Universidad del Valle de Guatemala
Facultad de Ingeniería
Departamento de Ciencias de la Computación
CC3067 Redes

Proyecto 1

Uso de un protocolo existente

1 Antecedentes

Los *Large Language Models* (LLMs) son una de las invenciones más importantes de los últimos años en la Inteligencia Artificial, y su adopción e integración en nuestra vida diaria es un hecho. Sin embargo, los LLMs tienen limitaciones importantes: solo pueden responder sobre su base de conocimiento, y por si solos, no pueden comunicarse con el mundo real. Por ejemplo, un LLM no es capaz de responder cual es el clima actual, y mucho menos, ajustar un termostato doméstico en base a la temperatura.

Aquí es donde entran los chatbots y agentes, que le proveen herramientas a los LLMs para aumentar sus capacidades. Por ejemplo, para responder al clima actual, el chatbot Claude (de Anthropic) utiliza información recolectada de su aplicación (dirección IP) para establecer una ubicación aproximada y luego utiliza una función que le permite buscar en la Web cuál es el clima para dicha ubicación. El problema es que cada empresa (OpenAI, Anthropic, Google, Meta, etc.) define sus propios protocolos de cómo integrar estas herramientas a sus agentes, de modo que un desarrollador que defina una herramienta para ChatGPT (OpenAI), debe migrarla para poderla usar en Claude, es decir, no existe interoperabilidad.

Model Context Protocol, es un protocolo abierto propuesto por Anthropic en noviembre del 2024, que busca ser el estándar para proveer estas herramientas a los agentes y chatbots. Esto ofrece una importante ventaja pues el desarrollo de las herramientas es independiente del LLM (solo se expone cómo debe invocarse y usarse la herramienta). MCP utiliza JSON-RPC (Remote Procedure Call), un protocolo a nivel de la capa de aplicación de los modelos OSI y TCP/IP.

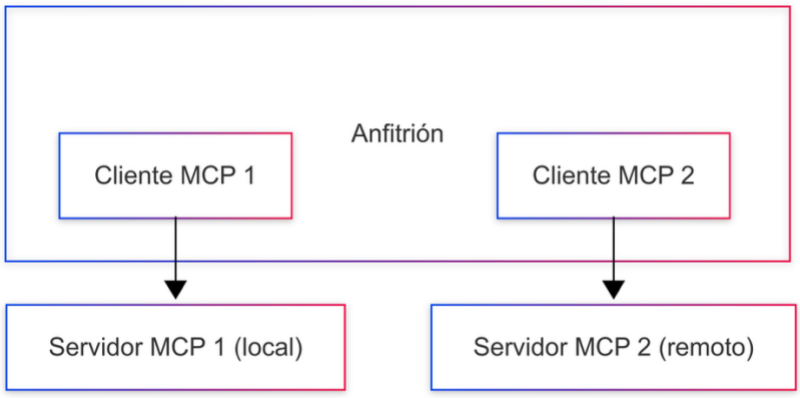
Debido a que los LLMs son capaces de analizar y mantener contexto, son capaces también de coordinar el uso de estas herramientas abriendo miles de escenarios posibles. Por ejemplo, un chatbot con acceso a una herramienta que permita interactuar con sistema de archivos de una computadora, y con acceso a una herramienta que permita interactuar con GitHub, podría ser capaz de clonar un repositorio y realizar modificaciones a los archivos (el LLM no ejecuta las acciones, pero instruye al agente como usar las herramientas, en base a las especificaciones de estas).

Existen tres actores importantes:

- **Servidor**: es la herramienta que ejecuta las acciones. Puede ejecutarse de forma local o remoto.

- **Cliente**: es el componente que mantiene la conexión con el servidor, y obtiene la información sobre como utilizarlo.
- **Anfitrión**: es la aplicación de IA que coordina múltiples clientes. Por ejemplo, Cursor, VS Code, Claude Desktop, ChatGPT, etc., son aplicaciones que pueden hacer uso de diversos servidores MCP a través de los clientes.

El siguiente diagrama (basado en el diagrama de arquitectura de Anthropic) muestra la relación entre los actores:



2 Objetivos del proyecto

- Implementar un protocolo en base a los estándares.
- Comprender el propósito del protocolo MCP.
- Comprender cómo funcionan los servicios del protocolo MCP.
- Implementar servidores MCP locales y remotos
- Comprender como interactuar con un LLM a nivel de la API.

3 Desarrollo

3.1 Características y limitaciones

El proyecto consiste en implementar un chatbot (en consola) que utilice diversos servidores MCP. El chatbot será el anfitrión. Se sugiere el uso de los LLMs de Anthropic, pues proporcionan $5 créditos gratuitos a su API (no es necesario utilizar una tarjeta de débito/crédito), suficiente para el desarrollo de este proyecto. El desarrollo será individual. Se deben implementar como mínimo las siguientes funcionalidades:

Funcionamiento general del chatbot (15%, 5% cada funcionalidad)

1) Conexión con un LLM a nivel de su API (por ejemplo, debe ser capaz de responder preguntas generales sobre su base de entrenamiento)
2) Mantener contexto en una sesión: debe ser capaz de mantener el contexto en una conversación. Por ejemplo, si se le pregunta ¿Quien fue Alan Turing?, y luego, en una segunda pregunta, ¿En que fecha nació?, el chatbot debe entender que la segunda pregunta se refiere a la fecha de nacimiento de Alan Turing.
3) Mantener y mostrar un log sobre todas las interacciones (solicitudes y respuestas) con los servidores MCP.

Servidores MCP – Primera parte (30%, 15% cada funcionalidad)

4) Uso de servidores MCP locales existentes (oficiales): agregar a su chatbot el uso del Filesystem MCP server y Git MCP server (Anthropic). Demostrar un escenario, por ejemplo, solicitar a su chatbot que cree un repositorio, cree un archivo README, lo agregue al repositorio y realice un commit.
5) Crear un servidor MCP que se ejecute de forma local y que sea utilizado por su chatbot. La funcionalidad a implementar es libre, pero no debe ser trivial: por ejemplo, consultar el clima es una funcionalidad trivial y por lo tanto, no será permitida. Analizar una partida de Chess.com utilizando el motor Stockfish e indicar cuál fue el primer error de la partida es una funcionalidad más compleja y permitida. Consultar con su catedrático sobre la función que desea realizar antes de implementarla. Se debe proporcionar la especificación del servidor, como utilizarlo y ejemplos de uso. Se debe subir a su repositorio de GitHub.

Servidores MCP – Segunda Parte (45%, 15% cada funcionalidad)

6) Uso de servidores MCP locales existentes (de otros estudiantes): cada estudiante hará público en su cuenta de GitHub el servidor MCP que implementó en la primera parte, indicando las instrucciones de uso/instalación y la definición del servidor. Se publicarán los accesos a dichos repositorios y cada estudiante deberá elegir dos servidores cualesquiera a implementar en su chatbot. Debe mostrar un escenario(s) del uso individual o conjunto de dichos servidores.
7) Crear un servidor MCP que se ejecute de forma remota. Para ello se debe implementar el servidor en un servicio de nube, como Google Cloude o Cloudfare, etc. La funcionalidad puede ser trivial, siempre que se ejecute de forma remota, y cada estudiante deberá agregar su servidor MCP remoto a su chatbot. Debe mostrar un escenario de su uso.

8) Análisis de la comunicación entre el servidor remoto y el cliente: utilizando WireShark, capturar todas las interacciones entre el anfitrión y el servidor, indicar que mensajes JSON-RPC corresponden a los mensajes de sincronización, cuales a solicitud o petición, y cuales a las respuestas.

Reporte (10%)

9) Para los servidores MCP desarrollados: indicar la especificación, parámetros, endpoints, etc.
10) En base al análisis del punto 8, explicar que sucede a nivel de la capa de enlace, red, transporte y aplicación.
11) Conclusiones y comentario sobre el proyecto

El proyecto debe estar funcionar a nivel de una terminal o linea de comandos. Puede utilizar cualquier lenguaje de programación . Pueden utilizarse librerías y SDKs que faciliten la comunicación con el protocolo MCP, se sugieren los SDKs oficiales de Anthropic.

3.2 Documentación y uso de control de versiones

Cada estudiante debe de llevar su proyecto con un sistema de control de versiones de su preferencia, el cual debe de ser privado y darle acceso a las cuentas de los docentes y auxiliares. El control de versiones debe ser coherente con el desarrollo gradual del proyecto. Por ejemplo, un proyecto con pocos commits no refleja un versionamiento apropiado.

El código debe de estar debidamente documentado, utilizando las mejores prácticas para el lenguaje. Adicional, debe de poseer al menos un documento README.md que describa las características del proyecto, las funcionalidades implementadas y la manera en que un usuario puede instalarlo y usarlo en su computadora (en **inglés**).

No es necesario, pero se motiva que al menos los comentarios para cada *commit* en el proyecto se encuentren también en inglés.

El repositorio donde se publique el servidor MCP local debe ser público, y es un repositorio independiente.

3.3 Presentación del proyecto

Cada estudiante debe de demostrar la funcionalidad de su proyecto y explicar:

- Características implementadas
- Dificultades
- Lecciones aprendidas

Se debe realizar una entrega parcial con la publicación del servidor MCP que cada estudiante defina. Las publicaciones serán compiladas y puestas a disposición de la clase dos semanas antes de la presentación.

4 Evaluación

El proyecto tiene una ponderación de 15 puntos netos, los cuales están distribuidos de la siguiente manera:

- [70%] Funcionamiento del proyecto
- [15%] Documentación y uso de sistema de control de versiones
- [15%] Presentación del proyecto

4.1 Extras

Se otorgará 15% extra al proyecto que implemente el protocolo MCP con el intercambio directo de los elementos en JSON-RPC (sin usar un SDK para MCP), implementando correctamente al menos el 90% de las funcionalidades descritas. **Esta calificación es binaria.**

Se otorgará 15% extra al proyecto que implemente una **User Interface** (UI) a su proyecto, ya sea en la terminal, o bien mediante un chatbot Web. La UI debe considerar los conocimientos adquiridos en el curso de Interacción Humano-Computadora (HCI): psicología del color, organización de los elementos gráficos, usabilidad, etc.

4.2 Rúbrica

Para las funcionalidades se manejarán los siguientes rangos:

- 100%: la funcionalidad está implementada correctamente
- 99% - 75%: la funcionalidad falla en casos muy concretos o eventuales.
- 75% - 50%: la funcionalidad falla constantemente, pero funciona en algunos casos.
- 50% - 0%: la funcionalidad no está implementada, contiene errores en el código o no funciona en ningún escenario.

Para cada rango, se explicará el por qué de la nota obtenida durante la calificación.

La documentación manejará los siguientes rangos:

- Documentación (35%)
    - 100%: El archivo README contiene con detalle todas las seciones solicitadas , los comentarios son utilizados apropiadamente en el código.
    - 99% - 75%: el archivo README contiene las secciones solicitadas, pero no se detallan con exactitud (Ej: faltan algunas instrucciones para instalar o usar el cliente), los comentarios son utilizados de forma apropiada.
    - 75% - 50%: el archivo README existe, pero el nivel de detalle es escaso y no es posible instalar o usar el cliente únicamente con las instrucciones proporcionadas. Los comentarios son genéricos.
    - 50% - 0%: no existe el archivo README o no hay comentarios.
- Control de versiones (75%)
    - 100%: el número de commits es adecuado al desarrollo gradual del proyecto durante las cinco semanas de desarrollo.
    - 99% - 50%: el número de commits no refleja el desarrollo gradual del proyecto, existen commits con varios días de distancia.
    - 50% - 0: existen muy pocos commits, o la mayoría de commits fueron realizados en los días previos a la entrega del proyecto.

La presentación manejará los siguientes rangos:

- 100%: La presentación contiene las funcionalidades realizadas, las dificultades y cómo se resolvieron y las lecciones aprendidas.
- 99% - 50%: la presentación contiene las secciones, pero hay poco detalle visual o verbal.
- 50% - 0%: la presentación no aborda las secciones solicitadas, los detalles son genéricos.

5 Recomendaciones

Se sugiere comprender el funcionamiento de los servicios de MCP y practicar con los distintos tipos de mensajes antes de comenzar el proyecto. Se sugiere el uso de Claude Desktop como anfitrión para probar los servidores MCP, antes de implementarlos en su chatbot. El sitio web de Antrohpic sobre MCP contiene una explicación muy completa de la arquitectura y especificación del protocolo. Además, proporciona ejemplos sobre como crear e instalar servidores y clientes MCP sencillos, así como acceso a decenas de ejemplos utilizando los SDKs.

Leer con detalle la arquitectura MCP y la especificación del protocolo.

Revisar la documentación de cada librería/SDK.

6 Integridad académica

El plagio total o parcial conlleva la anulación del proyecto y la notificación a la Dirección para las sanciones administrativas que correspondan según la gravedad del caso.

- Si se utiliza código desarrollado por terceros como referencias, se deben indicar las fuentes en los comentarios.
- El uso IA generativa están permitidas, siempre y cuando se sigan los lineamientos indicandos en el reglamento de uso de IA de la UVG (disponible en Canvas).

7 Recursos

JSON-RPC: https://www.jsonrpc.org/

MCP Architecture: https://modelcontextprotocol.io/docs/learn/architecture

MCP protocolo definition: https://modelcontextprotocol.io/specification/2025-06-18

MCP servers and SDKs: https://github.com/modelcontextprotocol/servers

Tutorial de implementación de un servidor MCP remoto en la Google Cloud: https://cloud.google.com/blog/topics/developers-practitioners/build-and-deploy-a-remote-mcp-server-to-google-cloud-run-in-under-10-minutes