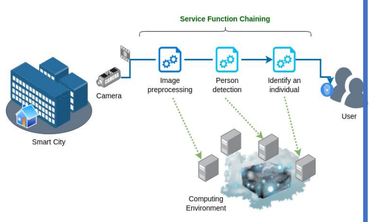
Task Scheduling with Duplication Avoidance for ICN-based Autonomous In-Network Computing

Yuki Niibe, Noriaki Kamiyama (Ritsumeikan Univ., Japan)

1. Introduction

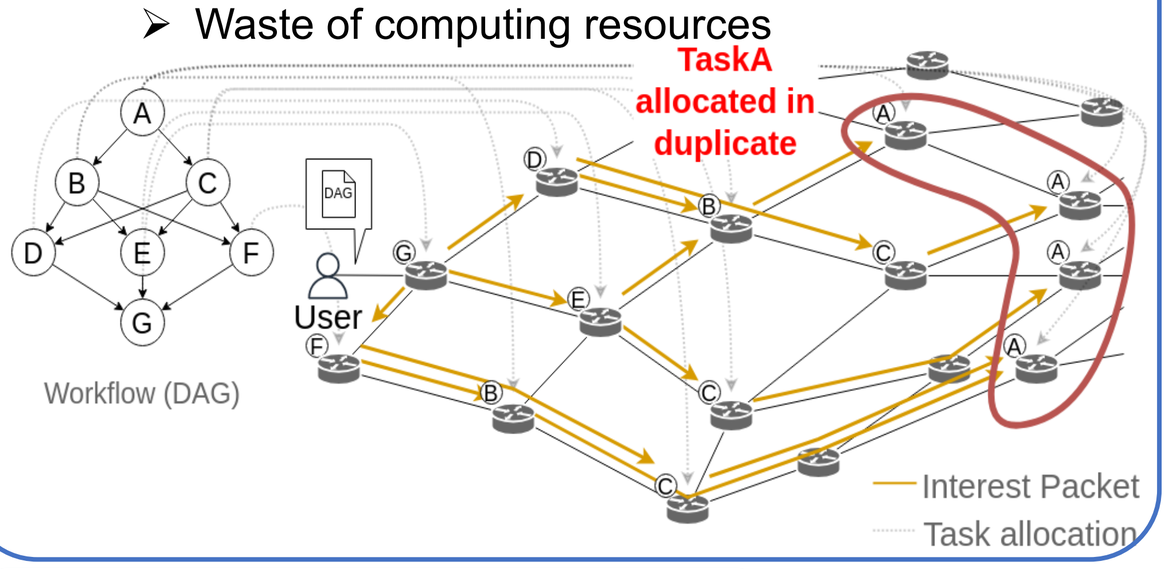
- The rise of IoT and the arrival of new applications
  - The emergence of **data-intensive applications** utilizing data obtained from IoT devices, **IoT Big Data (IoTBD)**
  - An environment capable for such requests is necessary
- Processing a wide variety of requests for IoTBD
  - Common Method
    - Processing data on a remote cloud
      - Network edge <-> Network core
      - Inefficiency in data transfer distance
  - **In-network computing**
    - Processing data while transferring from the edge to the core network
- Key technology in In-network computing
  - Service Function Chaining (SFC)
    - Method for linking deplyed functions
    - Used to link virtualized network functions (VNF).
    - Adaptable to various tasks other than network functions



- **Centralized management** v.s. **Autonomous management**
  - Traditionally, a central server manages the execution and linking of each task.
    - Considering the future increase in IoT devices, it will become necessary to handle devices that are more widely distributed throughout society
      - Limits to centralized management
  - **Autonomous In-network computing**
    - Information-centric networking + Service Function Chaining

2. Challenges in ICN-based Autonomous INC

- Duplicate task execution
  - Each node determines the destination for sending Interest to preceding tasks from its own routing table (FIB)
  - Even for the same task, Interest may be sent to different locations ➡ **Duplicate allocation / Duplicate execution**
    - ➢ Waste of computing resources

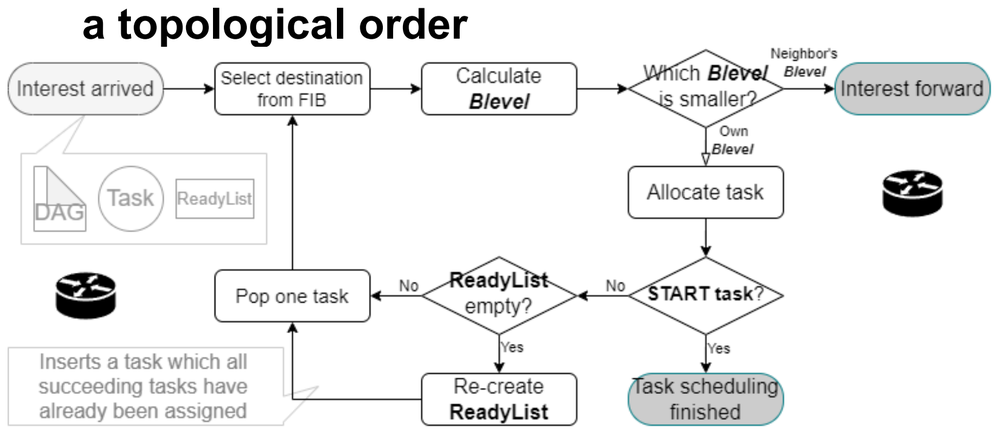


3. Purpose of This Work

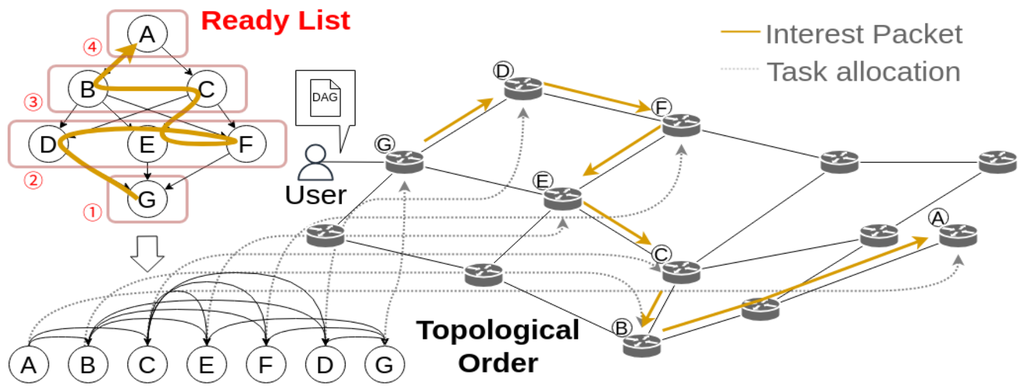
- Autonomous ICN-based In-network computing(ICN + SFC)
  - ➢ Avoiding duplicate task execution
    - ✓ Algorithm for avoiding duplicate task execution
    - ✓ Without destroying dependencies necessary for task execution

5. Proposed Method

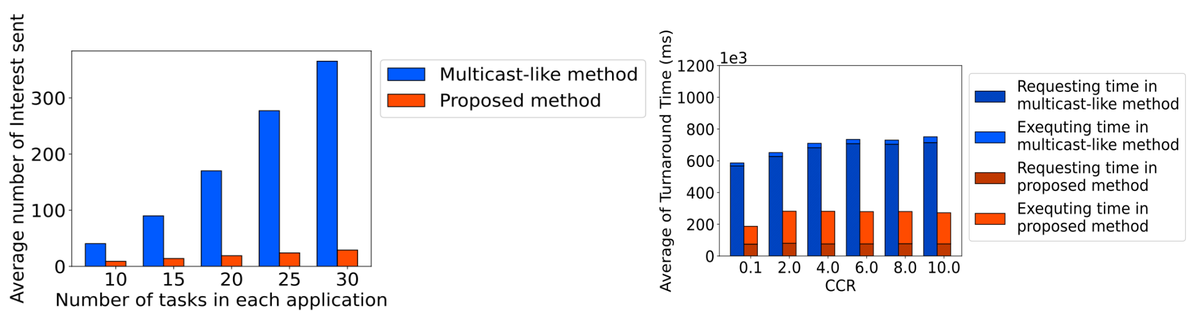
- Determining the assignment order to avoid duplication
  - Creating a **Topological Order**
    - Parallel task assignments will result in duplicate
    - Assignment operations should be performed in a one-stroke order, not parallel
    - Serialize the DAG ➡ **Topological Order**
  - Creating a Topological Order using **ReadyList**
    - **ReadyList** : A set of tasks for which all successor tasks are allocated
  - When sending interest to predecessor task, select from ReadyList.
    - **Tasks can be assigned one by one while creating a topological order**



- Deciding where to assing tasks
  - Automatically determine assignment destinations by calculating **blevel**
    - Each time an Interest arrives at a node, calculate the **blevel** for assigning tasks to that node and its neighbor nodes
      - **blevel** : Estimated longest remaining time of task
  - Select the node with the smaller blevel from among the node and its neighbor nodes



6. Evaluation



- Duplicate assignments could be prevented
- Previous method : long task assignment phase
  - Causes queuing delays because it performs a large number of assignments
- Proposed method : fast assignment phase
  - the execution phase is slow
  - By assigning tasks in a one-stroke, the distance between each task increases
- More optimal allocation is needed while avoiding duplication

This presentation is supported by ASTER