UNIVERSIDAD TECNICA ESTATAL DE QUEVEDO

2023 - 2024

- COELLO CASTILLO RAUL STEVEN
- LETURNE PLUAS JHON BYRON
- OCHOA GILCES GEOVANNY ALEXANDER

APLICACIONES DISTRIBUIDAS

AGENTES MOVILES

UTEQ
UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

Tabla de contenido

Tabla de figuras

1. Introducción

Los agentes móviles son programas independientes con la capacidad de transitar entre computadoras dentro de una red, eligiendo libremente cuándo y dónde realizar sus viajes. El estado del programa en ejecución se almacena y se transmite al destino correspondiente. Al llegar al destino, el programa prosigue su procesamiento desde el estado previamente guardado.

Estos agentes ofrecen un marco conveniente, eficiente y sólido para la implementación de aplicaciones distribuidas y entornos inteligentes. Este enfoque presenta beneficios tales como mejoras en la latencia y el ancho de banda en aplicaciones cliente-servidor, así como la reducción de la vulnerabilidad ante desconexiones de red [1].

Un agente móvil tiene la capacidad de trasladarse de una computadora, como una portátil, a través de Internet para recopilar información en beneficio de su usuario. Su eficiencia radica en la capacidad de acceder a los recursos necesarios mientras se desplaza en la red, evitando así la transferencia de múltiples solicitudes y respuestas a través de la conexión de una computadora portátil con ancho de banda limitado. Al no depender continuamente de la conexión de la computadora portátil, el agente no se ve afectado por pérdidas repentinas de conexión y puede proseguir con sus tareas incluso si el usuario apaga su dispositivo o se desconecta de la red.

Cuando el usuario se vuelve a conectar, el agente retorna a la computadora portátil con los resultados obtenidos durante sus desplazamientos. En contraste, una aplicación que reside en la red puede enviar un agente móvil a la computadora portátil. Este agente actúa como un sustituto de la aplicación, interactuando eficientemente con el usuario y continuando su interacción incluso en situaciones de desconexión prolongada. Aunque no todas las aplicaciones para sistemas distribuidos requieren el uso de agentes móviles, muchas encontrarán en ellos la técnica más eficaz para implementar total o parcialmente sus tareas.

La tecnología de agentes móviles se considera un tipo de tecnología de agentes de software, pero no siempre se requiere que posea capacidades inteligentes, como comportamientos reactivos, proactivos y sociales que son características de las tecnologías de agentes de software existentes. Esto se debe a que estas capacidades tienden a ser demandantes en términos de escala y procesamiento, y se prefiere que

ningún agente móvil consuma recursos computacionales excesivos, como procesadores, memoria, archivos y redes, en sus destinos. Además, esta tecnología se percibe como un enfoque de implementación de sistemas distribuidos en lugar de sistemas inteligentes [1],[2].

Este documento este constituido por tres fases las cuales son desarrollo donde damos a conocer los conceptos necesarios del tema tratado características, ventajas, además de dar a conocer un ejemplo real de cómo es el funcionamiento de un agente móvil para su oportuna comprensión finalmente presentamos una conclusión del tema.

2. Objetivos

En esta sección se describen los objetivos que se han planteado para la elaboración del presente informe.

2.1. General

- El objetivo general del informe es explorar y analizar de manera exhaustiva el concepto de agentes móviles, sus aplicaciones en sistemas distribuidos y entornos inteligentes, así como sus beneficios y desafíos asociados.

2.2. Específicos

- Investigar Aplicaciones Prácticas de Agentes Móviles: Analizar casos de estudio y ejemplos prácticos que ilustren cómo los agentes móviles se utilizan, centrándose en su capacidad para mejorar la eficiencia en la recopilación de información, la optimización de recursos y la interacción con usuarios en entornos distribuidos.
- Evaluar Desafíos y Soluciones en la Implementación de Agentes Móviles: Identificar y analizar los desafíos comunes asociados con el uso de agentes móviles, como cuestiones de seguridad, privacidad y consumo de recursos. Proponer posibles soluciones y mejores prácticas para mitigar estos desafíos y garantizar una implementación eficiente y segura de la tecnología de agentes móviles.

3. Desarrollo

A continuación, se expone de manera detallada el conjunto teórico que abarca el informe sobre la utilización de agentes móviles como paradigma dentro del contexto de los sistemas distribuidos. En este apartado, se proporciona una ampliación y profundización en los conceptos fundamentales relacionados con la presencia y función de los agentes móviles en entornos distribuidos.

3.1. Sistema distribuido

Hoy en día las telecomunicaciones han permitido que un gran número de usuarios estén conectados entre sí mediante los sistemas, mediante de transmisión de video, voz texto etc. Todo esto refleja un gran consumo dentro de una red al este asociado a solo un nodo central para lo cual las aplicaciones distribuidas en diferentes áreas han permitido flexibilidad en la implementación de sistemas, así como también interactuar con otros sistemas de manera distribuida. ***Tanenbaum 1996*** menciona que un sistema distribuido de define como un conjunto de máquinas que se presentan antes los usuarios como un único recurso [3].

Un sistema distribuido puede definirse como una red de elementos informáticos, denominados nodos, caracterizados por dos características fundamentales. En primer lugar, cada nodo, que representa un dispositivo de hardware o un proceso de software, posee la capacidad de operar independientemente de otros nodos dentro del sistema. En segundo lugar, los usuarios, ya sean individuos o aplicaciones, perciben e interactúan con el sistema distribuido como una entidad unificada, a pesar de su naturaleza descentralizada.

En esencia, la autonomía de los nodos individuales es un atributo clave que les permite funcionar de forma independiente mientras contribuyen colectivamente a los objetivos del sistema general. La colaboración entre estos nodos autónomos se vuelve fundamental, constituyendo un desafío central en el desarrollo de sistemas distribuidos.

Es importante destacar que esta definición se abstiene de hacer suposiciones sobre la naturaleza o el tipo de nodos, reconociendo que pueden abarcar desde poderosas computadoras centrales hasta dispositivos con recursos limitados en redes de sensores. Además, no existen suposiciones predefinidas sobre los mecanismos de

interconexión específicos entre nodos, lo que permite flexibilidad en la arquitectura del sistema distribuido [4].

Ventajas

1. Velocidad: Un sistema al estar distribuido puede tener mayor velocidad de procesamiento a diferencia a uno que está alojado en un solo nodo [5].
2. Confiabilidad: Un sistema seguro ya que si ocurre cualquier tipo de falla en un nodo esta puede seguir trabajando sin problemas [3], [5].
3. Crecimiento del sistema: Cada vez que se requiera se puede elevar lo poder de procesamiento de cómputo [5].

Desventajas

1. Los problemas por transmisión de grandes volúmenes de datos siguen siendo un problema por ejemplo archivos multimedia [5].
2. Tolerancia a fallas por los problemas que se sucinta la mayoría de las veces tanto en forma operativa o en componentes [3], [5].

3.2. JAVA-RMI (Java Remote Method Invocation)

Java ofrece diferentes oportunidades al momento de la programación de acuerdo a esto esta mismo la invocación a RMI la cual es una función del lenguaje que gestiona un tipo flexible de llamadas a métodos remotos. Estos métodos pueden ser invocados desde una máquina virtual diferente de java además de describir interfaces que permitir a los métodos del objeto invocarse remotamente [6].

RMI fue diseñado por javasoft la cual esta misma soporta llamadas a métodos remotos entre objetos. Esta implementación de procesos sirve para la ejecución de tareas en sistemas remotos o heterogéneos además el servidor de java/RMI tiene que ser escrito en lenguaje Java [6].

Características de Java/RMI.

1. Comunicación entre objetos remotos.
2. Facilidad de uso.
3. Uso de interfaces para su implementación.
4. Paso de parámetros a través de la llamada a métodos remotos.

3.3. ¿Qué son los agentes móviles?

Los agentes móviles es un tipo de paradigma de programación en el cual es una alternancia a de Cliente/Servidor. En algunas aplicaciones representan una solución a ciertas tareas ya que estas reducen considerablemente el tráfico de la red además de reducir la latencia de la misma. Hay una representación considerable dentro de esta, ya que si la red esta fuera de servicio esta permite reanudar sus tareas cuando ya esté operativa [7].

Para ***Gray 1995*** define a un agente móvil como un programa transportable dentro de los nodos de la red en la cual esta puede elegir a donde ubicarse, además de poder ejecutar tareas e información en el servidor donde este alojado. Los agentes móviles se pueden mover por una red WAN como WWW en la cual van interactuando en cada uno de sus nodos recolectando información de las maquinas a la cual se esta conectando una vez finalizado regresa a su lugar de partida con la información recolectada [5], [7].

Por otro lado, ***Vitek 1996*** define a un agente móvil como objetos que se están ejecutando en nombre de un usuario, es decir, ejecutar tareas remplazando al usuario final. Este agente puede moverse por la red enchutando diversas tareas la cual este puede ser interrumpido y reanudarse cundo todo esté resuelto [7].

El paradigma o comportamiento de estos agentes móviles facilita mucho la gestión de microservicios que son complejos de gestionar ya que permite tener un mejor autonomía y transferencia de datos entre ordenadores [8].

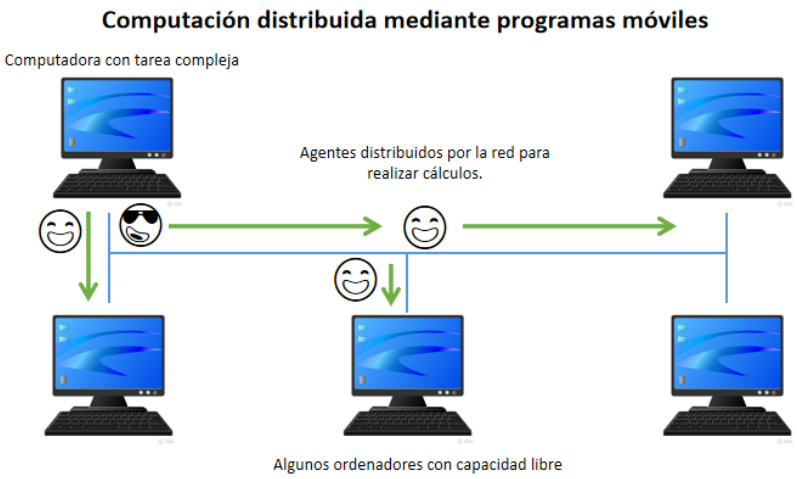


Tabla 1. Representación del flujo de los agentes móviles entre una red de computadoras [9]

Ventajas

1. Eficiencia al manejo de grandes volúmenes de datos.
2. Reduce al tráfico de la red al tener diferentes nodos a cuál pueda obtener información.
3. Comunicación en tiempo real con los nodos.
4. La ejecución de las tareas es asíncrona [8], [9].
5. Las tareas se encapsulan: esto permite su traslado hacia computadoras remotas con el fin de cumplir sus objetivos. Esta movilidad evita la necesidad de instalar un proceso de servidor especializado en cada máquina para facilitar el acceso a un servicio específico.

   En cambio, se emplea un único proceso de servidor, denominado plataforma de agente móvil, que posibilita que múltiples agentes con distintas funcionalidades viajen hacia esa computadora para ofrecer los servicios requeridos. Esta característica proporciona una considerable flexibilidad en el desarrollo de aplicaciones destinadas a entornos distribuidos [9].
6. Reducen la carga de la red: al posibilitar que un agente móvil se desplace hacia donde residen los datos y acceda a ellos localmente, aplicando un filtro para enviar únicamente los datos necesarios a través de la red. Esta estrategia de trasladar el proceso de cómputo hacia los datos, en lugar de seguir la lógica inversa, resulta en un ahorro significativo de recursos, especialmente cuando se enfrenta el análisis de grandes volúmenes de datos.

   Además, al procesar datos directamente en computadoras remotas, los agentes móviles prescinden de la necesidad constante de conexiones de red, utilizando la red únicamente durante el desplazamiento del agente móvil hacia otra computadora. Este enfoque optimizado contribuye a la eficiencia en la utilización de recursos y a la reducción de la carga en la infraestructura de red [9].
7. Superan la latencia de la red: al tener la capacidad de trasladarse a otra computadora, optimizando así el tiempo de respuesta. La habilidad de los agentes móviles para disminuir la latencia resulta esencial en sistemas que requieren respuestas en tiempo real, como en el caso de robots involucrados en procesos de fabricación.

De esta manera, es posible despachar agentes desde un controlador central hacia una computadora específica para controlar el robot de manera local, evitando así múltiples interacciones remotas y, en consecuencia, reduciendo las comunicaciones a través de la red. Este enfoque no solo mejora la eficiencia del control en tiempo real, sino que también ahorra recursos de red al minimizar las interacciones a distancia [9].

8. Son asincrónicos y autónomos: En las arquitecturas convencionales cliente/servidor sincrónico, el cliente debe mantener activa la conexión mientras el servidor procesa su solicitud. En caso de una interrupción en la conexión, el cliente se ve obligado a reenviar la solicitud al servidor, que la procesará desde el principio.

   En contraste, un agente móvil no requiere mantener una comunicación constante con su computadora de origen mientras lleva a cabo sus tareas. Esta característica es especialmente valiosa en dispositivos móviles, que suelen depender de conexiones inalámbricas costosas y poco confiables. De esta manera, un dispositivo móvil puede enviar un agente móvil a una computadora en la red fija y desconectarse posteriormente.

   El agente se vuelve independiente de su dispositivo de origen, representando una forma flexible de delegar tareas a la red. Al restablecer la conexión, el dispositivo puede recuperar el agente móvil o sus resultados de manera eficiente [9].
9. Se adaptan dinámicamente a su entorno: Poseen la capacidad de adaptarse dinámicamente a su entorno al tener la habilidad de percibir el entorno circundante. Estos agentes pueden ser programados para reaccionar de manera autónoma y ajustarse a los cambios que ocurran en su entorno. Un ejemplo ilustrativo de esta capacidad es la capacidad de desplazarse hacia otra computadora en situaciones de sobrecarga en la computadora actual, contribuyendo así a equilibrar la carga del sistema de manera eficiente [9].

Características

Los agentes móviles exhiben características claramente definidas dentro de su paradigma, lo cual simplifica la toma de decisiones respecto a su utilización. A continuación, se resaltan algunas de estas características distintivas, que contribuyen a entender su funcionalidad y utilidad en diversos contextos:

1. La movilidad constituye la característica central en el concepto de agentes móviles, otorgándoles la capacidad de desplazarse de forma autónoma entre nodos dentro del mismo entorno o entre nodos en entornos diferentes. Funcionando como proxy, estos agentes móviles pueden actuar en representación de individuos o en beneficio de entidades específicas, como sistemas de software. Para ejercer en nombre de otros, es imperativo que los agentes móviles posean al menos un nivel mínimo de autonomía [10].
2. En cuanto a su naturaleza proactiva, los agentes móviles deben ser entidades orientadas a objetivos, tomando la iniciativa para adaptarse al entorno y responder de manera eficaz. Se espera que estos agentes móviles exhiban un grado de inteligencia al basarse en el conocimiento para llevar a cabo sus funciones con eficiencia. Coordinativamente, las autoridades de gestión deben tener la capacidad de realizar actividades de transferencia de datos, compartiéndolas con otros agentes dentro del entorno específico [10].
3. El aprendizaje se refiere a la capacidad de los agentes móviles para adquirir información sobre el entorno actual, lo cual contribuirá a la modificación de su comportamiento. En un enfoque cooperativo, se espera que los agentes móviles puedan coordinarse con otros para alcanzar un propósito común. Adicionalmente, los agentes móviles deben ser capaces de manejar irregularidades y errores de manera efectiva durante su ocurrencia [10].

3.4. Ciclo de vida de los agentes móviles

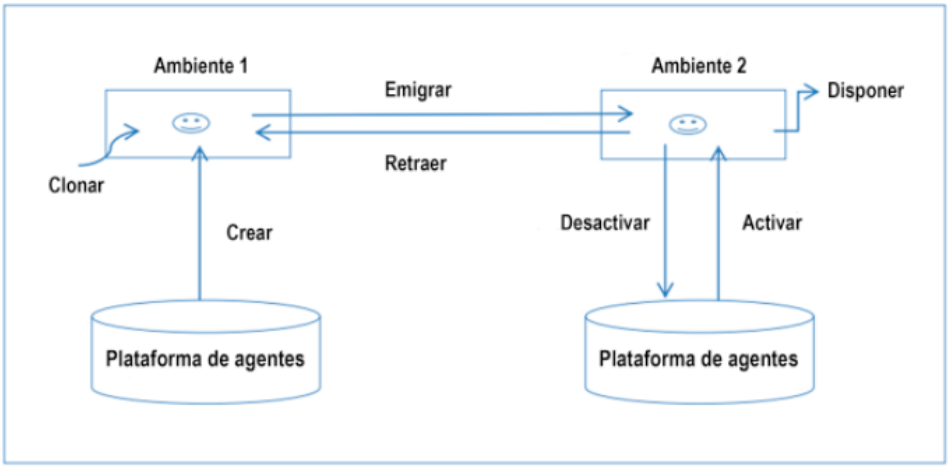


Tabla 2. Diagrama sobre el ciclo de vida de un agente móvil [11].

Los agentes móviles siguen una serie de procesos para llevar a cabo una tarea, conocidos como el Ciclo de Vida del Agente Móvil, que se ilustra en la figura y se destaca de la siguiente manera [11]:

- **Creación**: Inicia el ciclo de vida de un agente móvil, donde se crea una instancia del agente y se establece su estado al recibir una solicitud.
- **Despacho**: Implica el traslado del agente móvil de un nodo a otro, siendo posible al especificar la dirección de destino.
- **Clonación**: Consiste en la creación de una copia del objeto original del agente móvil, generando un agente gemelo con el mismo estado que el original.
- **Desactivación**: Se produce cuando un agente móvil se suspende, almacenando su estado en el disco del host.
- **Activación**: Se realiza cuando se reactiva un agente móvil previamente desactivado, restaurando su estado desde el disco.
- **Retracción**: Se refiere al retorno de un agente desde un host remoto junto con su estado a la máquina de origen tras finalizar su tarea.
- **Eliminación**: Marca el final del ciclo de vida del agente móvil. En esta etapa, el agente se termina y su estado se pierde de manera irreversible.

3.5. Ataques y protecciones con agentes móviles

Bomba lógica o ataque desencadenado por evento, inicia cualquiera de los ataques basado en un evento externo.

- **En el caso de los hosts:** Una bomba lógica "estalla" cuando el código, oculto dentro de un agente móvil aparentemente inofensivo, se activa por un evento específico, como el tiempo (por ejemplo, el virus del Día de Colón), la ubicación o la llegada de una persona específica. Este tipo de ataque es característico de un programa troyano. Un ejemplo sería un agente móvil dentro del perímetro de un firewall que utiliza un módem para dar acceso periódicamente a colaboradores en Internet [11],[12].
- **En el caso de los agentes móviles:** Los agentes móviles pueden ser atacados debido a lo que transportan consigo o porque provienen de un remitente específico [12].

Un ataque compuesto se compone de múltiples ataques que pueden lograr más que un solo ataque.

- **En relación con los hosts:** Mediante el uso de técnicas de cooperación provenientes de la investigación de agentes, los agentes móviles pueden colaborar entre sí para llevar a cabo una serie de ataques [12].
- **En relación con los agentes móviles:** Los hosts o agentes móviles pueden rastrear o acechar a un agente móvil con la intención de robarle, descubrir su origen o destino, entre otros propósitos. Múltiples hosts colaboradores pueden rastrear al agente móvil hasta que este posea algo de interés, para luego llevar a cabo un ataque [11],[12].

A continuación, algunas recomendaciones para proteger los hosts y agentes móviles de otros agentes móviles:

- **En cuanto a la protección del host contra el agente móvil:** se pueden presentar amenazas como la suplantación, donde un agente no autorizado asume la identidad de otro para obtener acceso indebido a servicios y recursos, dañando la confianza y reputación en la comunidad de agentes. También se puede enfrentar a ataques de denegación de servicio, donde agentes móviles consumen excesivos recursos de la plataforma, ya sea intencionalmente mediante scripts de ataque o inadvertidamente debido a errores de programación.

  Estos agentes maliciosos pueden llevar consigo código diseñado para interrumpir servicios, degradar el rendimiento del host o extraer información no autorizada. Los mecanismos de control de acceso son esenciales, requiriendo que la plataforma autentique la identidad del agente móvil antes de la instanciación para prevenir accesos no autorizados que podrían perjudicar a otros agentes y a la plataforma misma [12].
- **Protección del agente móvil contra otros agentes móviles:** se abordan amenazas donde los agentes aprovechan debilidades de seguridad de otros o lanzan ataques, como suplantación, acceso no autorizado, denegación de servicio y repudiación. Los componentes de las plataformas, muchos de los cuales son agentes, proporcionan servicios a nivel del sistema, como directorios e intercomunicación.

Un agente podría intentar engañar al agente con el que se comunica ocultando su identidad o haciéndose pasar por un proveedor conocido. Además, se destaca la posibilidad de ataques de denegación de servicio y repudiación, donde un agente sobrecarga a otro con mensajes o niega la realización de una transacción. En este contexto, los mecanismos de control de acceso son esenciales, ya que evitan que usuarios o procesos no autorizados accedan a servicios y recursos sin permisos.

La autenticación de la identidad del agente móvil antes de su instanciación en la plataforma y registros detallados de transacciones son cruciales. También se aborda el riesgo de interferencia directa entre agentes, donde un agente malicioso podría modificar el código o datos de otro, transformándolo en uno no confiable. En resumen, la seguridad de los agentes móviles implica salvaguardar la integridad de las transacciones y proteger los recursos de la plataforma y otros agentes contra posibles amenazas y ataques [11], [12].

- **Protección del Agente Móvil contra el Anfitrión:** La protección del agente móvil frente al anfitrión se presenta como el desafío más significativo entre los tres problemas fundamentales de seguridad. Aunque se han realizado esfuerzos de investigación para abordar esta cuestión, aún no se ha encontrado una solución integral. Hasta ahora, solo se han logrado soluciones parciales.

  Las amenazas en esta categoría pueden incluir suplantación, ataques de denegación de servicio, escuchas y alteraciones. Debido a los problemas mencionados anteriormente, muchos anfitriones pueden restringir la interacción con los agentes, ya que estos podrían ser percibidos como gusanos (virus), dada la similitud en el funcionamiento entre los agentes y los gusanos.

  A pesar de ello, un ataque de un anfitrión malicioso contra un agente, al modificar su comportamiento o robar información confidencial como números de tarjetas de crédito, podría representar una amenaza económica considerable [12].

La siguiente figura muestra cómo pueden existir ataques mediante agentes, en este caso la Plataforma 2 contiene un malware lo que hace que los agentes que se dirigen hacia la plataforma 3 y 4 se infecten y por ende contagien a los siguientes agentes y plataformas.

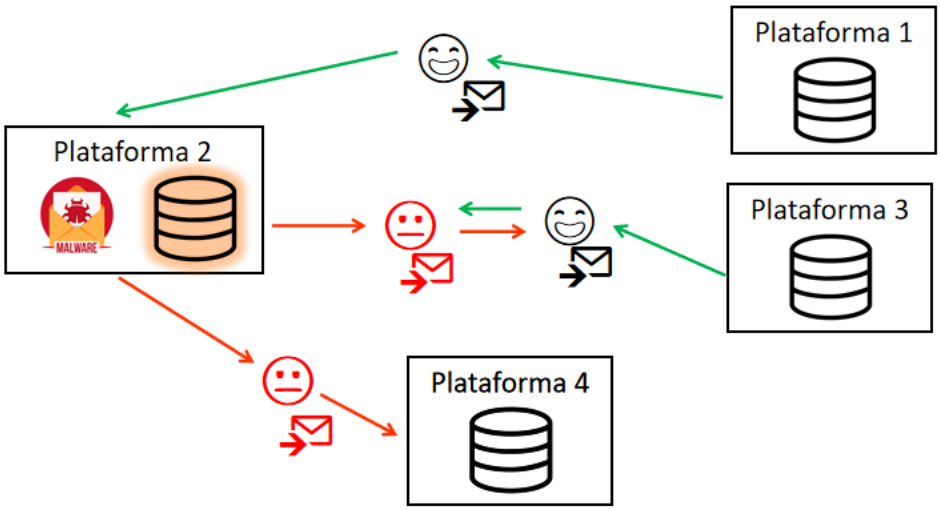


Tabla 3. Diagrama sobre un host infectando a varios agentes y sus host [13].

A continuación, el diagrama sobre el recorrido que hace un sistema de seguridad para protege una plataforma.

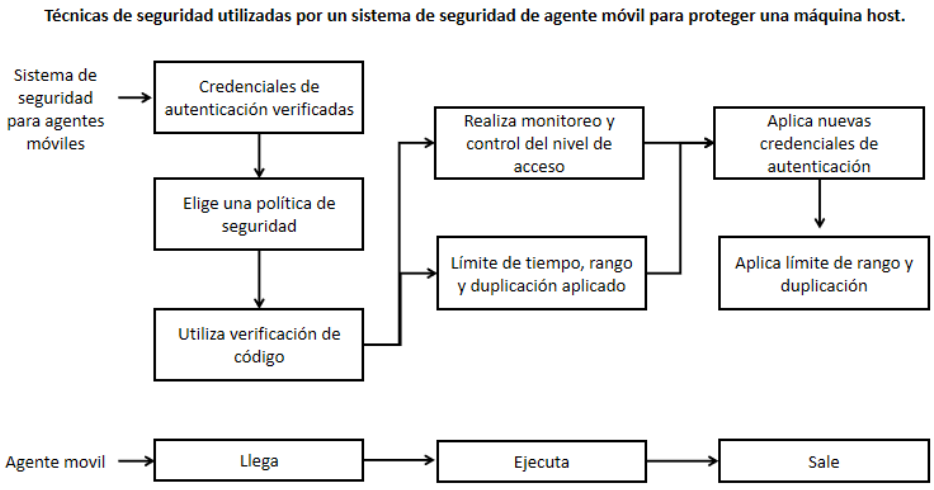


Tabla 4. Esquema de seguridad de una plataforma de agentes [13].

4. Ejemplo

El siguiente ejemplo está construido en el lenguaje de programación java basándonos en los métodos remotos que ofrece java o también llamados JAVA-RMI los cuales estos sirven para construir agentes y servidores que nos van a permitir ver el funcionamiento de un agente móvil.

El ejemplo consta de un agente en el cual se va a estar moviendo sobre tres servidores para recolectar información del SO de las maquinas. Finalmente, cuando acabe la recolección de los datos estos serán enviados a un servidor central el cual se encargará de guardarlos a la base de datos.

Agente móvil

Esta clase hace referencia al agente que se va a estar moviendo entre los servidores recolectando información y trayéndola devuelta al punto de partida.

```
package com.mycompany.ejemploagentemovil;

import java.rmi.Naming;
import java.util.ArrayList;
import java.util.List;
/**
 *
 * @author USUARIO
 */
public class EjemploAgenteMovil {

    public static List<String> lst_nodos;

    public static void main(String[] args)
    {
        lst_nodos=new ArrayList<String>();
        lst_nodos.add("serviceInfo1");
        lst_nodos.add("serviceInfo2");
        lst_nodos.add("serviceInfo3");
        try
        {

            Thread thread_response=new Thread(new Runnable() {
                @Override
                public void run()
                {
                    try
                    {
                        Entity epc1,epc2,epc3;
```

```
                        System.out.print("Moviendose a el primer nodo...\n");
                        Iespecificaciones dd1=(Iespecificaciones)
Naming.lookup("//localhost:3331/"+lst_nodos.get(0));
                        epc1= dd1.imprimir();



                        Thread.sleep(3000);

                        System.out.print("Moviendose a el segudo nodo...\n");
                        Iespecificaciones dd2=(Iespecificaciones)
Naming.lookup("//localhost:3332/"+lst_nodos.get(1));
                        epc2 = dd2.imprimir();

                        Thread.sleep(3000);

                        System.out.print("Moviendose a el tercer nodo...\n");
                        Iespecificaciones dd3=(Iespecificaciones)
Naming.lookup("//localhost:3333/"+lst_nodos.get(2));
                        epc3 = dd3.imprimir();



                        Thread.sleep(3000);

                        System.out.print("Moviendose a el nodo central para
guardar informacion...\n");
                        Iespecificaciones dd4=(Iespecificaciones)
Naming.lookup("//localhost:3334/serviceCental");
                        List<Entity> lst_epc=new ArrayList<>();
                        lst_epc.add(epc1);
                        lst_epc.add(epc2);
                        lst_epc.add(epc3);
                        dd4.guardar_base_dato(lst_epc);
                        dd4.mostrarTodo();

                    }
                    catch(Exception ex)
                    {
                        System.out.println(ex.getMessage());
                    }
                }
            });

            thread_response.start();
            Thread.sleep(5000);
        }
        catch(Exception ex)
```

```
        {
            System.out.println(ex.getMessage());
        }
    }
}
```

Clase intermediaria

```
package com.mycompany.ejemploagentemovil;

import java.io.Serializable;
/**
 *
 * @author USUARIO
 */
public class Entity implements Serializable
{
    private String procesador;
    private String tipo_disco;
    private String velocidad_red;
    private String version_tarjeta_wifi;
    private String sistema_operativo;
    private String servidor;

    public Entity(String procesador, String tipo_disco, String velocidad_red,
String version_tarjeta_wifi, String sistema_operativo, String servidor) {
        this.procesador = procesador;
        this.tipo_disco = tipo_disco;
        this.velocidad_red = velocidad_red;
        this.version_tarjeta_wifi = version_tarjeta_wifi;
        this.sistema_operativo = sistema_operativo;
        this.servidor = servidor;
    }


    public String getProcesador()
    {
        return procesador;
    }

    public void setProcesador(String procesador) {
        this.procesador = procesador;
    }

    public String getTipo_disco() {
        return tipo_disco;
    }
```

```
    public void setTipo_disco(String tipo_disco) {
        this.tipo_disco = tipo_disco;
    }

    public String getVelocidad_red() {
        return velocidad_red;
    }

    public void setVelocidad_red(String velocidad_red) {
        this.velocidad_red = velocidad_red;
    }

    public String getVersion_tarjeta_wifi() {
        return version_tarjeta_wifi;
    }

    public void setVersion_tarjeta_wifi(String version_tarjeta_wifi) {
        this.version_tarjeta_wifi = version_tarjeta_wifi;
    }

    public String getSistema_operativo() {
        return sistema_operativo;
    }

    public void setSistema_operativo(String sistema_operativo) {
        this.sistema_operativo = sistema_operativo;
    }

    public String getServidor() {
        return servidor;
    }

    public void setServidor(String servidor) {
        this.servidor = servidor;
    }
}
```

Clase remota

Esta clase es la que se va a comunicar con el agente para la obtención de información.

```
package com.mycompany.ejemploagentemovil;

import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.ResultSet;
import java.util.List;
import java.util.logging.Level;
```

```java
import java.util.logging.Logger;
/**
 *
 * @author USUARIO
 */
public class EspecificacionesPC extends UnicastRemoteObject implements
Iespecificaciones{

    private String procesador;
    private String tipo_disco;
    private String velocidad_red;
    private String version_tarjeta_wifi;
    private String sistema_operativo;
    private String servidor;

    private ResultSet rs;

    public ResultSet getRs() {
        return rs;
    }

    public void setRs(ResultSet rs) {
        this.rs = rs;
    }

    public String getServidor() {
        return servidor;
    }

    public void setServidor(String servidor) {
        this.servidor = servidor;
    }

    public String getProcesador() {
        return procesador;
    }

    public void setProcesador(String procesador) {
        this.procesador = procesador;
    }

    public String getTipo_disco() {
        return tipo_disco;
    }

    public void setTipo_disco(String tipo_disco) {
        this.tipo_disco = tipo_disco;
    }
```

```
    public String getVelocidad_red() {
        return velocidad_red;
    }

    public void setVelocidad_red(String velocidad_red) {
        this.velocidad_red = velocidad_red;
    }

    public String getVersion_tarjeta_wifi() {
        return version_tarjeta_wifi;
    }

    public void setVersion_tarjeta_wifi(String version_tarjeta_wifi) {
        this.version_tarjeta_wifi = version_tarjeta_wifi;
    }

    public String getSistema_operativo() {
        return sistema_operativo;
    }

    public void setSistema_operativo(String sistema_operativo) {
        this.sistema_operativo = sistema_operativo;
    }



    public EspecificacionesPC()throws RemoteException{};

    @Override
    public Entity imprimir()  throws RemoteException{
        System.out.println("PROCESADOR: "+this.getProcesador());
        System.out.println("TIPO DISCO: "+this.getTipo_disco());
        System.out.println("VELOCIDAD DE RED: "+this.getVelocidad_red());
        System.out.println("VERSION DE TARJETA WIFI: "+this.getProcesador());
        System.out.println("SISTEMA OPERATIVO: "+this.getProcesador());
        return new
Entity(this.getProcesador(),this.getTipo_disco(),this.getVelocidad_red(),this.
getProcesador(),this.getProcesador(),this.getServidor());
    }

    @Override
    public EspecificacionesPC especificaciones(String procesador, String
tipo_disco,
            String velocidad_red, String version_tarjeta_wifi, String
sistema_operativo,String servidor) throws RemoteException {
        try {
            EspecificacionesPC epc=new EspecificacionesPC();
```

```
            epc.setProcesador(procesador);
            epc.setTipo_disco(tipo_disco);
            epc.setVelocidad_red(velocidad_red);
            epc.setVersion_tarjeta_wifi(version_tarjeta_wifi);
            epc.setSistema_operativo(sistema_operativo);
            epc.setServidor(servidor);
            return epc;
        } catch (RemoteException ex) {
            Logger.getLogger(EspecificacionesPC.class.getName()).log(Level.SEV
ERE, null, ex);
        }
        return null;
    }

    @Override
    public Boolean guardar_base_dato(List<Entity> lst_epc) throws
RemoteException {
        try
        {
            conexionPostgres postg=new conexionPostgres();
            for(Entity lepc : lst_epc)
                postg.query(lepc);
            setRs(postg.mostrar_inst());
        }
        catch(Exception ex)
        {
            System.out.println(ex.getMessage());
        }
        return false;
    }

    @Override
    public void mostrarTodo() throws RemoteException
    {
        try
        {
            ResultSet rs=getRs();
            while(rs.next())
                System.out.println("ID= "+rs.getInt("id_c")+
                        " PROCESADOR= "+rs.getString("procesador")+
                        " Tipo Disco= "+rs.getString("velocidad_red")+
                        " Version Tarjeta wifi= "+rs.getString("v_tarj_wifi")+
                        " Sistema operativo=
"+rs.getString("sistema_opertivo")+
                        " Servidor= "+rs.getString("servidor"));
            rs.close();
        }
        catch(Exception ex)
```

```
        {
            System.out.println(ex.getMessage());
        }
    }
}
```

Interfaz para el control de la clase remota

Sirve para que los métodos que implementen sean remotos y se puedan comunicar por medio de la red.

```
package com.mycompany.ejemploagentemovil;

import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
/**
 *
 * @author USUARIO
 */
public interface Iespecificaciones extends Remote
{
    public EspecificacionesPC especificaciones(String procesador,
            String tipo_disco,String velocidad_red,
            String version_tarjeta_wifi,String sistema_operativo,String
servidor) throws RemoteException;
        public Entity imprimir() throws RemoteException;

    public Boolean guardar_base_dato(List<Entity> lst_epc) throws
RemoteException;

    public void mostrarTodo() throws RemoteException;
}
```

Servidor central

Para el guardado de todos los datos recolectados.

```
package com.mycompany.ejemploagentemovil;

import java.net.InetAddress;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
/**
 *
 * @author USUARIO
 */
```

```
public class ServidorCentral {

    public static void main(String[] args)
    {
        try
        {
            Registry rs = LocateRegistry.createRegistry(3334);
            rs.rebind("serviceCental", new EspecificacionesPC());
            System.out.println("Servidor central
activo"+InetAddress.getLocalHost().getHostName());
        }
        catch(Exception ex)
        {
            System.out.println(ex.getMessage());
        }
    }
}
```

Servidor uno

```
package com.mycompany.ejemploagentemovil;

import java.net.InetAddress;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
/**
 *
 * @author USUARIO
 */
public class ServidorUno {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        try
        {
            Registry rs = LocateRegistry.createRegistry(3331);
            rs.rebind("serviceInfo1", new EspecificacionesPC()
                                            .especificaciones("I7", "SSD",
                                                    "1MB/S", "3.0",
                                                    "Windows
10","serviceInfo1"));
            System.out.println("Servidor uno
activo"+InetAddress.getLocalHost().getHostName());
        }
        catch(Exception ex)
```

```
        {
            System.out.println(ex.getMessage());
        }
    }
}
```

Servidor dos

```
package com.mycompany.ejemploagentemovil;

import java.net.InetAddress;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;

/**
 *
 * @author USUARIO
 */
public class ServidorDos {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
         try
        {
            Registry rs = LocateRegistry.createRegistry(3332);
            rs.rebind("serviceInfo2", new EspecificacionesPC()
                                                .especificaciones("I9", "HDD",
                                                        "5MB/S", "2.0",
                                                        "Windows
11","serviceInfo2"));
            System.out.println("Servidor dos
activo"+InetAddress.getLocalHost().getHostName());
        }
        catch(Exception ex)
        {
            System.out.println(ex.getMessage());
        }
    }
}
```

Servidor tres

```
package com.mycompany.ejemploagentemovil;
```

```
import java.net.InetAddress;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
/**
 *
 * @author USUARIO
 */
public class ServidorTres {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        try
        {
            Registry rs = LocateRegistry.createRegistry(3333);
            rs.rebind("serviceInfo3", new EspecificacionesPC()
                                                .especificaciones("AMD",
"SSD",
                                                        "10MB/S", "4.0",
                                                        "CentOS","serviceInfo3
"));
            System.out.println("Servidor tres
activo"+InetAddress.getLocalHost().getHostName());
        }
        catch(Exception ex)
        {
            System.out.println(ex.getMessage());
        }
    }
}
```

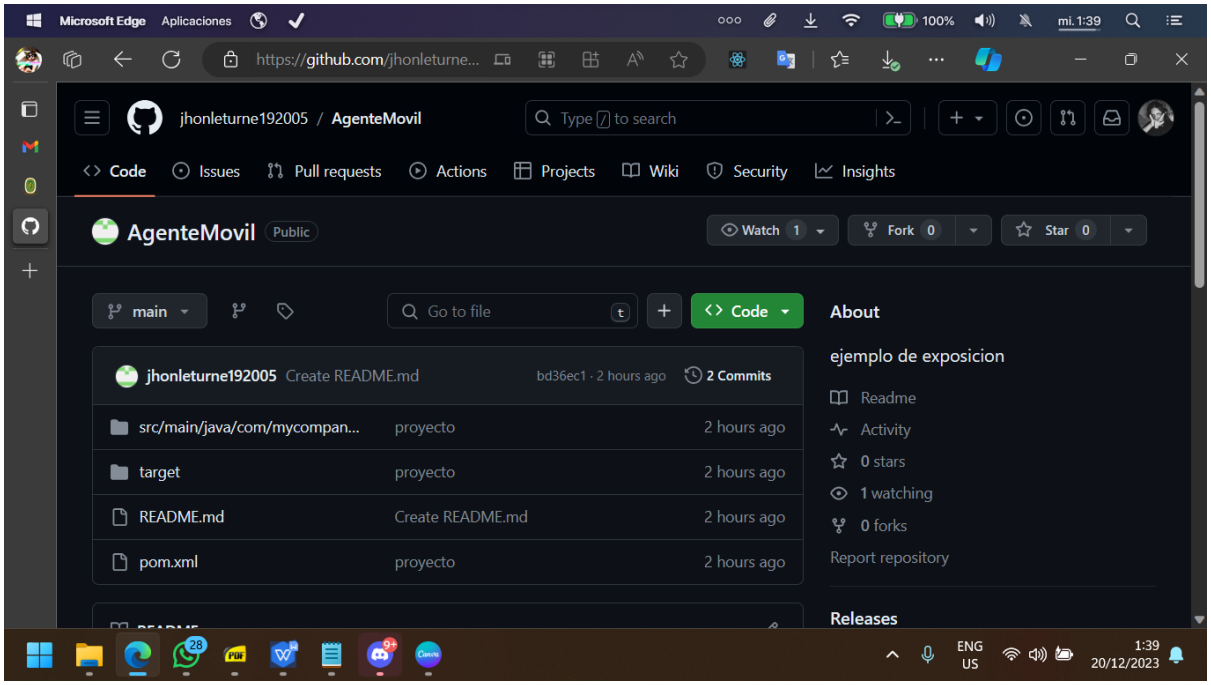
Conexión postgresql

```
package com.mycompany.ejemploagentemovil;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
/**
 *
 * @author USUARIO
 */
public class conexionPostgres
{
```

```
    static String login="postgres";
    static String password ="123";
    static String url="jdbc:postgresql://localhost/agenteMovil";

    private Statement statement;
    private Connection conexion;

    public conexionPostgres() throws ClassNotFoundException, SQLException
    {
        conexion=null;
        statement=null;

        Class.forName("org.postgresql.Driver"); // se hace referencia al
driver
        conexion=DriverManager.getConnection(url,login,password);


        if(conexion!=null)
        {
            statement = conexion.createStatement();
            System.out.println("Conexion a la base de datos "+url+"
......OK");
        }
    }


    public long query(Entity epc) throws Exception
    {
        return statement.executeUpdate("insert into server_central_info
(procesador,tipo_disco,velocidad_red,v_tarj_wifi,sistema_opertivo,servidor)
values
('"+epc.getProcesador()+"','"+epc.getTipo_disco()+"','"+epc.getVelocidad_red()
+"','"+epc.getVersion_tarjeta_wifi()+"','"+epc.getSistema_operativo()+"','"+ep
c.getServidor()+"')");
    }


    public ResultSet mostrar_inst() throws Exception
    {
        return statement.executeQuery("select * from server_central_info");
    }
}
```

Repositorio de GITHUB

jhonleturne192005/AgenteMovil: ejemplo de exposicion (github.com)



5. Conclusión

La tecnología de agentes móviles puede ofrecer un nuevo paradigma para la comunicación a través de canales de red heterogéneos. Se han propuesto e identificado diversas ventajas de utilizar paradigmas de cómputo de agentes móviles.

Estas ventajas incluyen superar la latencia de la red, reducir la carga de la red, ejecutar de manera asíncrona y autónoma, adaptarse dinámicamente, operar en entornos heterogéneos y tener un comportamiento robusto y tolerante a fallos. Sin embargo, los problemas de seguridad y estandarización aún representan un obstáculo significativo.

La tecnología de agentes móviles es un paradigma que implica la migración de un agente de un anfitrión a otro para realizar una tarea en nombre del usuario, suspendiendo la ejecución en una ubicación y reanudándola en otra desde donde se detuvo. Se han presentado diversos beneficios para el uso de la tecnología de agentes móviles, como la reducción del ancho de banda, la disminución de la latencia de red, la ejecución autónoma, la tolerancia a fallos, entre otros.

A pesar de estos beneficios, la adopción de la tecnología está disminuyendo debido a obstáculos relacionados con el lenguaje de implementación, la coordinación, la estandarización, la seguridad y los intereses socioeconómicos. La investigación orientada a abordar estos desafíos de la tecnología de agentes móviles es, por lo tanto, de suma importancia para fomentar una alta adopción de esta tecnología.

La tecnología de agentes móviles presenta un conjunto de desafíos y amenazas en términos de seguridad que deben abordarse para su adopción generalizada. Los ataques pueden manifestarse en diversas formas, como ataques de denegación de servicio y alteraciones, tanto desde el punto de vista del host como del agente móvil.

La seguridad del host implica protegerlo de agentes no autorizados que puedan reclamar la identidad de otros agentes, realizar ataques de denegación de servicio o interferir directamente con la funcionalidad del agente. Por otro lado, la seguridad del agente móvil se refiere a salvaguardarlo de amenazas, ataques de denegación de servicio y alteraciones perpetradas por otros agentes.

Para abordar estos desafíos de seguridad, se necesitan mecanismos eficientes de control de acceso, autenticación y políticas de comunicación entre agentes y plataformas. La investigación continua en esta área es crucial para desarrollar soluciones robustas que fortalezcan la seguridad de la tecnología de agentes móviles. Además, la conciencia sobre las implicaciones de seguridad y la promoción de prácticas seguras en el desarrollo y la implementación de agentes móviles son fundamentales para mitigar riesgos y fomentar la aceptación generalizada de esta tecnología innovadora en sistemas distribuidos.

La conclusión principal que se puede extraer de nuestros hallazgos e investigaciones es que la tecnología de agentes móviles tiene el potencial de mejorar el rendimiento de las redes y de los programas que adoptan agentes móviles. Debido a su naturaleza como una tecnología futurista desde la perspectiva del entorno de programación, aún se requiere mucho trabajo antes de que el programador promedio pueda construir aplicaciones basadas en el paradigma de tecnología de agentes móviles de manera sencilla.

6. Bibliografía

[1] S. Ichiro, “Mobile Agents”, 2010. Accedido el 18 de diciembre de 2023. [En línea]. Disponible: https://research.nii.ac.jp/~ichiro/papers/satoh-mobile-agent.pdf

[2] R. Gray, D. Kotz, S. Nog, D. Rus y G. Cybenko, “Mobile Agents: The Next Generation in Distributed Computing”, p. 17. Accedido el 18 de diciembre de 2023. [En línea]. Disponible: https://doi.org/10.1109/aispas.1997.581620

[3] Sistemas Distribuidos. Cuajimalpa: UNA, 2015. Accedido el 19 de diciembre de 2023. [En línea]. Disponible: http://ilitia.cua.uam.mx:8080/jspui/bitstream/123456789/173/1/X56.pdf

[4] M. van Steen y A. S. Tanenbaum, Distributed Systems, 3a ed. Pearson Educ., Inc, 2018. [En línea]. Disponible: http://www.dgma.donetsk.ua/docs/kafedry/avp/metod/van%20Steen%20%20Distributed%20Systems.pdf

[5] A. R. Valdez Alvarado, “Agentes Móviles”, Agentes Movil., p. 6, 2008. [En línea]. Disponible: https://www.researchgate.net/publication/338491991_Agentes_Moviles

[6] J. MAASSEN et al., “Efficient Java RMI for Parallel Programming”, vol. 23, n.º 6, pp. 747–775, 2001. Accedido el 18 de diciembre de 2023. [En línea]. Disponible: https://webster.cs.uga.edu/~maria/pads/papers/p747-maassen.pdf

[7] P. Lezama, “Coleccion de Tesis Digitales”, Univ. Am. Puebla, 2013. [En línea]. Disponible: http://catarina.udlap.mx/u_dl_a/tales/documentos/msp/perez_l_cv/capitulo2.pdf

[8] M. Higashino. “Application of mobile agent technology to microservice architecture | Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services”. ACM Other conferences. Accedido el 20 de diciembre de 2023. [En línea]. Disponible: https://dl.acm.org/doi/10.1145/3151759.3151840

[9] J. Pavon Maestras, “Agentes Móviles”, 5 de diciembre de 2006. [En línea]. Disponible: https://www.fdi.ucm.es/profesor/jpavon/doctorado/AgentesMoviles.pdf

[10] A. Aneiba, “MOBILE AGENTS TECHNOLOGY AND MOBILITY”, 2002. [En línea]. Disponible: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=95d4761f59930977450760ca37f82b1952bfd6ea

[11] E. Ebietomere y G. Ekuobase, “Issues on Mobile Agent Technology Adoption”, 2014. [En línea]. Disponible: https://afrjcict.net/wp-content/uploads/2017/08/vol-7-no-1-march-2014.pdf

[12] M. S. Greenberg, J. C. Byington, T. Holding y D. G. Harper, “Mobile Agents and Security”, 1998. [En línea]. Disponible: https://sci-hub.se/https://doi.org/10.1109/35.689634

[13] H. Idrissi, A. Revel y E. M. Souidi. “Security of Mobile Agent Platforms using RBAC based on Dynamic Role Assignment”. [En línea]. Disponible: https://www.researchgate.net/publication/302594887_Security_of_Mobile_Agent_Platforms_using_RBAC_based_on_Dynamic_Role_Assignment