Composite Service Recommendation Using Contract-Net-Based LLM Multi-Agent Systems

Yuki Tobisawa
College of Information Science and Engineering
Ritsumeikan University
Kusatsu, Japan

Yohei Murakami
College of Information Science and Engineering
Ritsumeikan University
Kusatsu, Japan

Abstract—**Selecting appropriate Web service APIs from user requirements described in natural language to compose composite services is a critical challenge in automatic service composition. Large Language Model (LLM)-based agents that infer Web services from natural language requirements have been proposed; however, single-agent approaches must consolidate numerous API specifications into a single prompt, which tends to degrade inference accuracy due to prompt bloating and context-length limitations. This paper proposes a service composition method based on a multi-agent system in which multiple LLM agents coordinate through the Contract Net Protocol (CNP). In the proposed method, contractor agents are specialized to individual APIs and distributedly maintain API specifications to alleviate prompt constraints. The reasoning process for service composition is decomposed into four steps—core function decomposition, category mapping, API matching, and API selection—and distributed between a manager agent and contractor agents through the CNP announcement-bidding-award framework. Furthermore, treating the responsibility assignment of reasoning tasks as a design variable, we design and compare three protocols—manager-led, contractor-led, and collaborative—to clarify how the granularity of task decomposition and the degree of specialization affect recommendation accuracy and inference cost. Evaluation experiments using the ProgrammableWeb dataset confirm that the manager-led protocol, which narrows candidate categories top-down, achieves the highest recommendation accuracy compared with a single-agent baseline and the other two protocols. Meanwhile, we reveal a trade-off: while the fully autonomous contractor-led protocol struggles to maintain consistency at the integration stage, hierarchical decision-making that retains a degree of centralized control is more effective for service composition.**

Index Terms—**service composition, multi-agent system, large language model, contract net protocol, API recommendation**

I. Introduction

With the development of service computing, composite services that combine diverse Web services to provide new added value have been actively constructed. Real-time speech translation linking translation and speech recognition services, and map display services integrating geographic and weather information services, are examples of flexible functionality that individual Web services alone cannot realize. However, only a fraction of publicly available Web services are actually used in composite services, and selecting services suited to a user's purpose from a vast pool of candidates remains a significant burden.

Conventional composite service composition is broadly divided into two approaches: horizontal and vertical service composition. The former selects combinations of Web services suitable for executing each task in a given workflow, but incurs high workflow-design costs. The latter uses AI planning techniques to generate service execution sequences that reach a goal state expressed in logical formulas; however, it requires users to describe requirements using formal expressions such as temporal logic or predicate logic, which is not user-friendly for end users. To mitigate this, methods that estimate service combinations from requirements described in natural language, and methods that vectorize WSDL documents, service descriptions, and service networks through text mining or graph embedding and cluster them by function, have been proposed.

Meanwhile, Large Language Models (LLMs), which have advanced rapidly in recent years, possess the ability to generate plans and invoke external tools or APIs from requirements described in natural language, and are expected to serve as a framework for service composition without requiring formal descriptions from users. However, when a single general-purpose LLM agent handles a broad service domain, its understanding of individual service specifications and execution environments tends to become shallow. Moreover, packing numerous API specifications and usage examples into a single prompt causes performance degradation due to prompt bloating and context-length limitations.

In this paper, we propose a service composition method based on a multi-agent system composed of multiple LLM agents. By specializing each agent to a specific API category or functional domain, we aim to localize the knowledge handled within prompts while achieving highly specialized service recommendations. Furthermore, we introduce the Contract Net Protocol (CNP) [1] as a task allocation mechanism, designing a framework in which a manager agent and contractor agents cooperatively perform service composition through task decomposition and bidding. We compare three types of task decomposition protocols—manager-led, contractor-led, and collaborative—and evaluate the differences in service composition accuracy and performance compared with a single-agent approach, thereby clarifying the effectiveness and challenges of LLM-based multi-agent service composition.

The two challenges addressed in this study are as follows:

- **Task Decomposition:** Performing appropriate task

decomposition from user requirements described in natural language based on CNP. A mechanism is needed that can switch among different task decomposition strategies—manager-led, contractor-led, and collaborative—according to the granularity and ambiguity of requirements, while still obtaining consistent service composition results.
- **Organization of Contractor Agents:** Specializing agents to specific API categories or functional domains to suppress prompt bloating while sufficiently reflecting specialized knowledge of service specifications and execution environments. It is also necessary to clarify how differences in task decomposition protocols affect information sharing and reasoning processes among agents.

The remainder of this paper is organized as follows. Section II reviews horizontal, vertical, and LLM-based service composition approaches. Section III presents the multi-agent method based on the Contract Net Protocol and the design of task decomposition protocols. Section IV describes the evaluation metrics, experimental settings, and results. Section V provides discussion. Section VI concludes the paper and outlines future work.

II. Related Work

A. Horizontal Service Composition

Horizontal service composition selects, from candidates capable of executing each task in a pre-defined workflow, the combination that is most desirable for the overall composition. Zeng et al. proposed a composition method that selects services based on QoS (non-functional properties) rather than functional differences, assuming situations where many services provide the same functionality [2]. Specifically, they define a QoS vector for each service from non-functional information such as price, response time, and availability, and evaluate candidates using constraints and preferences (weights) specified by the user.

B. Vertical Service Composition

Vertical service composition uses AI planning techniques to generate a sequence of Web service executions (a composition procedure) that reaches a goal state specified by the user. Carman et al. treated service composition as a planning problem and proposed a composition algorithm suited to the open and uncertain Web environment [3]. Their approach combines semantic type matching—which judges whether an output type is semantically related to an input type of another service or a goal type—with an iterative composition that interleaves search and execution, thereby maintaining flexibility under incomplete information.

C. LLM-Based Service Composition

In LLM-based service composition, LLMs' natural language understanding is exploited to estimate the intent and constraints behind user requirements and to perform task splitting and API selection accordingly. However, it has been noted that simply applying LLMs via prompts degrades recommendation accuracy due to insufficient up-to-date API knowledge and hallucination. Furthermore, conventional models such as BERT cannot fully capture API domain-specific knowledge or complex mashup usage patterns.

Qin et al. proposed LLMAR based on Llama-3.2-3B [4], showing a framework that injects API domain knowledge into the model through instruction tuning and multi-stage fine-tuning. Matsumoto et al. proposed Guided Reasoning Chains (GRC) [5], which hierarchically decompose the reasoning process and guide the LLM through core function extraction, category selection, candidate enumeration, and final recommendation according to the abstraction level of the requirement.

III. Proposed Method

A. Contract Net Protocol

The Contract Net Protocol (CNP), proposed by Reid G. Smith [1], is a high-level communication protocol for distributed problem solving. In CNP, the agent that wishes to delegate a task is called the *manager*, and the agent that undertakes task execution is called the *contractor*. The protocol specifies a series of interactions between them: task announcement (call for proposals), bidding (proposal submission), award (contract), and task execution with status reporting.

B. System Overview

The proposed service composition system is based on an LLM multi-agent system and aims to recommend combinations of Web service APIs from composite service requirements given in natural language.

The system comprises a manager agent that accepts user requirements and orchestrates the overall composition, multiple contractor agents specialized to individual APIs, and API description texts referenced by each agent. The CNP is adopted as the inter-agent coordination framework, distributing reasoning tasks through the Call-for-Proposals, Proposals, and Award flow.

The reasoning process for service composition is defined as the following four-step common flow:

1) **Core Function Decomposition:** Estimate the user's ultimate objective from the requirement text and extract the required functional units.
2) **Category Mapping:** Estimate API categories (e.g., payment, maps, booking management) that can realize the extracted functions.
3) **API Matching:** Extract candidate APIs based on the estimated categories and API description texts relevant to the requirement.
4) **API Selection:** Compare the extracted candidate APIs and finalize the adopted APIs from the perspective of requirement fitness.

These four steps are not executed monolithically by a single agent but are treated as reasoning tasks distributed between the manager and contractors within the CNP announcement-bidding-award framework. The manager handles acceptance of the requirement text, definition and announcement of reasoning tasks, aggregation and consistency checking of bids, and finalization of the API set. Each contractor uses its assigned API knowledge to present judgments, candidate APIs, and rationale as bids in response to announced tasks.

C. LLM-Based Agents

1) Contractor Agent: A contractor agent is an LLM agent specialized to an individual API. It receives a task specification announced by the manager as input and performs reasoning using only the API description text it can reference, judging whether it can handle the task. If it judges it can, it returns an API candidate it considers suitable and its rationale as a bid. If it judges it cannot, it returns a decline message.

2) Manager Agent: The manager agent is an LLM agent that accepts user requirements and oversees the entire service composition. Based on the defined reasoning process, the manager constructs reasoning tasks and announces them to contractor agents via CNP. After receiving bids, the manager compares and integrates the API candidates and rationale presented by multiple contractors, verifies consistency with the overall requirement, and finalizes the adopted API set.

D. Task Decomposition Protocols

In this study, the responsibility assignment of reasoning tasks—which reasoning step is handled by which agent—is treated as a design variable within CNP. We define three protocols with different task distribution policies: manager-led, contractor-led, and collaborative.

1) Manager-Led Protocol: In the manager-led protocol, the manager handles the majority of the reasoning required for service composition. The role assignment across the four-step reasoning process is as follows:

- **Manager:** (1) Core Function Decomposition, (2) Category Mapping, (4) API Selection
- **Contractor:** (3) API Matching

The manager agent analyzes the user requirement, determines the necessary core functions and corresponding API categories in advance, and announces them to contractors as task specifications. Each contractor performs API matching based on the announced functions and categories using its own API description text, and returns suitable API candidates and rationale as bids. The manager compares and integrates the collected bids and finalizes the API set.

2) Contractor-Led Protocol: In the contractor-led protocol, the majority of reasoning is delegated to contractor agents.

- **Manager:** (4) API Selection
- **Contractor:** (1) Core Function Decomposition, (2) Category Mapping, (3) API Matching

The manager announces the user requirement text itself as a task without performing detailed decomposition or category estimation. Each contractor interprets the requirement text based on its own API knowledge, performs core function decomposition and category mapping, and presents suitable API candidates and rationale as bids. The manager aggregates the bids and finalizes the adopted API set.

3) Collaborative Protocol: The collaborative protocol combines the roles of the manager and contractors.

- **Manager:** (1) Core Function Decomposition
- **Contractor:** (2) Category Mapping, (3) API Matching, (4) API Selection

The manager performs core function decomposition from the user requirement and announces the extracted functional units as tasks. Each contractor estimates category candidates for the announced functions, performs API matching and selection based on its API knowledge, and returns API candidates and rationale as bids. The manager integrates the bids and finalizes the API set.

IV. Evaluation

A. Experimental Settings

We compare the three CNP-based multi-agent protocols (manager-led, contractor-led, and collaborative) together with a single-agent baseline. For contractor agent organization, we adopt a single-API-per-agent design; assigning multiple APIs per agent by category is left as future work. In all settings, given a user requirement (mashup specification) as input, an API set is generated and evaluated against the ground truth. The LLM used is gpt-oss20B.

1) Dataset: We use composite service (mashup) data and atomic service (Web API) data published on ProgrammableWeb. Mashup data contains the mashup name, categories, description, constituent APIs, and ID. Atomic service data contains the API name, categories, description, provider, consumers, and ID. After filtering records whose constituent APIs do not exist in the atomic service data, we obtained 4,284 mashup records. From these, 100 mashup records were sampled as the evaluation dataset targeting 909 corresponding Web APIs.

2) Evaluation Metrics: We use Precision, Recall, and F1 score based on API-name set matching to evaluate recommendation quality. In addition to the final recommendation, we compute these metrics at intermediate stages of each protocol to analyze at which point accuracy changes during the reasoning process:

- **Category Assignment:** All APIs belonging to the estimated categories form the candidate set P, evaluated against the ground truth G.
- **Bidding:** The API candidate set presented by contractor bids is evaluated.
- **Final Recommendation:** The API set finalized by the manager after aggregating all bids is evaluated.

We also measure input/output token counts for each protocol and compare average values.

B. Results

Table I summarizes the recommendation results at each stage. Table II reports the average token counts for each

TABLE I
RECOMMENDATION RESULTS AT EACH STAGE

Stage	Protocol	Prec.	Rec.	F1
Category Assignment	Manager-led	0.111	0.607	0.177
	Contractor-led	0.145	0.357	0.184
	Collaborative	0.047	0.347	0.078
	Single-agent	0.174	0.310	0.208
API Matching	Manager-led	0.047	0.485	0.079
	Contractor-led	0.020	0.570	0.031
	Collaborative	0.034	0.260	0.058
	Single-agent	0.106	0.121	0.105
API Selection	Manager-led	0.159	0.298	0.193
	Contractor-led	0.034	0.116	0.049
	Collaborative	0.083	0.223	0.114
	Single-agent	0.137	0.141	0.130

TABLE II
AVERAGE TOKEN COUNTS PER PROTOCOL

Protocol	Input	Output	Total
Manager-led	9,843	2,717	12,560
Contractor-led	1,952,788	999,576	2,952,364
Collaborative	522,411	71,828	594,240
Single-agent	14,324	28,580	42,904

protocol. Table III shows the average number of APIs in the final recommendation.

V. DISCUSSION

A. Comparison with Single-Agent Baseline

A single general-purpose LLM agent that consolidates numerous API specifications is prone to degraded inference accuracy due to prompt bloating and context-length constraints. In contrast, the manager-led protocol progressively refines candidates through task decomposition, category assignment, bidding, and integration, localizing domain context in each contractor's proposals. As shown in Table I, the single-agent baseline achieves Precision 0.137 and Recall 0.141 at the final selection stage, whereas the manager-led protocol achieves Precision 0.159 and Recall 0.298. Although the single-agent achieves a higher F1 at the category assignment stage (0.208 vs. 0.177), it falls behind at the final selection (0.130 vs. 0.193), suggesting that processing a large number of APIs monolithically introduces spurious candidates and lowers precision. The multi-agent protocols constrain the candidate space through task decomposition and category constraints, suppressing irrelevant APIs in the final recommendation, as also reflected in the lower average API counts (Table III).

B. Comparison Among Protocols

1) Recommendation Accuracy: The manager-led protocol demonstrated the most balanced performance (Precision 0.159, Recall 0.298). Its top-down category narrowing effectively limited the search space and excluded noise from irrelevant bids. The collaborative protocol (Precision 0.083, Recall 0.223) offers flexibility by having contractors estimate categories, but lacked system-wide consistency, leading to information loss at the integration stage. The contractor-led protocol (Precision 0.034, Recall 0.116) yielded the lowest accuracy. Although its bottom-up approach allows contractors to leverage

TABLE III
AVERAGE NUMBER OF APIS IN FINAL RECOMMENDATION

Protocol	Avg. APIs
Manager-led	3.19
Collaborative	3.57
Contractor-led	4.61
Single-agent	5.94

local API knowledge—as evidenced by a high recall of 0.570 at the API matching stage—the lack of unified criteria across agents made integration extremely difficult, resulting in many irrelevant candidates in the final recommendation.

2) Inference Cost and Efficiency: The advantage of the manager-led protocol in inference cost is pronounced. Its total token count (approx. 12,560) is roughly 1/50 of the collaborative (approx. 594,240) and 1/4 of the single-agent (approx. 42,904). This is because the manager-led protocol defines tasks and target categories upfront and solicits bids only from relevant contractors, achieving selective communication. The contractor-led protocol reached approx. 2,952,364 total tokens—approximately 236 times that of the manager-led—because each contractor independently interprets the requirement, generating massive traffic and filtering overhead.

3) Trade-off Between Responsibility Assignment and Flexibility: From the perspective of responsibility assignment, the following trade-off is observed. The manager-led protocol concentrates responsibility on the manager, posing a risk similar to a single point of failure if the initial category judgment is incorrect. However, the experimental results show that the benefit of improved search efficiency outweighs this risk. The collaborative and contractor-led protocols prioritize contractor autonomy and have the potential to compensate for the manager's judgment errors with local knowledge, but at the cost of increased integration complexity and communication overhead. In the current experimental setting, the efficiency-oriented manager-led protocol proved advantageous; however, for cases where requirements are highly ambiguous and category definition itself is difficult, the collaborative approach may be reevaluated.

VI. CONCLUSION

This paper proposed a Web service composition method based on an LLM multi-agent system. We introduced a task decomposition mechanism based on the Contract Net Protocol and designed and compared three task decomposition protocols: manager-led, contractor-led, and collaborative.

The contributions of this study are as follows:

- **Reasoning Task Allocation:** We formalized the reasoning process as a responsibility assignment model based on CNP and compared three protocols. The manager-led protocol narrows categories top-down, suppressing search space and noise while achieving the highest precision and recall. The collaborative protocol offers flexibility but suffers from communication cost and information loss during integration. We showed the necessity of adaptive protocol design according to the granularity and ambiguity of requirements.

- **Organization of Contractor Agents:** We demonstrated that distributing API knowledge to contractor agents rather than consolidating it in a single LLM alleviates context-length constraints and hallucination. Even with a single-API-per-agent configuration, scalability and update resilience for large API collections were confirmed. Optimizing the granularity by assigning multiple APIs per agent by category remains as future work.

As future work, we plan to evaluate multi-API-per-agent configurations to find the optimal assignment granularity, and to verify model dependency and robustness by conducting the same experiments with models other than gpt-oss20B.

ACKNOWLEDGMENT

The authors would like to thank Prof. Yohei Murakami and Kenji Matsumoto for their guidance and valuable advice throughout this research.

REFERENCES

[1] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. C-29, no. 12, Dec. 1980.

[2] L. Zeng, B. Benatallah, A. H. H. Ngu, M. Dumas, J. Kalagnanam, and H. Chang, "QoS-aware middleware for Web services composition," *IEEE Trans. Softw. Eng.*, vol. 30, no. 5, pp. 311–327, 2004.

[3] M. A. Carman, L. Serafini, and P. Traverso, "Web service composition as planning," in *Proc. ICAPS 2003 Workshop on Planning for Web Services*, 2003, pp. 1636–1642.

[4] S. Qin, Y. Zhao, H. Wu, L. Zhang, and Q. He, "Harnessing the power of large language model for effective Web API recommendation," *IEEE Trans. Ind. Inform.*, vol. 21, no. 7, Jul. 2025.

[5] K. Matsumoto and Y. Murakami, "Guided reasoning chains for API recommendation," in *Proc. ICSOC Workshops*, 2025.