4. A2A

이기종 Agent 문제

- 최고의 전문가들로 팀을 구성했지만...
  - 리서치 팀: crewAI의 역할 기반 협업 모델을 사용해, 자료 조사와 초안 작성에 특화된 Agent 팀 구축
  - 품질 검수 전문가: Google ADK로 작성된 글의 사실 관계와 논리적 오류를 검증하는 단일 Agent 구축
- 문제 상황
  - 두 팀 다 퍼포먼스가 좋다 하더라도 이들이 다른 프레임워크로 구성되어
  - 리서치 팀이 검수 전문가에게 작업을 전달하고 피드백을 받을 방법이 없다면?

이기종 Agent 문제

- 프레임워크 종속성 (Framework Dependency)
  - 각 Agent 프레임워크는 자신만의 고유한 방식으로 Agent, Task, State를 정의하고 관리
  - crewAI
    - Agent는 role, goal, backstory로 정의되고, 작업은 Task 객체로 관리되며, process에 따라 협업
  - ADK
    - Agent는 Agent 클래스를 상속받아 구현되고, 모든 실행 과정은 Trajectory와 Event로 기록
  - 서로 다른 언어와 문법을 극복하는 방법은?

임시 해결책

- 표준화된 작업 스키마 (Standardized Task Schema)
  - 완전한 프로토콜을 도입하기 전 시도된 첫걸음
  - 시스템 내의 모든 Agent가 작업을 요청하고 결과를 반환할 때 준수해야 하는 공통 데이터 구조를 정의
  - Python의 Pydantic 라이브러리를 활용
    - 데이터 유효성 검사 기능을 제공하여, 이러한 표준 스키마를 정의하고 강제하는 데 매우 유용

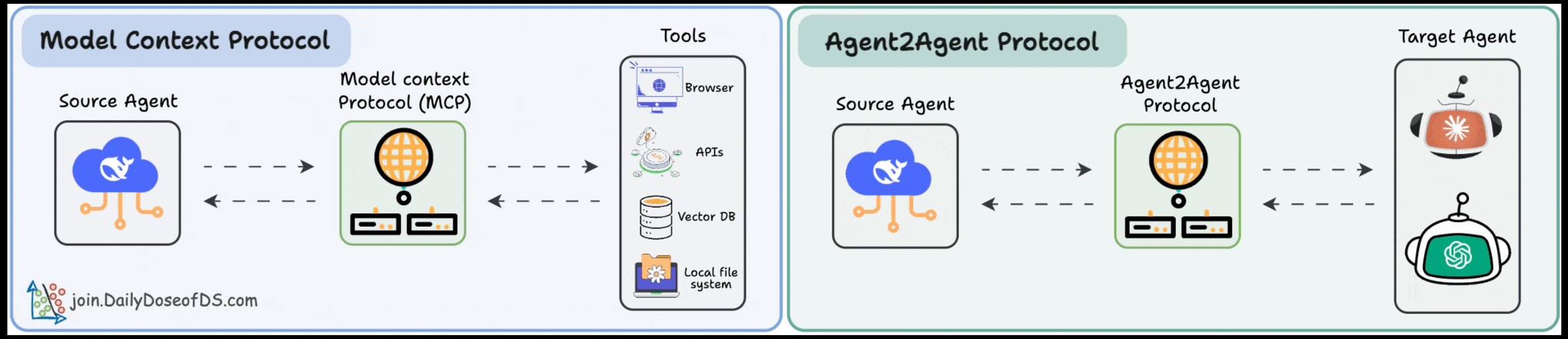
```python
from pydantic import BaseModel, Field
from typing import Dict, Any, Optional

class Task(BaseModel):
    task_id: str = Field(..., description="작업의 고유 식별자")
    source_agent: str = Field(..., description="작업을 요청한 에이전트의 ID")
    target_agent: str = Field(..., description="작업을 수행할 에이전트의 ID")
    task_name: str = Field(..., description="수행할 작업의 이름")
    payload: Dict[str, Any] = Field(..., description="작업 수행에 필요한 데이터")
    dependencies: Optional[list[str]] = Field(None, description="선행되어야 할 작업 ID 목
```

임시 해결책

- 작업 스키마의 한계
  - 표준 작업 스키마는 '무엇을' 보낼지에 대한 약속이지만 아래의 질문에는 답하지 못함
  - 내 시스템에 어떤 Agent들이 존재하는지 어떻게 찾지?(Discovery)
  - 시스템과 해당 Agent를 어떻게 안전하게 연결하지?(Connection)
  - 단순 작업 전달 외에, 복잡한 협상이나 다단계 대화는 어떻게 진행하지?(Interaction)

A2A

- Google A2A (Agent-to-Agent)
  - AI 생태계를 위한 개방형 프로토콜 제안
  - Agent들이 마치 인터넷의 웹사이트처럼 서로를 동적으로 발견하고, 연결하며, 자율적으로 상호작용하는 세상을 구축하는 것이 목표



https://blog.dailydoseofds.com/p/a-visual-guide-to-agent2agent-a2a

A2A

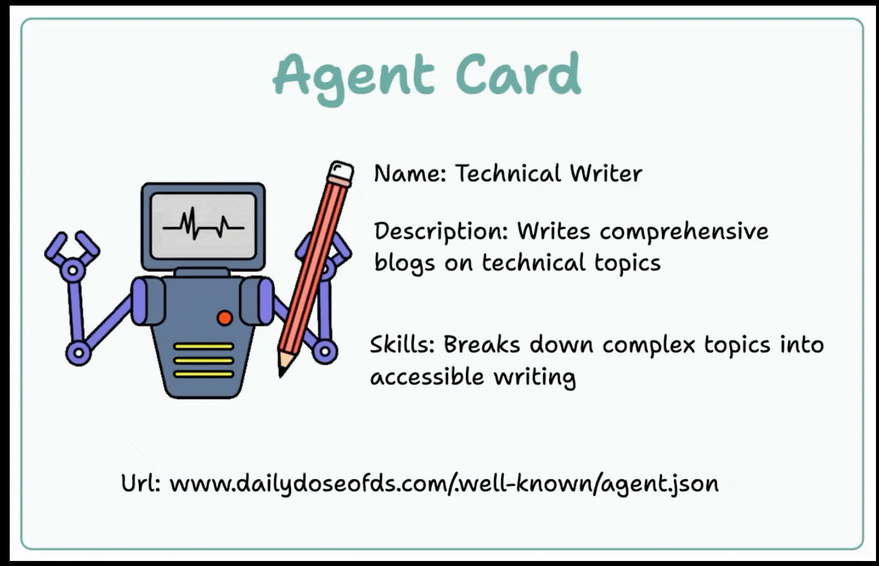
- Discovery
  - DNS가 google.com이라는 도메인 이름을 IP 주소로 찾아주듯
  - A2A Discovery는 '항공권 예약 Agent'의 주소를 찾아줌
- Connection
  - TCP/IP가 안정적인 데이터 전송 채널을 만들듯
  - A2A Connection은 Agent 간의 신뢰할 수 있는 통로를 구축
- Interaction
  - HTTP가 웹 콘텐츠 요청/응답 방식을 정의하듯
  - A2A Interaction은 Agent 간의 서비스 요청/응답 방식을 정의

A2A의 핵심 구성 요소

- Agent ID (DID)
  - 각 Agent는 did:key: 와 같은 탈중앙화 식별자(Decentralized Identifier)를 통해 고유한 신원을 가짐
  - 이는 중앙 기관 없이도 Agent의 신원을 암호학적으로 검증 가능하게 함
- Endpoint
  - Agent가 메시지를 수신할 수 있는 네트워크 주소(URL)
  - Agent는 하나 이상의 엔드포인트를 가질 수 있음

A2A의 핵심 구성 요소

- Capabilities Document
  - Agent가 무엇을 할 수 있는지(능력)를 기술한 기계가 읽을 수 있는 문서
  - 이를 통해 다른 Agent가 해당 Agent의 기능을 파악
- Registry
  - Agent들이 자신의 ID, Endpoint, Capabilities를 등록하여 다른 Agent가 자신을 찾을 수 있도록 하는 서비스
  - 중앙화 또는 탈중앙화 방식으로 구현 가능



https://blog.dailydoseofds.com/p/a-visual-guide-to-agent2agent-a2a

A2A 통신

- A2A 통신의 3단계
  - Discovery – Connection – Interaction 3단계
  - Discovery (발견)
    - Agent가 특정 능력을 가진 다른 Agent를 Registry에 질의(Query)하여 찾음
    - Registry는 질의 조건에 맞는 Agent의 ID와 Endpoint, Capabilities 정보를 반환

A2A 통신

- A2A 통신의 3단계
  - Connection (연결)
    - 두 Agent가 처음 통신할 때, 신뢰 관계를 형성하기 위한 프로토콜 (connection/1.0)
    - request와 response 메시지를 교환하여 서로의 신원을 확인하고 통신 채널을 수립
    - 이 과정에서 DID를 활용한 암호학적 검증이 이루어짐
  - Interaction (상호작용)
    - 연결이 수립된 후, 특정 목적을 가진 프로토콜을 사용하여 메시지를 교환
    - 예: 단순 메시지 교환을 위한 basic-message/1.0, 작업 위임을 위한 delegated-task/1.0 등

A2A의 메시지 구조

- 표준화된 JSON 메시지 객체를 통해 전달
- 주요 필드
  - @type: 이 메시지가 어떤 프로토콜에 속하는지를 나타내는 식별자
    - 예: https://a2a-protocol.org/basic-message/1.0/message
  - @id: 메시지 자체의 고유한 ID
  - from: 메시지를 보낸 Agent의 DID
  - to: 메시지를 받는 Agent의 DID
  - body: 프로토콜별로 정의된 실제 내용이 담기는 객체

a2a-py 라이브러리

- A2A GitHub 프로젝트는 Python 참조 구현 라이브러리인 a2a-py를 제공
  - A2A 프로토콜의 복잡한 세부 사항을 몰라도 Agent를 쉽게 구축 가능

```python
from a2a.agent import Agent
from a2a.protocols.basic_message import BasicMessage, BasicMessageHandler

# 1. Agent 인스턴스 생성 (고유 DID 및 Endpoint 설정)
my_agent = Agent(did="did:key:z6M...", endpoint="http://localhost:8001/a2a")

# 2. 특정 프로토콜 메시지를 처리할 핸들러 등록
class MyMessageHandler(BasicMessageHandler):
    async def handle(self, message: BasicMessage, sender: Agent):
        print(f"메시지 수신: '{message.content}' from {sender.did}")
        # 여기에 메시지 처리 로직 구현

my_agent.add_protocol(MyMessageHandler())
```

a2a-py 라이브러리

```python
import asyncio
from a2a.agent import Agent
from a2a.protocols.basic_message import BasicMessage

async def main():
    # 보내는 Agent 생성
    sending_agent = Agent()

    # 받는 Agent의 정보 (DID, Endpoint) - Discovery를 통해 획득
    recipient_did = "did:key:z6M..."
    recipient_endpoint = "http://localhost:8001/a2a"

    # 1. 메시지 객체 생성
    message = BasicMessage(content="안녕하세요, A2A 테스트 메시지입니다.")
```

- a2a-py를 이용한 Agent 간 통신 예시
  - 실제 운영 환경에서는 recipient_did와 recipient_endpoint를 하드코딩하는 대신 Registry를 통한 Discovery 과정이 선행