AGENT2AGENT PROTOCOL

สร้างเครือข่ายเอเจนต์อัจฉริยะแห่งอนาคต

อนุชิต ชโลธร

สำนักพิมพ์ก๊อปวาง

AGENT2AGENT PROTOCOL

สร้างเครือข่ายเอเจนต์อัจฉริยะแห่งอนาคต

อนุชิต ชโลธร

บทนำ

ในช่วงทศวรรษที่ผ่านมา การพัฒนาปัญญาประดิษฐ์ (Artificial Intelligence - AI) ได้ก้าวหน้าอย่างรวดเร็ว โดยเฉพาะในด้านของระบบที่มีความสามารถเฉพาะทาง (specialized agents) ไม่ว่าจะเป็นระบบประมวลผลภาษา ระบบวางแผนอัตโนมัติ หรือระบบแนะนำ (recommender systems) อย่างไรก็ตาม แม้ AI เหล่านี้จะมีความสามารถสูงในการทำงานแต่ละด้าน แต่การเชื่อมโยงและประสานงานระหว่างกันยังคงเป็นความท้าทายที่สำคัญ

แนวคิดในการออกแบบระบบ AI ที่ประกอบด้วยเอเจนต์อิสระหลายตัว (multi-agent systems) ได้รับความสนใจเพิ่มขึ้นอย่างต่อเนื่อง โดยเฉพาะในบริบทของระบบที่ต้องการความสามารถแบบองค์รวม (holistic intelligence) เช่น ระบบช่วยเหลือผู้ใช้ในชีวิตประจำวัน ระบบสนับสนุนการตัดสินใจในองค์กร หรือระบบอัตโนมัติที่ทำงานในสภาพแวดล้อมที่ซับซ้อน การทำให้ AI agents เหล่านี้สามารถ “เข้าใจ” และ “ทำงานร่วมกัน” ได้อย่างมีประสิทธิภาพ จึงกลายเป็นจุดเปลี่ยนสำคัญของการพัฒนาเทคโนโลยี AI ในระดับระบบ

Agent2Agent Protocol (A2A) เป็นมาตรฐานเปิด ออกแบบมาเพื่อตอบโจทย์ข้างต้น โดยมีวัตถุประสงค์เพื่อสร้างรูปแบบการสื่อสารที่เป็นกลาง มีโครงสร้างชัดเจน และสามารถใช้งานร่วมกันได้ระหว่าง AI agents จากระบบ เทคโนโลยี หรือองค์กรที่แตกต่างกัน A2A มุ่งเน้น

การลดความซับซ้อนของการออกแบบระบบ multi-agent ในทางปฏิบัติ พร้อมทั้งให้สามารถนำ AI ที่มีอยู่แล้วมาทำงานร่วมกันได้ โดยไม่จำเป็นต้องพัฒนาขึ้นใหม่ทั้งหมด

หนังสือเล่มนี้มีเป้าหมายเพื่อเป็นแนวทางเชิงปฏิบัติสำหรับนักพัฒนา นักวิจัย และผู้สนใจทั่วไป ที่ต้องการทำความเข้าใจและประยุกต์ใช้ Agent2Agent Protocol ในบริบทต่างๆ ภายในเล่ม ผู้อ่านจะได้เรียนรู้ตั้งแต่แนวคิดพื้นฐาน โครงสร้างของโปรโตคอล รูปแบบการสื่อสาร ไปจนถึงกรณีศึกษาที่แสดงให้เห็นถึงศักยภาพของ A2A ในการสร้างระบบ AI ที่สามารถร่วมมือกันได้อย่างมีประสิทธิภาพ

เราหวังว่าหนังสือเล่มนี้จะเป็นจุดเริ่มต้นสำหรับทุกท่านที่ต้องการก้าวเข้าสู่ยุคของ **AI ที่ทำงานร่วมกันได้อย่างแท้จริง**

สารบัญ

ทำความรู้จัก A2A Protocol

Agent2Agent (A2A) Protocol เป็นมาตรฐานเปิดที่สร้างขึ้นมาเพื่อช่วยให้ AI ต่างๆ คุยกันรู้เรื่องและทำงานร่วมกันได้ง่ายขึ้น

> *หัวใจสำคัญคือ ทำยังไงให้ AI จากคนละทีม คนละเทคโนโลยี หรือแม้แต่คนละบริษัท สามารถคุยกันและทำงานด้วยกันได้*

ทุกวันนี้ AI เก่งเฉพาะด้านมากขึ้นเรื่อยๆ ทำให้เรายิ่งต้องการให้ AI หลายๆ ตัวมาช่วยกันแก้ปัญหา ตัวอย่างเช่น ถ้าเราสั่งให้ AI ผู้ช่วยส่วนตัวของเราวางแผนเที่ยวต่างประเทศ แค่คำสั่งเดียวนี้ อาจต้องใช้ AI หลายตัวมาช่วยกันทำงาน เช่น

- AI จองตั๋วเครื่องบิน
- AI จองโรงแรม
- AI แนะนำและจองทัวร์ท้องถิ่น
- AI ให้ข้อมูลแลกเปลี่ยนเงินตรา
- AI ให้คำแนะนำด้านความปลอดภัย

ถ้าจะให้ AI เหล่านี้ทำงานร่วมกันโดยไม่มีมาตรฐานกลางในการสื่อสาร อาจจะกลายเป็นเรื่องยุ่งยากมากสำหรับนักพัฒนา เพราะ

ต้องเชื่อมต่อ AI แต่ละตัวเข้าด้วยกันโดยตรง ทำให้ระบบทั้งซับซ้อน ขยายระบบยาก และดูแลรักษาลำบาก

แนวทางของ A2A

A2A มีวิธีช่วยให้ AI ที่ทำงานแยกกัน เหมือนกล่องดำที่เราไม่เห็นข้างใน สามารถเชื่อมต่อและคุยกันได้ โดยกำหนดเรื่องสำคัญๆ ไว้ เช่น

- **วิธีส่งข้อมูล** ใช้มาตรฐาน **JSON-RPC 2.0 ผ่าน HTTP(S)** เป็นรูปแบบกลางในการส่งข้อความหากัน
- **วิธีหา AI ตัวอื่น (Agent Cards)** ให้ AI แต่ละตัวมี “นามบัตร” บอกว่าตัวเองทำอะไรได้บ้าง เพื่อให้ AI ตัวอื่นค้นหาและเรียกใช้ได้
- **ขั้นตอนการทำงาน** กำหนดวิธีเริ่มงาน ทำงาน และจบงานให้ชัดเจน แม้จะเป็นงานที่ใช้เวลานานหรือมีหลายขั้นตอน
- **รองรับข้อมูลได้หลายแบบ** ไม่ได้คุยกันได้แค่ตัวหนังสือ แต่ส่งไฟล์ ส่งข้อมูลที่เป็นตาราง เช่น ฟอร์ม หรือสื่ออื่นๆ ก็ได้
- **ความปลอดภัยและการทำงานแบบไม่รอได้ (Asynchronous)** ออกแบบมาให้ปลอดภัย และรองรับงานที่เวลาประมวลผลนาน หรือมีคนเข้ามาเกี่ยวข้องในบางขั้นตอน

หลักการออกแบบ A2A

A2A ออกแบบโดยเน้นหลักการเหล่านี้

- **เน้นเรียบง่าย (Simplicity)** ใช้เทคโนโลยีที่คนคุ้นเคยกันดีอยู่แล้ว เช่น HTTP, JSON-RPC และ Server-Sent Events (SSE) ไม่ต้องเรียนรู้ของใหม่ให้วุ่นวาย

- **พร้อมสำหรับองค์กรใหญ่ (Enterprise Readiness)** คิดเรื่องความปลอดภัย การยืนยันตัวตน การอนุญาตให้ใช้สิทธิ์ การติดตามงาน และการตรวจสอบระบบไว้ตั้งแต่ต้น

- **ทำงานแบบไม่รอได้ (Asynchronous First)** เหมาะกับงานที่ใช้เวลานาน หรือเวลาที่ AI หรือคนอาจจะไม่ได้ออนไลน์ตลอดเวลา โดยมีวิธีส่งข้อมูลแบบต่อเนื่อง (streaming) หรือแจ้งเตือนเมื่อพร้อม

- **ข้อมูลแบบไหนก็ได้ (Modality Agnostic)** AI คุยกันได้ด้วยข้อมูลหลายรูปแบบ ไม่ใช่แค่ข้อความอย่างเดียว

- **ไม่ต้องโชว์ไส้ใน (Opaque Execution)** AI ทำงานร่วมกันได้โดยไม่ต้องเปิดเผยว่าข้างในทำงานยังไง (เหมือนกล่องดำที่เราไม่เห็นข้างใน) หรือใช้เครื่องมืออะไรเป็นพิเศษ เพื่อช่วยรักษาความลับทางการค้าและเพิ่มความปลอดภัย

ประโยชน์ของการใช้ A2A

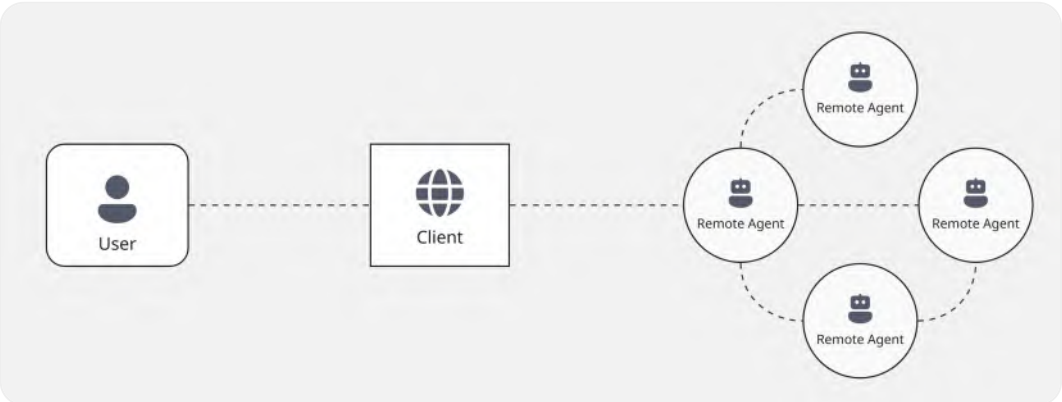
ถ้าใช้ A2A ในระบบ AI จะมีข้อดีหลายอย่าง เช่น

- **ทำงานร่วมกันได้ดีขึ้น (Interoperability)** AI จากคนละค่าย คนละคนสร้าง ก็ยังทำงานด้วยกันได้
- **AI เก่งขึ้น** นักพัฒนาเอาความสามารถของ AI หลายๆ ตัวมารวมกัน สร้างเป็นแอปพลิเคชันที่ทำอะไรซับซ้อนได้มากขึ้น
- **เชื่อมต่อง่าย ไม่ปวดหัว** มีมาตรฐานกลางแล้ว ทีมพัฒนาก็ไปเน้นสร้าง AI ให้เก่งๆ ดีกว่า ไม่ต้องเสียเวลากับการเชื่อมระบบ
- **เกิดนวัตกรรมใหม่ๆ** เปิดทางให้มี AI เฉพาะทางเยอะขึ้น แล้ว AI เหล่านี้ก็มารวมพลังกันเป็นระบบที่ใหญ่และเก่งขึ้นได้ง่ายๆ
- **พร้อมสำหรับอนาคต** โครงสร้างของ A2A ยืดหยุ่นพอที่จะปรับตัวตามเทคโนโลยี AI ที่เปลี่ยนไปได้เสมอ

การมีมาตรฐานกลางให้ AI คุยกันได้ ก็เหมือนการวางเสาหลักให้ระบบ AI ต่างๆ ทำงานร่วมกันได้จริงและมีประสิทธิภาพ A2A จึงเป็นเครื่องมือชิ้นสำคัญที่จะช่วยให้เรานำ AI ไปใช้ในงานต่างๆ ได้เร็วขึ้น และสร้างระบบอัจฉริยะที่ทำงานประสานกันได้อย่างลงตัวในอนาคต

แนวคิดหลักของโปรโตคอล A2A

โปรโตคอล Agent2Agent (A2A) ออกแบบมาเพื่อให้ AI Agent ต่างๆ ทำงานร่วมกันได้ โดยกำหนดวิธีที่ระบบ AI จะเข้าใจและสื่อสารกันอย่างมีประสิทธิภาพ การทำความเข้าใจแนวคิดหลักของ A2A จึงสำคัญมากสำหรับนักพัฒนาที่ต้องการสร้างระบบใหม่ หรือปรับปรุงระบบเดิมให้เป็นไปตามมาตรฐานนี้



บทบาทหลักในระบบ A2A

บทบาทหลักในระบบ A2A

ผู้ใช้ (User)

คือผู้ที่ต้องการความช่วยเหลือจาก Agent อาจเป็นคนจริงๆ หรือระบบอัตโนมัติก็ได้ ผู้ใช้จะเป็นคนเริ่มส่งคำขอหรือตั้งเป้าหมายให้ Agent

A2A Client (Client Agent)

A2A Client (Client Agent) คือแอปพลิเคชัน, บริการ หรือ AI Agent ที่ทำหน้าที่เป็นตัวแทนของผู้ใช้ Client Agent จะใช้โปรโตคอล A2A เพื่อส่งคำขอไปยัง Agent ที่อยู่ไกลออกไป (Remote Agent) เพื่อขอข้อมูลหรือสั่งให้ทำงานบางอย่าง

A2A Server (Remote Agent)

A2A Server (Remote Agent) คือ Agent หรือระบบ Agent ที่ทำหน้าที่เป็นผู้ให้บริการ โดยจะเปิด HTTP endpoint ให้ Client Agent อื่นๆ เข้ามาเชื่อมต่อตามมาตรฐาน A2A Server จะรับคำขอจาก Client, ประมวลผล แล้วส่งผลลัพธ์หรือสถานะกลับไป Client ไม่จำเป็นต้องรู้ว่า Server ทำงานอย่างไรเบื้องหลัง (เป็นเหมือนกล่องดำ หรือ opaque system)

องค์ประกอบพื้นฐานในการสื่อสาร

Agent Card

Agent Card เป็นไฟล์ JSON ที่ให้ข้อมูลสำคัญ (metadata) เกี่ยวกับ Agent หนึ่งๆ Client Agent สามารถค้นหา Agent Card นี้ได้จาก URL มาตรฐาน เช่น /.well-known/agent.json ภายใน Agent Card จะบอกรายละเอียด เช่น

- ชื่อและคำอธิบายของ Agent
- URL ของ endpoint ที่ใช้ติดต่อ
- เวอร์ชันของระบบ
- ความสามารถพิเศษที่รองรับ เช่น การส่งข้อมูลแบบสตรีมมิง, การแจ้งเตือนแบบ push
- ทักษะ (skills) ที่ Agent นี้ทำได้
- รูปแบบข้อมูลนำเข้า (input) และผลลัพธ์ (output) ที่รองรับ
- ข้อกำหนดในการยืนยันตัวตน (authentication) Agent Card ช่วยให้ Client Agent ค้นหาและเชื่อมต่อกับ Server Agent ได้อย่างถูกต้อง ปลอดภัย และมีประสิทธิภาพ

Task

เมื่อ Client ส่งคำขอไปยัง Server Agent, Server อาจสร้าง “งาน” (Task) ขึ้นมาเพื่อจัดการคำขอนั้นๆ แต่ละ Task จะมีสถานะที่ชัดเจน เช่น

- **submitted** ส่งงานแล้ว
- **working** กำลังทำงาน
- **input-required** ต้องการข้อมูลเพิ่มเติม
- **completed** ทำงานเสร็จแล้ว
- **failed** ทำงานไม่สำเร็จ

ทุกๆ Task จะมี task ID ที่ไม่ซ้ำกัน เพื่อใช้อ้างอิง การทำงานหนึ่งๆ อาจมีการส่งข้อความโต้ตอบกันหลายครั้งระหว่าง Client และ Server จนกว่า Task นั้นจะเสร็จสมบูรณ์

Message

Message คือข้อมูลที่ Client และ Agent ส่งหากันในแต่ละครั้งของการโต้ตอบ โครงสร้างของ Message ประกอบด้วย

- **role** ระบุว่าผู้ส่งเป็น user (ฝั่ง Client) หรือ agent (ฝั่ง Server)
- **Part** เนื้อหาของข้อความ ซึ่งอาจมีได้อย่างน้อยหนึ่งส่วน (ดูรายละเอียด Part ด้านล่าง)
- **messageId** ID ของข้อความที่ไม่ซ้ำกัน (ผู้ส่งเป็นคนกำหนด)

Message ใช้สำหรับส่งคำสั่ง, คำถาม, คำตอบ หรือการอัปเดตสถานะทั่วไปที่ไม่ใช่ผลลัพธ์หลักของงาน (Artifact)

Part

Part เป็นหน่วยย่อยที่สุดของเนื้อหาที่อยู่ใน Message หรือ Artifact มีหลายประเภท ได้แก่

- **TextPart** สำหรับข้อความธรรมดา

- **FilePart** สำหรับไฟล์แนบ อาจเป็นข้อมูลไฟล์ที่เข้ารหัสแบบ base64 หรือเป็น URI (ที่อยู่ของไฟล์) พร้อมระบุประเภทไฟล์ (MIME type) และชื่อไฟล์
- **DataPart** สำหรับข้อมูลที่มีโครงสร้างแบบ JSON เหมาะสำหรับข้อมูลแบบฟอร์ม หรือข้อมูลที่ให้โปรแกรมคอมพิวเตอร์อ่านและเข้าใจได้ง่าย

Artifact

Artifact คือผลลัพธ์ที่เป็นชิ้นเป็นอันที่ได้จากการทำงานของ Remote Agent เช่น ไฟล์เอกสาร, รูปภาพ, ไฟล์ Excel หรือข้อมูล JSON Artifact สามารถส่งกลับมาให้ Client แบบต่อเนื่อง (streaming) ได้

รูปแบบการโต้ตอบ

Request/Response (Polling)

Request/Response (Polling) เป็นรูปแบบพื้นฐานที่ Client ส่งคำขอ (request) ไปยัง Server และรอรับผลลัพธ์ (response)

1. Client ส่งคำขอผ่าน message/send
2. ถ้า Server ทำงานเสร็จทันที ก็จะส่งผลลัพธ์กลับมา

3. ถ้างงานนั้นต้องใช้เวลานาน Server อาจตอบกลับมาก่อนว่า working (กำลังทำงาน) พร้อมให้ task ID มา
4. จากนั้น Client จะต้องคอยสอบถามสถานะของ Task เป็นระยะๆ (polling) โดยใช้ tasks/get พร้อมกับ task ID จนกว่า Task นั้นจะเสร็จ

Streaming (Server-Sent Events - SSE)

Streaming เหมาะสำหรับงานที่ผลลัพธ์ทยอยออกมาเรื่อยๆ หรือต้องการการอัปเดตแบบ real-time

1. Client เรียกใช้ message/stream เพื่อเริ่มการเชื่อมต่อแบบสตรีม
2. Server จะตอบกลับด้วยการเชื่อมต่อ HTTP ที่เปิดค้างไว้ (keep-alive) และจะทยอยส่งข้อมูล (event) กลับมาให้ Client อย่างต่อเนื่อง
3. Event ที่ Server อาจส่งมา เช่น TaskStatusUpdateEvent (อัปเดตสถานะงาน) หรือ TaskArtifactUpdateEvent (อัปเดตผลลัพธ์งาน) Server Agent ต้องประกาศความสามารถในการทำ Streaming นี้ไว้ใน Agent Card ด้วย

Push Notifications

Push Notifications ใช้ในกรณีที่ Task ใช้เวลานานมาก หรือ Client ไม่สามารถเปิดการเชื่อมต่อค้างไว้เพื่อรอได้

1. ตอนที่ Client เริ่ม Task, Client จะระบุ URL ของตัวเอง (เรียกว่า webhook URL) ที่ต้องการให้ Server แจ้งเตือนกลับไป (หรือตั้งค่าผ่าน tasks/pushNotificationConfig/set ในภายหลัง)
2. เมื่อ Task มีการเปลี่ยนแปลงสถานะที่สำคัญ เช่น ทำงานเสร็จแล้ว หรือเกิดข้อผิดพลาด Server จะส่งข้อมูลแจ้งเตือน (POST request) ไปยัง webhook URL ที่ Client ได้ให้ไว้

Server Agent ต้องประกาศความสามารถในการทำ Push Notification นี้ไว้ใน Agent Card ด้วย

แนวคิดสำคัญ

- **Context (contextId)** เป็น ID ที่ใช้เชื่อมโยง Task หลายๆ Task ที่เกี่ยวข้องกันเข้าไว้ด้วยกัน ทำให้เห็นภาพรวมและความต่อเนื่องของงาน
- **Transport และ Format (ช่องทางและรูปแบบการส่งข้อมูล)** A2A ใช้โปรโตคอล HTTP หรือ HTTPS ในการรับส่งข้อมูล และใช้ JSON-RPC 2.0 เป็นรูปแบบของข้อความที่สื่อสารกัน