ANÁLISIS DE LA COMUNICACIÓN PROTOCOLOS EN SISTEMAS MULTIAGENTE

DRASKO, B. y RAKIC, K.

Resumen: El uso de Sistemas Multiagente (MAS) en diferentes campos de implementación presenta desafíos. Algunos sistemas son generalmente más complejos, mientras que otros no requieren una infraestructura robusta. En ambos casos, la elección del protocolo de comunicación MAS adecuado es esencial. Este artículo busca enumerar y explicar los protocolos clave, como FIPA-ACL, KQML, MQTT, AMQP, WebRTC y CoAP. Se presentará un análisis y una comparación entre protocolos, mostrando sus fortalezas y debilidades. Por ejemplo, FIPA-ACL y KQML ofrecen flexibilidad en la coordinación de agentes, mientras que protocolos ligeros como MQTT y CoAP son ideales para sistemas con recursos limitados, como el IoT. Además, el artículo analiza las ventajas y desventajas en términos de rendimiento y seguridad al utilizar protocolos punto a punto como WebRTC y sistemas de mensajería robustos como RabbitMQ. El artículo proporcionará directrices para seleccionar el protocolo adecuado según los requisitos del sistema, considerando la eficiencia y la confiabilidad en aplicaciones MAS.

Palabras clave: MAS, FIPA-ACL, KQML, MQTT, AMQP

Datos de los autores: Dipl.-Ing. Drasko, B[oris]*; Prof. Adjunto Dr. Sc. Rakic, K[resimir]*,
* Universidad de Mostar, Trg hrvatskih velikana 1, 88000, Mostar, Bosnia y Herzegovina,
boris.drasko@sum.ba, kresimir.rakic@fsre.sum.ba

Esta publicación debe referenciarse como: Drasko, B[oris] & Rakic, K[resimir] (2024).
Análisis de protocolos de comunicación en sistemas multiagente, capítulo 08 en DAAAM International Scientific Book 2024, págs. 103-116, B. Katalinic (Ed.), publicado por DAAAM International, ISBN 978-3-902734-42-6, ISSN 1726-9687, Viena, Austria

DOI: 10.2507/daaam.scibook.2024.08

1. Introducción

Al diseñar un MAS, surge la pregunta de qué protocolo utilizar considerando diversas circunstancias, como la complejidad y robustez del sistema y los recursos disponibles. El propósito de este artículo es proporcionar un análisis detallado de los protocolos multiagente utilizados en la comunicación entre agentes para que los profesionales puedan evaluar y comparar fácilmente diferentes protocolos y, en consecuencia, decidir cuál se adapta mejor a sus necesidades al diseñar un MAS.

Cada agente en MAS tiene un objetivo específico y una tarea asignada. Para que MAS funcione eficientemente, los agentes necesitan intercambiar información, coordinar acciones y, en última instancia, tomar decisiones basadas en la información recopilada para alcanzar los objetivos colectivos establecidos. En resumen, los agentes necesitan una forma de comunicarse entre sí, lo que convierte la comunicación en un componente fundamental de MAS. A lo largo de los años se han desarrollado protocolos de comunicación para MAS, cada uno diseñado para satisfacer requisitos específicos en diversos entornos operativos.

Históricamente, el Lenguaje de Comunicación de Agentes (LCA), como FIPA-LCA, fue desarrollado por la Fundación para Agentes Físicos Inteligentes (FIPA), una organización de estándares creada para promover la interoperabilidad entre sistemas basados en agentes. Es uno de los protocolos más utilizados para la comunicación de agentes. Admite acciones comunicativas como solicitar, informar y proponer, facilitando así una interacción flexible y compleja entre agentes (FIPA, 1997).

Otro ACL desarrollado a principios de la década de 1990 por el investigador Tim Finin y su equipo es KQML (Lenguaje de Consulta y Manipulación de Conocimiento). Fue diseñado para compartir conocimiento y consultas, pero fue reemplazado por FIPA-ACL debido a su adopción más amplia y una semántica más clara (Finin et al., 1994).

Algunos de los protocolos que no son ACL que se utilizan ampliamente para la comunicación de agentes son MQTT (Transporte de telemetría de colas de mensajes), AMQP (Protocolo avanzado de colas de mensajes), WebRTC (Comunicación web en tiempo real) y CoAP (Protocolo de aplicación restringida).

Es importante señalar que la diferencia entre las ACL y las no ACL es que las ACL se desarrollaron principalmente con un enfoque en la comunicación entre agentes basada en la teoría de los actos de habla (Austin, 1962).

En esencia, significa que estos protocolos se basan en la interacción humana y pueden manejar interacciones complejas como acuerdos, negociaciones y delegación de tareas entre agentes (Rygallo et al., 2009).

Por otro lado, las no ACL no son capaces de una comprensión semántica compleja.
pero céntrate en la transmisión de datos y en la mensajería fiable.

2. Protocolos de comunicación clave en MAS

Podemos organizar los protocolos de comunicación en MAS por categorías, de la siguiente manera:

- Lenguajes de comunicación del agente (ACL): FIPA-ACL, KQML.
- Protocolos de publicación-suscripción: MQTT, AMQP.
- Protocolos punto a punto (P2P): WebRTC.
- Protocolos de solicitud-respuesta: HTTP/REST, CoAP.

• Sistemas de memoria compartida: espacios de tuplas.
• Sistemas de mensajería: JADE, RabbitMQ.
• Protocolos de seguridad: TLS, basado en Blockchain.

En las siguientes secciones se presenta un análisis detallado de cada protocolo por categoría, se explica el modo de funcionamiento, sus ventajas y desventajas y se da una comparación de los protocolos existentes.

3. Lenguajes de comunicación del agente

3.1 FIPA ACL

FIPA ACL utiliza un sistema de mensajería performativa para transmitir no solo los datos, sino también la intención subyacente. Un mensaje performativo transmite la intención subyacente en forma de solicitud, información, propuesta, confirmación o rechazo. De este modo, el agente que recibe el mensaje puede reconocer la intención subyacente y tomar decisiones informadas, respondiendo a ella según el contexto de la comunicación.

La estructura del mensaje FIPA-ACL incluye: (FIPA, 2002):
• Performativo. Define el propósito previsto del mensaje (por ejemplo, solicitar, informar, consulta, propuesta).
• Remitente. Identifica al agente que envía el mensaje.
• Receptor. Identifica al destinatario(s) del mensaje.
• Contenido. La información real que se intercambia o el asunto del mensaje.
• Idioma. Especifica el idioma utilizado para expresar el contenido (por ejemplo, Prolog, XML, …).
• Ontología. Define el vocabulario y el dominio del discurso que ambos agentes deben comprender para una comunicación significativa.
• Protocolo. Especifica el protocolo de interacción para gestionar la secuencia de mensajes. intercambios (por ejemplo, un protocolo de negociación o subasta).
• ID de conversación. Permite a los agentes identificar y rastrear una conversación específica.
• Responder con. Se utiliza cuando el remitente espera una respuesta al mensaje.
• En respuesta a. Proporciona una referencia al mensaje original al enviar una respuesta.
• Responder por. Una marca de tiempo que indica cuándo se espera una respuesta.

FIPA-ACL se puede implementar con diversos lenguajes de programación, como Java, Python, C++ y Prolog. Se utiliza especialmente en sistemas complejos distribuidos multiagente, donde los agentes autónomos necesitan intercambiar información semánticamente rica y estructurada para coordinar, negociar o colaborar dinámicamente, como en ciudades inteligentes, la gestión de la cadena de suministro o la robótica autónoma.

Estas son las principales desventajas de utilizar FIPA ACL (Bordini et al., 2005):

• Complejidad. El formato de mensaje estructurado y los performativos de FIPA ACL pueden añadir complejidad al desarrollo de agentes, dificultando su implementación, especialmente en sistemas simples.

• Sobrecarga. La abstracción y la riqueza de la comunicación en FIPA ACL pueden generar sobrecarga, lo cual podría no ser adecuado para entornos en tiempo real o con recursos limitados que requieren una comunicación rápida y de baja latencia.

Adopción limitada. A pesar de ser un estándar, FIPA ACL no se ha adoptado ampliamente en muchas industrias, lo que limita la disponibilidad de soporte, herramientas y recursos comunitarios para los desarrolladores.

Problemas de interoperabilidad. Si bien FIPA ACL busca promover la interoperabilidad entre agentes, las diferencias en la ontología, el lenguaje o las plataformas de los agentes pueden generar problemas de comunicación si no se gestionan con cuidado.

• Rendimiento. El protocolo de comunicación de alto nivel de FIPA ACL puede generar ineficiencias en sistemas que requieren un alto rendimiento y un retraso mínimo, como las operaciones financieras en tiempo real o la automatización industrial.

3.2 KQML

Un mensaje KQML consta de (Finin et.al., 1994):

• Performativo. Indica la intención comunicativa (p. ej., preguntar, decir, lograr).
• Remitente. El agente que envía el mensaje.
• Receptor. El/los destinatario(s) previsto(s).
• Contenido. La información o consulta que se comunica.
• Idioma. Especifica el idioma utilizado en el contenido.
• Ontología. Define el vocabulario para la comprensión compartida.

Las características principales de KQML incluyen:

• Envoltura de mensajes. KQML permite una envoltura de mensajes flexible para una variedad de actos comunicativos.

• Agnosticismo de contenido. Separa el contenido del mensaje de su estructura, lo que permite una variedad de... intercambios de datos.

Algunas de las ventajas de utilizar KQML son:

• Simplicidad. Más fácil de implementar que lenguajes de comunicación más complejos como

Ligamento cruzado anterior FIPA.

• Flexibilidad. Admite diversos idiomas de contenido, lo que lo hace versátil para diferentes dominios.

• Expresividad. Abarca una amplia gama de actos comunicativos (p. ej., preguntar, informar, dar órdenes).

El uso de KQML también conlleva ciertas desventajas:

• Falta de semántica. KQML no aplica una semántica sólida, lo que puede causar inconsistencias de interpretación.

• Sin protocolos de interacción integrados. A diferencia de FIPA ACL, carece de protocolos predefinidos. para interacciones de agentes.

Adopción limitada. Su uso ha disminuido en favor de alternativas más estructuradas . objeto FIPA ACL.

3.3 Comparación de FIPA ACL y KQML

Si bien KQML es más simple y flexible, FIPA ACL ofrece una comunicación más estructurada con protocolos de interacción integrados y una semántica más robusta. KQML es adecuado para necesidades de comunicación ligeras, mientras que FIPA ACL es mejor para entornos multiagente más complejos.

4. Protocolos de publicación-suscripción

Los protocolos de publicación-suscripción (Pub-Sub) se utilizan para una distribución de mensajes eficiente y escalable. Dos protocolos de Pub-Sub más comunes son MQTT y AMQP.

4.1 Transporte de telemetría de colas de mensajes

El Transporte de Telemetría de Colas de Mensajes (MQTT) es un protocolo Pub-Sub ligero, desarrollado para redes de bajo ancho de banda, alta latencia y poco fiables, comúnmente utilizado en aplicaciones de IoT (Banks y Gupta, 2014). MQTT utiliza un intermediario central que recibe mensajes de los publicadores y los reenvía a los suscriptores según los temas. El publicador y el suscriptor están completamente desacoplados, lo que permite una gestión flexible.
y comunicación dinámica.

MQTT admite tres niveles de calidad de servicio (QoS):

- QoS 0 (Máximo una vez). El mensaje se envía una vez y puede perderse si falla la red.
- QoS 1 (al menos una vez). Se garantiza la entrega del mensaje, pero puede ser...
  duplicado.
- QoS 2 (Exactamente una vez). Garantiza que el mensaje se reciba exactamente una vez, lo que lo hace ideal para la transmisión de datos críticos.

Las ventajas de utilizar MQTT incluyen:

- Bajo consumo de recursos. Ideal para dispositivos con recursos limitados.
- Fácil de implementar. El protocolo es fácil de integrar en sistemas con limitaciones
  capacidades.
- Escalabilidad. Admite implementaciones a gran escala con un consumo mínimo de recursos.

Las desventajas de utilizar MQTT son:

- No hay persistencia de mensajes integrada. Los mensajes pueden perderse si el agente falla.
- Seguridad básica. MQTT se basa en protocolos externos (por ejemplo, TLS) para funciones de seguridad.

4.2 Protocolo de cola de mensajes avanzado

El Protocolo de cola de mensajes avanzado (AMQP) es un protocolo de mensajería robusto diseñado para sistemas de nivel empresarial que requieren alta confiabilidad y funciones avanzadas.
A diferencia de MQTT, AMQP proporciona patrones de mensajería más complejos y garantiza la entrega de mensajes (Ayanoglu et al., 2017).

AMQP utiliza un intermediario para enrutar mensajes entre publicadores y suscriptores, pero ofrece patrones de intercambio de mensajes más sofisticados, como colas, claves de enrutamiento e intercambios.

AMQP permite la puesta en cola de mensajes, donde el intermediario puede almacenar los mensajes y entregarlos más tarde, lo que garantiza que no haya pérdida de datos.

Hay dos garantías de entrega:

• Al menos una vez. Garantiza que el mensaje se entregará, aunque se pueden encontrar duplicados.
posible.

• Exactamente una vez. Garantiza que cada mensaje se envíe exactamente una vez sin...
duplicación.

Las ventajas de utilizar AMQP incluyen:

• Confiabilidad. Ofrece mecanismos robustos para garantizar la entrega de mensajes.

• Funciones avanzadas. Admite múltiples patrones de enrutamiento de mensajes, incluyendo temas.
Comunicación basada y directa.

• Seguridad. AMQP incluye mecanismos integrados para la comunicación segura.

Las desventajas de utilizar AMQP son:

• Peso pesado. AMQP consume muchos recursos, por lo que no es adecuado para entornos ligeros o con recursos limitados.

• Complejidad. La implementación puede ser más desafiante debido a su amplia funcionalidad.
colocar.

4.3 Comparación de MQTT y AMQP

La comparación entre los dos protocolos discutidos anteriormente se muestra en la Tabla 1.

Característica	MQTT	AMQP
Caso de uso	IoT, entornos de bajo ancho de banda	Aplicaciones de nivel empresarial
Arriba	Bajo	Alto
Calidad de servicio	0, 1, 2 (Básico)	Al menos una vez, exactamente una vez
Seguridad	Externo (TLS)	Mecanismos incorporados
Persistencia del mensaje	No garantizado	Garantizado mediante colas

Tabla 1. Comparación de MQTT y AMQP

5. Protocolos de solicitud-respuesta

Estos protocolos son esenciales para las interacciones cliente-servidor. Dos ejemplos destacados son HTTP/REST y CoAP. Sus características se explican a continuación.

5.1. Protocolo de transferencia de hipertexto / Transferencia de estado representacional

El Protocolo de Transferencia de Hipertexto (HTTP) es el protocolo para la comunicación web, y la Transferencia de Estado Representacional (REST) es un estilo arquitectónico que se basa en él. Los servicios RESTful utilizan métodos HTTP como GET, POST, PUT y DELETE para la comunicación (Fielding, 2000).

REST no tiene estado, lo que significa que cada solicitud del cliente al servidor es independiente y contiene toda la información necesaria para su procesamiento. REST se usa ampliamente en API web, transfiriendo datos generalmente en formato JSON o XML.

Las ventajas de utilizar HTTP/REST incluyen:

• Escalabilidad. La falta de estado permite un fácil escalamiento horizontal.

• Simplicidad. Utiliza el protocolo HTTP establecido, lo que facilita su implementación.

Las desventajas de usar HTTP/REST son:

• Gastos generales. Los encabezados HTTP pueden ser grandes, lo que los hace ineficientes para entornos restringidos. entornos.

• Latencia. La alta sobrecarga de la red y la falta de estado pueden generar retrasos, especialmente en aplicaciones en tiempo real.

5.2 Protocolo de aplicación restringida

El Protocolo de Aplicación Restringida (CoAP) está diseñado para dispositivos y redes con restricciones, como las de las aplicaciones del IoT. Opera sobre UDP, lo que garantiza una menor sobrecarga que HTTP (Shelby, 2014). CoAP utiliza un modelo de solicitud-respuesta similar a HTTP, pero está optimizado para dispositivos con restricciones y de bajo consumo. Admite comunicación unicast y multicast.

Las ventajas de utilizar CoAP incluyen:

• Eficiencia. Utiliza menos ancho de banda y energía, con mensajes de menor tamaño.

• Interoperabilidad. CoAP se puede mapear fácilmente a HTTP, lo que facilita la integración con sistemas web.

Las desventajas de utilizar CoAP son:

• Conjunto de funciones limitado. CoAP carece de la robustez y las características de HTTP, como la integración en seguridad.

• Confiabilidad. Al estar basado en UDP, puede sufrir pérdida de paquetes, aunque Los mecanismos de retransmisión ayudan a mitigar esto.

5.3 Comparación de MQTT y AMQP

La comparación entre los dos protocolos discutidos anteriormente se muestra en la Tabla 2.

Característica	HTTP/REST	CoAP
Protocolo	Basado en TCP	Basado en UDP
Arriba	Alto	Bajo
Tamaño del mensaje	Más grande (JSON/XML)	Más pequeño (binario)
Fiabilidad	TCP garantiza la confiabilidad	UDP con retransmisión

Tabla 2. Comparación de HTTP/REST y CoAP

6. Sistemas de memoria compartida

Los sistemas de memoria compartida en MAS permiten a los agentes comunicarse escribiendo y leyendo datos de una memoria común. Los espacios de tuplas, originados en Linda, un lenguaje de coordinación desarrollado por David Gelernter, proporcionan un modelo de memoria compartida que facilita esta interacción mediante el uso de tuplas (listas ordenadas de elementos).

En los sistemas de memoria compartida dentro de sistemas multiagente (MAS), los agentes no se envían mensajes directamente, sino que interactúan a través de un espacio de memoria común. Los espacios de tuplas, que funcionan como memoria compartida, permiten a los agentes escribir, leer y tomar tuplas (conjuntos ordenados de datos), lo que facilita la comunicación indirecta. Este enfoque permite interacciones desacopladas, ya que los agentes no necesitan conocer la existencia o el estado de los demás, lo que resulta en un mecanismo de coordinación más flexible y escalable.

6.1 Concepto de espacios de tuplas

En un espacio de tuplas, los agentes realizan tres acciones principales (Gelernter, 1985):

- Escribir (fuera). Insertar una tupla en el espacio.
- Leer (rd). Recupera una tupla sin eliminarla.
- Tomar (en). Recuperar y eliminar una tupla del espacio.

Las ventajas de utilizar espacios de tuplas en sistemas distribuidos incluyen:

- Desacoplamiento. Los espacios de tuplas desacoplan a los agentes tanto en el tiempo como en el espacio, lo que hace que las interacciones sean flexibles y asincrónicas.
- Coordinación. Útil para coordinar agentes distribuidos sin intervención directa. comunicación.

Las desventajas de utilizar espacios de tuplas en sistemas distribuidos son:

- Escalabilidad. Gestionar grandes cantidades de tuplas puede resultar ineficiente.

Riesgos de seguridad. El acceso compartido puede exponer el sistema a vulnerabilidades de seguridad.

6.2 Uso en sistemas distribuidos

Los espacios de tuplas son muy adecuados para la computación distribuida, donde los agentes trabajan de forma asíncrona. El modelo basado en datos permite que los agentes interactúen mediante las tuplas compartidas, en lugar de mediante mensajes directos (Carriero y Gelernter, 1989).

Ventajas de utilizar espacios de tuplas en sistemas distribuidos:

- Interacción asincrónica. Sin necesidad de comunicación en tiempo real entre agentes.

Tolerancia a fallos. Los datos permanecen accesibles hasta su consumo.

Desventajas de utilizar espacios de tuplas en sistemas distribuidos:

- Retrasos en la recuperación. La búsqueda de tuplas específicas en espacios grandes puede ralentizar actuación.
- Consistencia. Garantizar la consistencia de los datos en entornos dinámicos es un desafío.

7. Sistemas de mensajería

Los sistemas de mensajería facilitan la comunicación entre agentes. Dos sistemas de mensajería destacados son JADE y RabbitMQ, cada uno con diferentes necesidades de comunicación entre agentes y distribución de mensajes.

7.1 Marco de desarrollo del agente Java

Java Agent Development Framework (JADE) es un marco de software que simplifica el desarrollo de sistemas multiagente al proporcionar una plataforma para la comunicación entre agentes mediante una arquitectura basada en mensajes. Cumple con las especificaciones, garantizando así la interoperabilidad entre diferentes agentes (Bellifemine et al., 2007).

Los agentes de JADE se comunican utilizando el lenguaje de comunicación del agente (ACL). Cada mensaje consta de un performativo (el tipo de acto comunicativo), un emisor, un receptor y un contenido.

Las ventajas de utilizar JADE incluyen:

Cumplimiento de la FIPA . Garantiza la estandarización y la interoperabilidad entre diferentes implementaciones de MAS.

- Extensibilidad. El diseño modular permite a los desarrolladores añadir fácilmente nuevas funcionalidades o comportamientos de los agentes.
- Tolerancia a fallos. Los agentes JADE se pueden distribuir en múltiples plataformas. garantizar la resiliencia del sistema.

Las desventajas de utilizar JADE son:

- Gastos generales. El cumplimiento de JADE con los estándares FIPA y su complejo protocolo de mensajería pueden generar gastos generales, especialmente para sistemas livianos.
- Rendimiento. JADE puede no ser tan eficiente en entornos de alto rendimiento donde la entrega de mensajes debe ser rápida.

7.2 RabbitMQ

RabbitMQ es un agente de mensajes de código abierto que implementa el Protocolo de Cola de Mensajes Avanzado (AMQP). Permite que las aplicaciones se comuniquen de forma asíncrona mediante colas de mensajes, lo que lo hace adecuado tanto para sistemas basados en agentes como para arquitecturas distribuidas generales (Alvaro y Videla, 2012).

RabbitMQ facilita la comunicación entre productores (que envían mensajes) y consumidores (que los reciben). Los mensajes se envían a los exchanges, que los enrutan a colas según las reglas de enrutamiento. RabbitMQ garantiza la entrega fiable de mensajes mediante mecanismos de acuse de recibo y admite la persistencia de mensajes para una mayor durabilidad.

Las ventajas de utilizar RabbitMQ incluyen:

- Alto rendimiento. RabbitMQ está diseñado para entornos de alto rendimiento y Admite una gran cantidad de mensajes simultáneos.
- Confiabilidad. Con funciones como acuse de recibo de mensajes, garantías de entrega y persistencia, RabbitMQ garantiza que los mensajes no se pierdan.

• Flexibilidad. Admite múltiples patrones de mensajería, como directo, de abanico y temático.
enrutamiento basado en

Las desventajas de usar RabbitMQ son:

• Complejidad. RabbitMQ puede ser más complejo de configurar, especialmente para sistemas que no requieren sus funciones avanzadas.
• Gastos generales. La necesidad de reconocimiento y enrutamiento de mensajes a través de centrales puede generar latencia en aplicaciones sensibles al tiempo.

7.3 Comparación de JADE y RabbitMQ

La comparación entre los dos protocolos discutidos anteriormente se muestra en la Tabla 3.

Desarrollo de JADE MAS, comunicación con agentes		RabbitMQ
Caso de uso		Mensajería general, sistemas distribuidos
Protocolo	FIPA-ACL	AMQP
Escalabilidad	Moderado	Alto
Fiabilidad	Tolerancia a fallos entre agentes	Reconocimiento de mensajes, persistencia
Complejidad	Alto (cumplimiento de FIPA)	Moderado (Funciones avanzadas)

Tabla 3. Comparación de JADE y RabbitMQ

8. Protocolos de seguridad

Los protocolos de seguridad son esenciales para garantizar la seguridad de las comunicaciones, la integridad de los datos y la fiabilidad del sistema. Dos protocolos ampliamente utilizados son la seguridad de la capa de transporte y la seguridad basada en blockchain.

8.1 Seguridad de la capa de transporte

La Seguridad de la Capa de Transporte (TLS) es un protocolo criptográfico diseñado para proporcionar comunicaciones seguras en una red. Evolucionó a partir de la Capa de Sockets Seguros (SSL) y se utiliza ampliamente para proteger el tráfico web, el correo electrónico y otras comunicaciones de Internet (Rescorla, 2001).

TLS utiliza una combinación de cifrado simétrico y asimétrico para garantizar la confidencialidad e integridad de los datos. Una sesión TLS comienza con un protocolo de enlace donde el servidor y el cliente intercambian claves y se autentican mutuamente mediante certificados digitales. Posteriormente, toda la comunicación se cifra mediante claves simétricas.

TLS también proporciona integridad de mensajes mediante el uso de funciones hash, lo que garantiza que los datos no se alteren durante la transmisión. El protocolo admite múltiples algoritmos de cifrado, lo que permite a clientes y servidores negociar el conjunto de cifrado más robusto disponible según sus capacidades. Además, TLS incorpora la confidencialidad directa perfecta (PFS), lo que garantiza que, incluso si se comprometen las claves a largo plazo, las comunicaciones anteriores se mantengan seguras mediante el uso de claves de sesión efímeras para cada conexión.

Algunas características clave de TLS son:

• Confidencialidad. TLS encripta los datos para evitar el acceso no autorizado.

• Integridad. La integridad de los mensajes se garantiza mediante hash criptográfico (p. ej., HMAC).

• Autenticación. Utiliza certificados X.509 para autenticar la identidad de las partes que se comunican.

Las ventajas de utilizar TLS incluyen:

• Ampliamente adoptado. TLS se utiliza ampliamente en la web, lo que lo convierte en un estándar para proteger las comunicaciones.

• Seguridad de extremo a extremo. TLS proporciona cifrado del cliente al servidor, protegiendo los datos en tránsito.

Las desventajas de utilizar TLS son:

• Consume muchos recursos. El protocolo de enlace TLS y los mecanismos de cifrado pueden introducir latencia y sobrecarga de recursos, especialmente en sistemas con recursos limitados.

• Gestión de certificados. TLS requiere una infraestructura de clave pública (PKI) para la gestión de certificados, cuya implementación puede ser compleja.

8.2 Seguridad basada en blockchain

La seguridad basada en blockchain aprovecha la naturaleza descentralizada e inmutable de la tecnología blockchain para proteger las comunicaciones, el almacenamiento de datos y las transacciones. Una blockchain es un libro de contabilidad distribuido que registra datos en bloques, que se vinculan entre sí en una cadena mediante hashes criptográficos (Nakamoto, 2008).

Blockchain se basa en mecanismos de consenso, como Prueba de Trabajo (PoW) o Prueba de Participación (PoS), para validar las transacciones y garantizar la integridad del libro mayor.

Cada participante de la red posee una copia de la cadena de bloques, lo que la hace resistente a manipulaciones.

La seguridad basada en blockchain utiliza la naturaleza descentralizada de la tecnología blockchain para garantizar que ninguna entidad controle el sistema, mejorando la resiliencia contra los ataques.

Cada bloque de la blockchain contiene un hash criptográfico del bloque anterior, lo que crea una cadena segura prácticamente imposible de alterar sin el consenso de toda la red. Mecanismos de consenso como la Prueba de Trabajo (PoW) o la Prueba de Participación (PoS) validan las transacciones, garantizando que solo las acciones legítimas se registren en el libro contable. Al distribuir el libro contable entre todos los participantes, la tecnología blockchain proporciona transparencia y resistencia a la manipulación, lo que la hace altamente segura para diversas aplicaciones.

Algunas características clave de la seguridad basada en Blochain son:

• Descentralización. Blockchain elimina la necesidad de una autoridad central, distribuyendo el control a través de la red.

• Inmutabilidad. Una vez que los datos se registran en un bloque, no se pueden alterar sin consenso de la mayoría de la red.

• Transparencia. Todas las transacciones son visibles para los participantes, lo que garantiza la rendición de cuentas.

Las ventajas de utilizar seguridad basada en Blochain incluyen:

• A prueba de manipulaciones. La inmutabilidad de la cadena de bloques garantiza que los datos o las transacciones... No puede ser alterado o falsificado.

• Confianza descentralizada. Blockchain elimina la necesidad de confiar en una única autoridad, distribuyéndola por toda la red.

Las desventajas de utilizar seguridad basada en Blochain son:

• Escalabilidad. La cadena de bloques puede presentar problemas de escalabilidad, ya que la velocidad de las transacciones suele ser más lenta que la de los sistemas centralizados.

• Consumo de energía. Los mecanismos de consenso como PoW consumen mucha energía, lo que hace que la cadena de bloques sea menos sostenible para transacciones de alta frecuencia.

8.3 Comparación de TLS y seguridad basada en blockchain

La comparación entre los dos protocolos discutidos anteriormente se muestra en la Tabla 4.

Característica	TLS	Basado en blockchain Seguridad
Arquitectura	Centralizado, cliente-servidor	Descentralizado, peer to peer
Confidencialidad	Cifrado con claves simétricas y asimétricas	Hashing criptográfico para la integridad de los datos
Escalabilidad	Alto (con infraestructura adecuada)	Moderado (depende del consenso)
Caso de uso clave	Comunicación web e Internet	Asegurar transacciones y datos distribuidos

Tabla 4. Comparación de TLS y seguridad basada en blockchain

9. Conclusión sobre los protocolos de comunicación en MAS

En MAS, los protocolos de comunicación desempeñan un papel fundamental para facilitar la interacción y la coordinación efectivas entre agentes. Estos protocolos garantizan que los agentes puedan intercambiar información de forma fiable y eficiente, independientemente de si operan en entornos centralizados, descentralizados o híbridos.

Los protocolos de solicitud-respuesta como HTTP/REST y CoAP proporcionan marcos estructurados para la comunicación directa entre agentes. REST es ideal para sistemas web escalables y CoAP está diseñado para entornos IoT con recursos limitados. Por otro lado, los protocolos de publicación-suscripción (Pub-Sub) como MQTT y AMQP facilitan la comunicación asíncrona, desacoplando al emisor del receptor, lo que mejora la escalabilidad en sistemas dinámicos a gran escala.

Además, protocolos de seguridad como TLS y sistemas basados en blockchain garantizan la seguridad y la protección de la comunicación entre agentes. TLS ofrece un cifrado punto a punto robusto, ideal para aplicaciones web MAS, mientras que blockchain proporciona un mecanismo descentralizado que garantiza la integridad y la confianza de los datos en entornos distribuidos.

La elección del protocolo de comunicación en MAS depende en gran medida del caso de uso específico, los requisitos del sistema y limitaciones como la escalabilidad, la disponibilidad de recursos y las necesidades de seguridad. A medida que MAS evoluciona, la selección y optimización de estos protocolos seguirá siendo un factor clave para lograr interacciones multiagente eficientes y fiables.

Además de las características específicas de cada protocolo de comunicación, la interoperabilidad y compatibilidad de estos protocolos en entornos MAS heterogéneos también son consideraciones cruciales. Muchas implementaciones de MAS involucran agentes con diversas capacidades que operan en diferentes plataformas y redes.

Por lo tanto, seleccionar protocolos que permitan una integración fluida entre diversos sistemas es esencial para garantizar una comunicación fluida entre agentes. Además, la capacidad de gestionar la latencia de la red, gestionar fallos y mantener un rendimiento constante bajo cargas variables es otro factor importante al evaluar los protocolos de comunicación.

Al equilibrar cuidadosamente estos factores, MAS puede lograr una coordinación sólida, escalable y eficiente en diferentes casos de uso y entornos.

Este artículo enumera los protocolos de comunicación más comunes en el diseño de MAS. Enumera las características principales de cada uno y analiza sus ventajas y desventajas por categoría. Finalmente, compara los protocolos por categoría.

9.1 Investigaciones adicionales sobre los protocolos MAS

Se han logrado avances significativos en el desarrollo de protocolos de comunicación para sistemas multiagente (MAS). Sin embargo, existen varias áreas que requieren mayor exploración. Un área crítica es la escalabilidad. Los protocolos de comunicación MAS actuales suelen presentar limitaciones de escalabilidad debido al aumento significativo del número de agentes, lo que genera cuellos de botella en el rendimiento. La investigación futura debe centrarse en el desarrollo de protocolos que mantengan la eficiencia y la fiabilidad en sistemas a gran escala, especialmente en entornos dinámicos y en tiempo real.

Otra importante línea de investigación es la integración de MAS con tecnologías emergentes como el 5G, la computación de borde y la comunicación cuántica. Estas tecnologías podrían mejorar la velocidad y la fiabilidad de la comunicación entre agentes, abriendo nuevas posibilidades para las aplicaciones de MAS en entornos de alto riesgo como los vehículos autónomos, las ciudades inteligentes y la automatización industrial.

Las preocupaciones sobre seguridad y privacidad también son importantes. A medida que el MAS se vuelve más común en sectores como las finanzas, la salud y la defensa nacional, garantizar la comunicación segura entre agentes es crucial. Los protocolos deben ser capaces de resistir ciberataques y garantizar la privacidad e integridad de los datos, especialmente en sistemas descentralizados donde los agentes pueden operar bajo diferentes jurisdicciones o regulaciones.

Por último, es necesario seguir investigando la adaptabilidad y flexibilidad de los protocolos de MAS. Dado que los MAS suelen operar en entornos impredecibles y cambiantes, los protocolos deben evolucionar y adaptarse dinámicamente a las nuevas condiciones sin intervención humana. (Popirlan y Stefanescu, 2011)

Técnicas como el aprendizaje automático y la inteligencia artificial podrían desempeñar un papel crucial en la creación de sistemas de comunicación más adaptativos que puedan aprender y optimizar las vías de comunicación basándose en datos en tiempo real.

Al abordar estos desafíos, las investigaciones futuras pueden contribuir al desarrollo de protocolos de comunicación MAS más sólidos, eficientes y seguros, garantizando su continua relevancia y eficacia en diversos dominios.

10. Referencias

Álvaro, A. y Videla, G. (2012). RabbitMQ en acción. Publicaciones Manning. https://www.manning.com/books/rabbitmq-in-action

Austin, JL (1962). Cómo hacer cosas con palabras. Clarendon Press.

Ayanoglu, E., Aytas, Y y Nahoum, D (2017). Dominando RabbitMQ por Packt Publishing

Banks, A., y Gupta, R. (2014). MQTT versión 3.1.1. Estándar OASIS. https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.pdf

Bellifemine, F., Caire, G. y Greenwood, D. (2007). Desarrollo de sistemas multiagente con JADE. John Wiley & Sons. https://onlinelibrary.wiley.com/doi/libro/10.1002/9780470058411

Bordini, RH, Dastani, M., Dix, J. y Seghrouchni, AEF (2005). "Programación multiagente: Plataformas y aplicaciones". Idiomas, https://link.springer.com/book/10.1007/978-0-387-89299-3

Carriero, N. y Gelernter, D. (1989). Linda en contexto. Comunicaciones de la ACM, https://dl.acm.org/doi/10.1145/63334.63337

Fielding, RT (2000). Estilos arquitectónicos y diseño de arquitecturas de software basadas en redes (Tesis doctoral, Universidad de California, Irvine). https://ics.uci.edu/~fielding/pubs/dissertation/top.htm

Finin, T., Fritzson, R., McKay, D. y McEntire, R. (1994). KQML como lenguaje de comunicación para agentes. En Actas de la 3.ª Conferencia Internacional sobre Gestión de la Información y el Conocimiento, Gaithersburg, MD. DOI: 10.1145/191246.191322

FIPA (2002). "Biblioteca de actos comunicativos del FIPA http://www.fipa.org/ Especificación." specs/fipa00037/SC00037J.html

Gelernter, D. (1985). Comunicación generativa en Linda. ACM Transactions on Programming Languages and Systems, https://dl.acm.org/doi/10.1145/2363.2433

Nakamoto, S. (2008). Bitcoin: Un sistema de efectivo electrónico entre pares. Bitcoin.org.

Popirlan, C y Stefanescu, C (2011), Una solución multiagente para la mejora del centro de contacto, DOI:10.2507/22nd.daaam.proceedings.576

Rescorla, E. (2001). SSL y TLS: Diseño y construcción de sistemas seguros. Addison-Wesley Professional. ISBN-10: 0201615983. ISBN-13: 978-0201615982.

Rygallo, A.; Zloto T. y Wolny, R. (2009). Un modelo gramatical de la Sistema Multiagente, LIBRO CIENTÍFICO INTERNACIONAL DAAAM 2009 DOI:10.1007/11802372_5

Shelby, Z., Hartke, K. y Bormann, C. (2014). El Protocolo de Aplicación Restringida (CoAP). RFC 7252, IETF. https://www.rfc-editor.org/info/rfc7252