

# Task Scheduling with Duplication Avoidance for ICN-based Autonomous In-Network Computing

Yuki Niibe

Graduate School of Information Science and Engineering  
Ritsumeikan University  
2-150 Iwakura-cho, Ibaraki, Osaka 567-8570, JAPAN  
gr0693pv@ed.ritsumei.ac.jp

Noriaki Kamiyama

College of Information Science and Engineering  
Ritsumeikan University  
2-150 Iwakura-cho, Ibaraki, Osaka 567-8570, JAPAN  
kamiaki@fc.ritsumei.ac.jp

**Abstract**—With the rise of IoT and 5G, there is increasing demand for linked processing of various tasks, such as IoT data processing. In-network computing combined with information-centric networking (ICN) offers a generic solution, but autonomous distributed computing face challenges like task duplication and inefficient resource use in DAG. This paper proposes a scheduling algorithm using step-by-step task ordering and one-stroke requests to prevent duplication.

**Index Terms**—In-network computing, ICN, SFC, Workflow, Task scheduling

## I. INTRODUCTION

In the 5G/Beyond 5G era, the digitalization of various things and the rise of data-intensive applications—such as AI-powered systems aggregating IoT data for value creation—are expected. However, edge-only processing is limited by computational capacity, while cloud reliance introduces inefficiencies and load concentration. To address this, in-network computing, which processes data en route from the edge to the core, has considered [1].

Conventional in-network computing focuses on limited area, but with the projected tenfold increase in IoT devices by 2030 [2], efficient large-scale distributed processing becomes essential. Centralized control is infeasible in such situation, prompting the need for autonomous in-network computing, where nodes independently manage task assignment and execution.

Information-Centric Networking-based Service Function Chaining (ICN-SFC) [3], [4] is a promising approach, but existing methods handle only simple task flows and lack support for DAG-based workflows essential for general-purpose processing. Extending ICN-SFC to support DAGs enables broader applications but raises the issue of redundant task assignment. In ICN, task placement is typically determined by recursively requesting results from exit to start tasks. However, in DAGs, multiple tasks can share a predecessor, and due to node-specific FIBs, the same task may be assigned to different nodes, wasting resources.

To solve this, we propose a scheduling method for workflows that allocates tasks in a one-stroke request sequence to prevent duplication. Our approach includes step-by-step task ordering using a ReadyList and request forwarding with

bundled unresolved requests. We validate the effectiveness of this method through simulation.

## II. PROPOSED METHOD OF TASK SCHEDULING

The method of extending the existing ICN-SFC to DAG applications (referred to as *multicast-like method*) is resulted in duplicate task assignments. This is because each task sends requests to multiple predecessors simultaneously, causing inconsistencies in the request destinations. Therefore, we propose a method in which tasks in a DAG are sequenced in one-stroke, requests are sent one by one according to the sequential order, and then tasks are allocated.

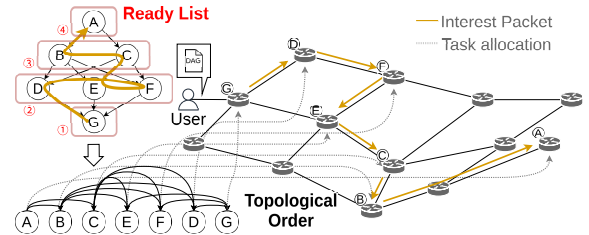


Fig. 1. Example of Step-by-step task ordering and one-stroke formation

### A. Step-By-Step Task Ordering and One-Stroke Formation Using ReadyList

Figure 1 shows the step-by-step task ordering and one-stroke formation using ReadyList, along with the request and allocation flow. First, to autonomously select the task assignment destination, we calculate the *blevel* upon Interest arrival at a node. This is performed for the current node and its neighboring nodes, and the node with the shortest Blevel is selected. This effectively determines whether to continue the request each time, allowing the system to naturally and autonomously find the task assignment destination. Next, we discuss the request method after a task is assigned. The multicast-like method sends the request to all predecessor tasks at once, whereas our method sends the request to a single task selected from the ReadyList. The ReadyList is a set of unallocated tasks to which all subsequent tasks have been allocated and is forwarded together with Interest packets. The ReadyList forwarded with the Interest packet is checked, and if it is not empty, a task is selected from it. If it is empty, it checks the DAG structure of the application and selects all

unallocated tasks for which all subsequent tasks have already been allocated at that stage. The selected tasks are inserted into the ReadyList. After the ReadyList is successfully created, a task is extracted from it. The retrieved task becomes the next request destination.

This process is repeated from the exit to the start task, forming a request and allocate path along the DAG in *topological order*. As a result, all tasks are allocated without duplication.

The ordering of tasks has a time complexity of  $O(V + E)$ , where  $V$  is the number of tasks, and  $E$  is the number of the communication edges between tasks. Task assignment requires  $O(P(V + E))$  to find the *blevel*, and then  $O(V \cdot P)$  to choose the assignment destination from that, where  $P$  is the number of processors. Therefore, the total time complexity is  $O(V + E + P(2V + E))$ . In dense DAGs where  $E = V^2$ , this becomes  $O(V^2P)$ —matching the complexity of standard algorithms. Hence, our decentralized method achieves comparable scheduling efficiency.

### B. Bundling Unresolved Interest

To preserve task dependencies in the proposed method, unresolved requests must be bundled into Interest packets and forwarded, ensuring each task receives all necessary predecessor results. A concern is increased latency due to longer Interest and data return paths, as the one-stroke path detours to avoid duplicate assignments. Unlike the multicast-like method, where delays follow the DAG’s critical path, our approach traverses all tasks, potentially increasing delay.

## III. NUMERICAL RESULTS AND DISCUSSION

We evaluated the proposed method in 200 times using ICN-based SFC workflow simulator. The experiment confirms that task duplication is consistently avoided and compares application Turnaround Time. For comparison, we also tested the multicast-like method, which shares the same task assignment but lacks topological ordering and request control. The test environment included a random network with 100 edge nodes and 500 intermediate router nodes. Node bandwidth ranged from 100 to 1000 MBps, and processing capacity from 2000 to 4000 MIPS. Applications consisted of random DAGs with 20 tasks, and we varied the communication-to-computation ratio (CCR) from 0.1 to 20 to assess performance.

Figure 2 shows the average number of requests as the number of tasks increases to deepen the DAG hierarchy. The proposed method issues only one request per task, avoiding duplication, whereas the multicast-like method generates significantly more requests as the hierarchy deepens.

Figure 3 shows the average turnaround time and its breakdown for a 20-task application. Turnaround time, defined as the duration from request submission to result return, increases with task count in both methods. However, the multicast-like method shows longer delays in the request phase, while the proposed method exhibits extended return phases.

This is due to the multicast-like method producing a large number of Interest packets, leading to longer queues for task assignment decisions. In contrast, the proposed method

forms a one-stroke request path, increasing path length. While our method reduces turnaround time by avoiding duplicate requests, it trades off some parallelism. Thus, performance may vary depending on the experimental environment.

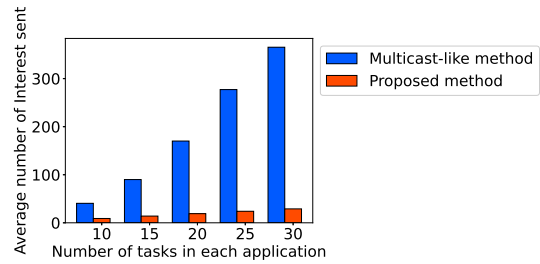


Fig. 2. Average number of requests sent

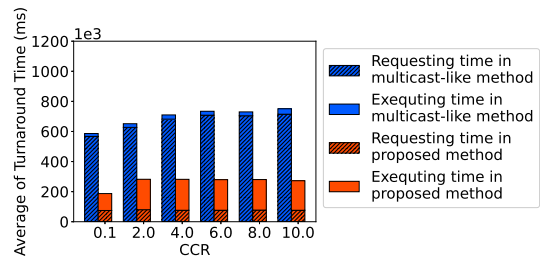


Fig. 3. Average turnaround time with 20 tasks

## IV. CONCLUSIONS

In this paper, we propose a method for step-by-step task scheduling and one-stroke formation using a ReadyList to avoid duplicate task assignments caused by the FNJ problem, as well as a method for bundling and transferring unresolved interests. Computer simulation results confirmed that the proposed method prevents duplicate task assignments. The proposed method was found to be effective in reducing turnaround time by preventing duplicate requests. However, since avoiding duplication involves a trade-off with parallelism, different results may be obtained depending on the experimental environment. Therefore, we plan to conduct experiments under more various network environments to clarify this trade-off relationship and conduct detailed analysis in the future.

### ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number 23K21664, 23K21665, and 23K28078.

### REFERENCES

- [1] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, “Fog computing and its role in the internet of things,” in *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, ser. MCC ’12. New York, NY, USA: Association for Computing Machinery, 2012, p. 13–16.
- [2] S. Al-Sarawi, M. Anbar, R. Abdullah, and A. B. Al Hawari, “Internet of things market analysis forecasts, 2020–2030,” in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*, 2020, pp. 449–453.
- [3] C. Tschudin and M. Sifalakis, “Named functions and cached computations,” in *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, 2014, pp. 851–857.
- [4] L. Liu, Y. Peng, M. Bahrami, L. Xie, A. Ito, S. Mnatsakanyan, G. Qu, Z. Ye, and H. Guo, “Icn-fc: An information-centric networking based framework for efficient functional chaining,” in *2017 IEEE International Conference on Communications (ICC)*, 2017, pp. 1–7.