DIDComm Tunnels

HTTP over DIDComm

Or

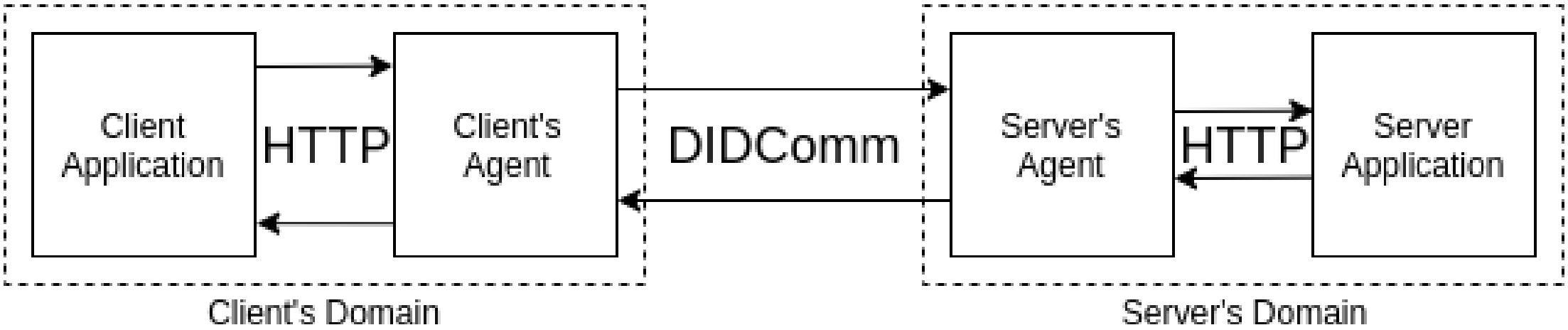
DIDComm as HTTP Transport Layer

Filip Burlacu

Outline

- HTTP over DIDComm (RFC 0335)
- Purpose Decorator (RFC 0351)

RFC 0335: HTTP over DIDComm

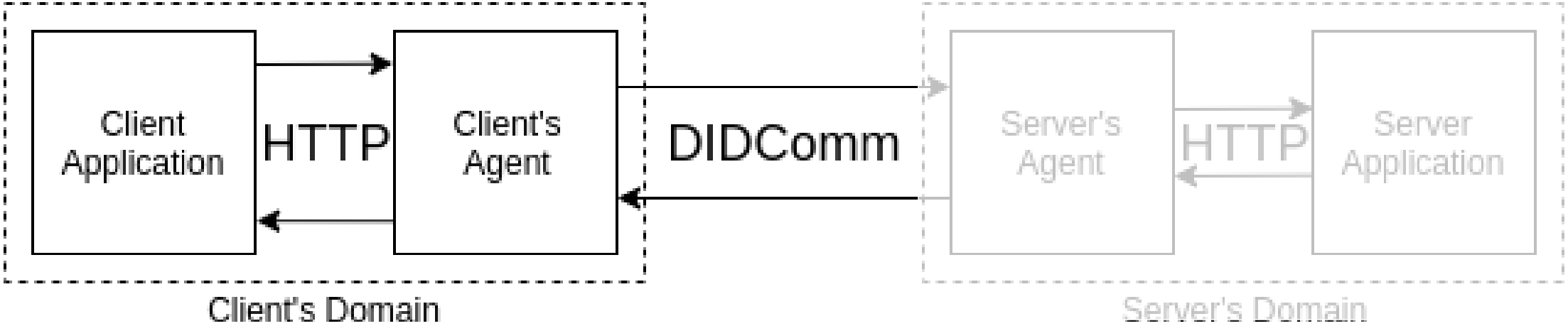


RFC 0335 Use Cases

- Porting HTTP(S) client-server systems to a DIDComm architecture
- Providing a DIDComm external API for a cloud service
- Making use of the wide universe of HTTP server-side frameworks and infrastructure (any REST server, etc) to provide services, while using Aries agents for identity management and secure communication

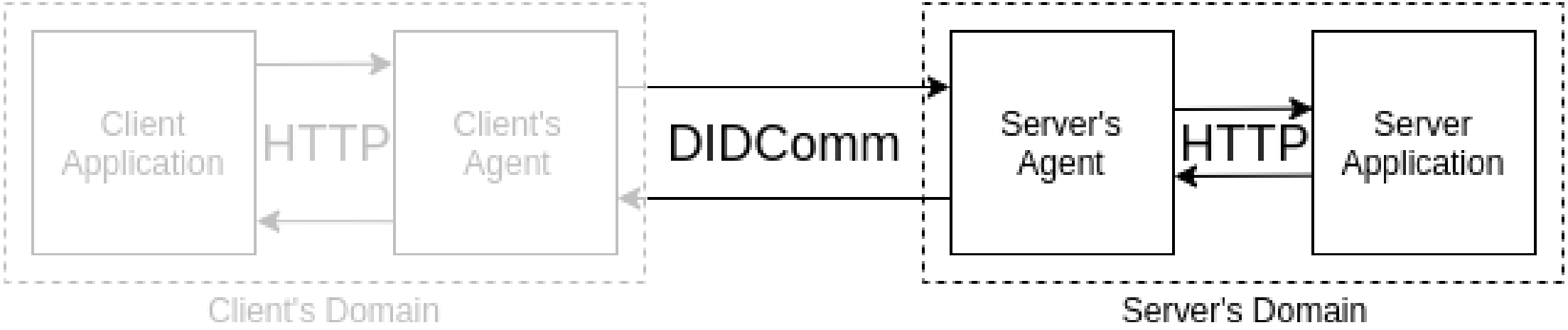
Client-Side

- Client sets up their Agent as a proxy for their application
- Application sends HTTP request to Server, through the Client agent as proxy
- Client agent creates DIDComm message from the request, sending this to the server.



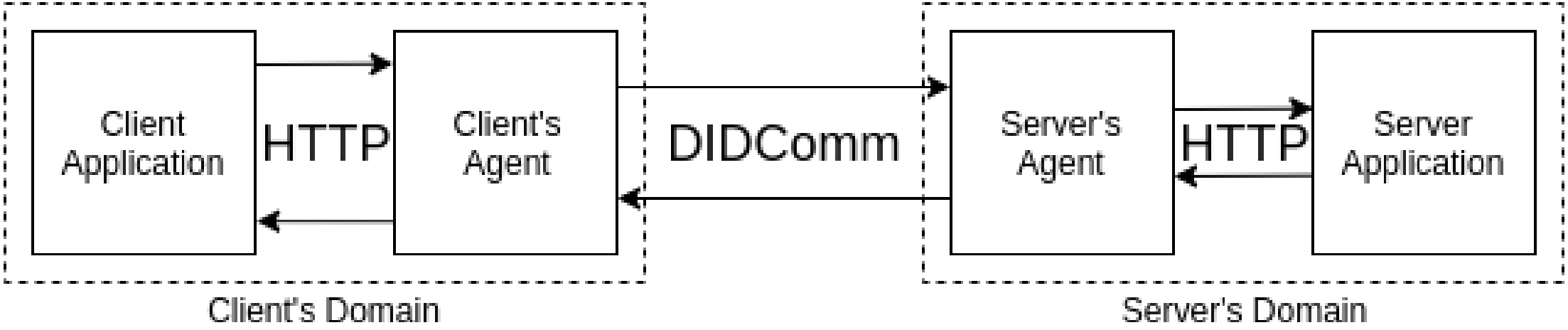
Server-Side

- Server *registers* with its agent to receive a certain class of messages (more on that later)
- Server agent receives a message
- If this message matches the server's registration, the message is translated to HTTP and sent to the server



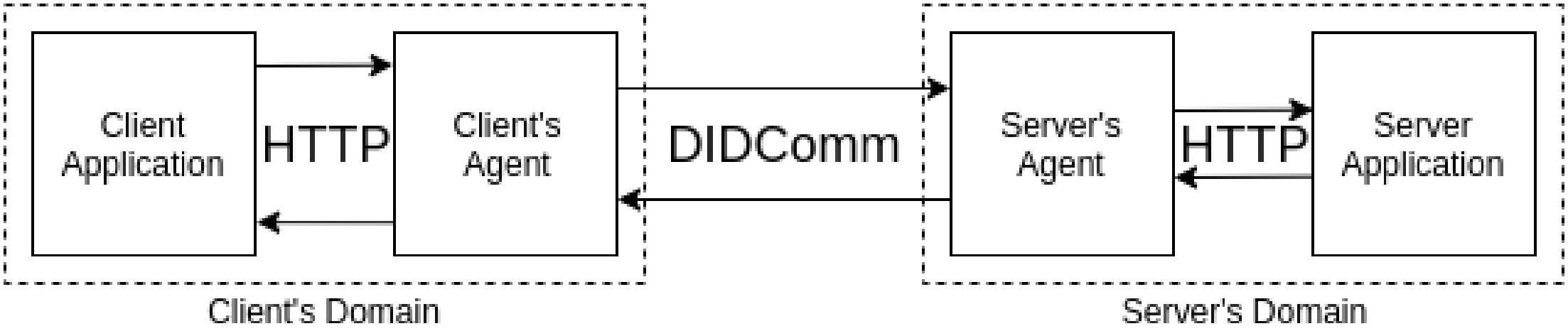
Responses

- Using Transport Return Route, Client agent can specify how the response should reach it
- Using Threading, Client agent can determine which client connection a response is meant for
- Using Timing, Client agent can specify when it will timeout its client app connection

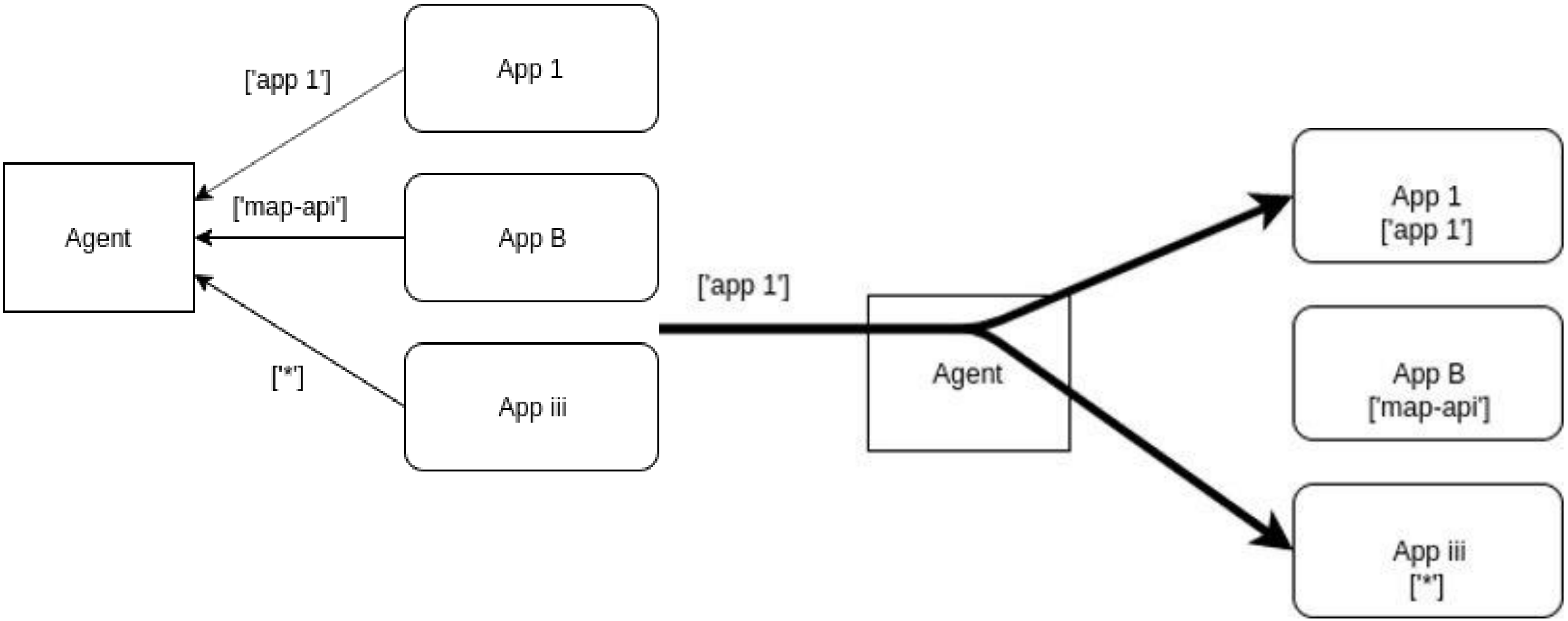


Security

- Agents see the contents of the messages
- Each end of the message flow owns their agent
- Agents and their hosts can be configured to use HTTPS



RFC 0351: Purpose Decorator



RFC 0351 Use Cases

- Using an Aries framework in a client application, the purpose decorator can filter messages for client-level protocols
- Using an Aries agent in a server, the agent can serve as the DIDComm endpoint for the server API
- Multiple client applications can use a single agent to perform transactions on the owner's identity
- The purpose decorator can be used to implement an agent-controller interface

RFC 0351: Purpose Filtering

- Non-DIDComm applications register with an agent on a *purpose* value
- When the agent receives a message which matches the purpose value, it sends the message to the application
- The purpose is a JSON array of strings, and there will be a standard matching algorithm (exact algorithm tbd) to filter different messages to different targets

Open Questions

- How should we handle cases where a message matches the registration of multiple applications?
  - An auditing service might need to receive all messages, but should not prevent other services from handling messages
  - If multiple services handle a message, and several of them send responses, how does the agent reconcile these?
- How, specifically, should purpose decorator values match to registrations?