN 13198025

[35] Wen, Heming ; Tiwary, Prabhat K. ; Le-Ngoc, Tho: *Wireless Virtualization*. Cham : Springer International Publishing, 2013 (SpringerBriefs in Computer Science). – ISBN 978–3–319–01290–2

[36] Yamanouchi, Akihiro ; Hashimoto, Takeshi ; Ohta, Tomoyuki ; Kojima, Hideharu ; Kakuda, Yoshiaki: Resource Management Middleware Using Mobile Agents for Mobile Ad Hoc Networks. En: *2010 IEEE 30th International Conference on Distributed Computing Systems Workshops*, IEEE, jun 2010. – ISBN 978–1–4244–7471–4, p. 1–6

[37] Yousuf, Farzana ; Zaman, Zahid: A Survey of Fault Tolerance Techniques in Mobile Agents and Mobile Agent Systems. En: *2009 Second International Conference on Environmental and Computer Science*, IEEE, 2009. – ISBN 978–1–4244–5590–4, p. 454–458