ICNを用いたネットワーク内処理における重複割当を回避したタスクスケジューリング

立命館大学

新部裕樹 上山憲昭

目次

- 研究背景
- 先行研究
- 課題と研究目的
- 提案手法
- 実験
- 実験結果
- 評価
- まとめ

研究背景

5G等の高速大容量ネットワークの実現

⇩

ネットワークに繋がる“モノ”の増加

(IoTデバイスの普及，より多くのモノ・コトが情報化)

⇩

多種多様な処理を実行させることへの需要増加

- 例：スマートシティに配備されたIoTデバイス
  - IoTデバイスからセンシングされる膨大なデータを連携しながら分析　➡ビッグデータの活用
  - 新たなサービス・ビジネスの創出

研究背景

- 多種多様な処理の実行
  - IoTデバイスからセンシングされるデータ
    - これらのデータは遠隔地のクラウド上で処理される

IoTデバイス：ネットワークのエッジ
データの収集
⇩
クラウド：ネットワークのコア
データの処理

- 効率性に問題
- ネットワーク
  トラヒックの圧迫

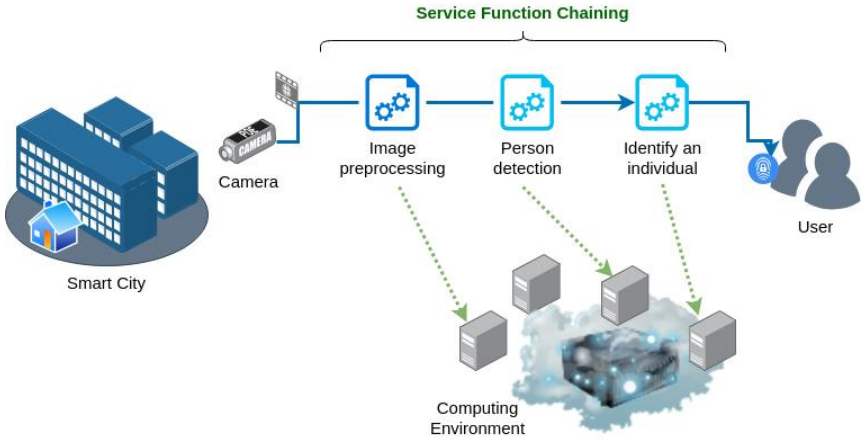
➡ネットワーク内処理[1]の適用

ネットワークのエッジからコアに向かってデータを転送させていく中で，処理も同時に行う

研究背景

- 多種多様な処理の連携: ネットワーク内処理を実現するために
  - **Service Function Chaining** (SFC)
    - ネットワーク仮想化における，機能の連携に用いられる技術
    - ネットワーク機能に限らず，様々な機能(タスク)にも適用できると考えられる

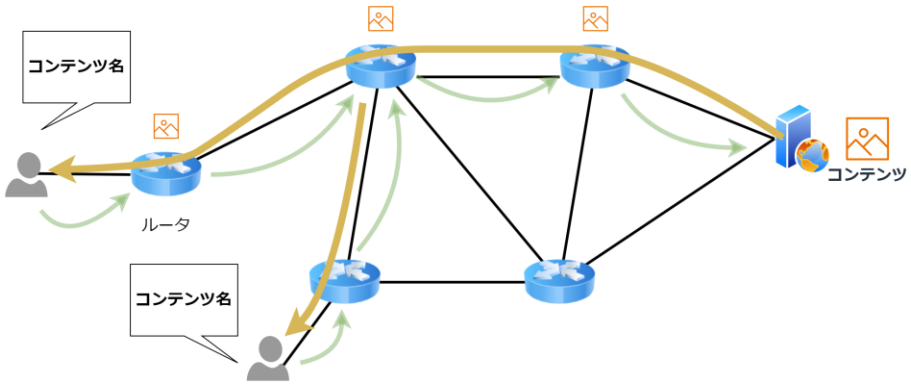
➡様々な処理を連携できる



研究背景

- 多種多様な処理の連携: ネットワーク内処理を実現するために
  - **情報指向ネットワーク**[2,3] (Information-Centric Networking: ICN)
    - 元々は効率的なコンテンツ配信のための手法
      - コンテンツの場所(e.g. IPアドレス)に依存しない，コンテンツ名でのやり取り
      - ネットワーク内キャッシュの利用

➡効率的で柔軟なやり取りができる
➡処理結果をキャッシュし再利用することで
計算資源を節約

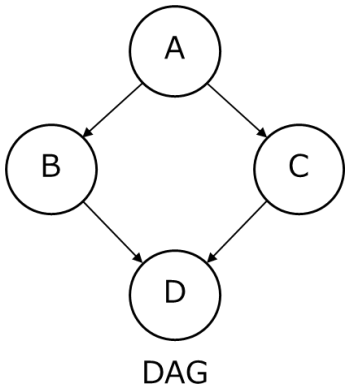


研究背景

- 多種多様な処理の連携: ネットワーク内処理を実現するために
  - **ICN**を組み合わせた**SFC**が有効 (**ICN-SFC**)
    - 効率よく柔軟に処理を連携し実行させることができる

先行研究

- 多種多様な処理の連携: ネットワーク内処理を実現するために
  - 様々なアプリケーション構造に対応する必要がある
    - 昨今のアプリケーション構造の複雑化
    - アプリケーション構造を一般化すると**DAG**(有向非巡回グラフ)になる
  - **DAG**アプリケーションを想定したICN-SFC[4]
    - DAGアプリケーションの処理フロー→ワークフロー

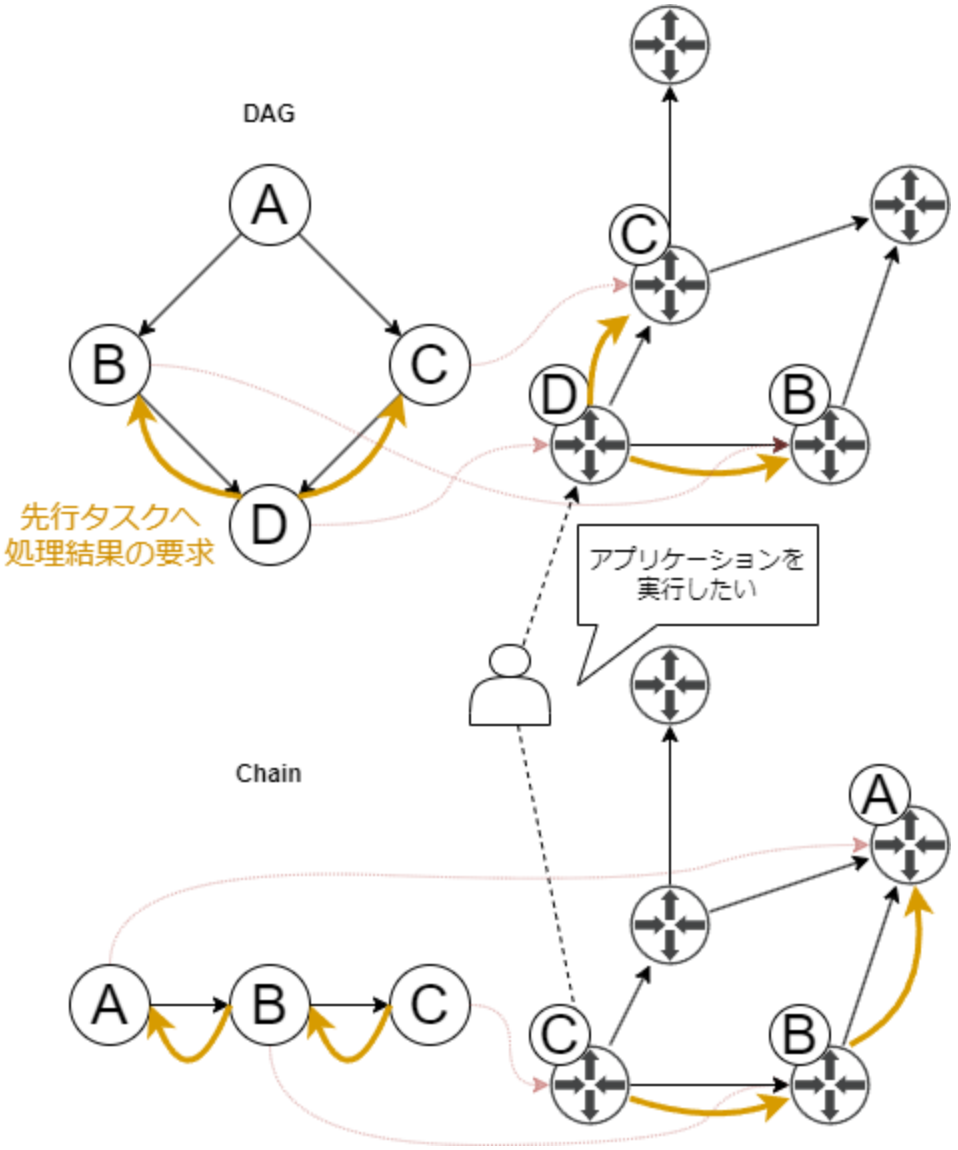


DAG

先行研究

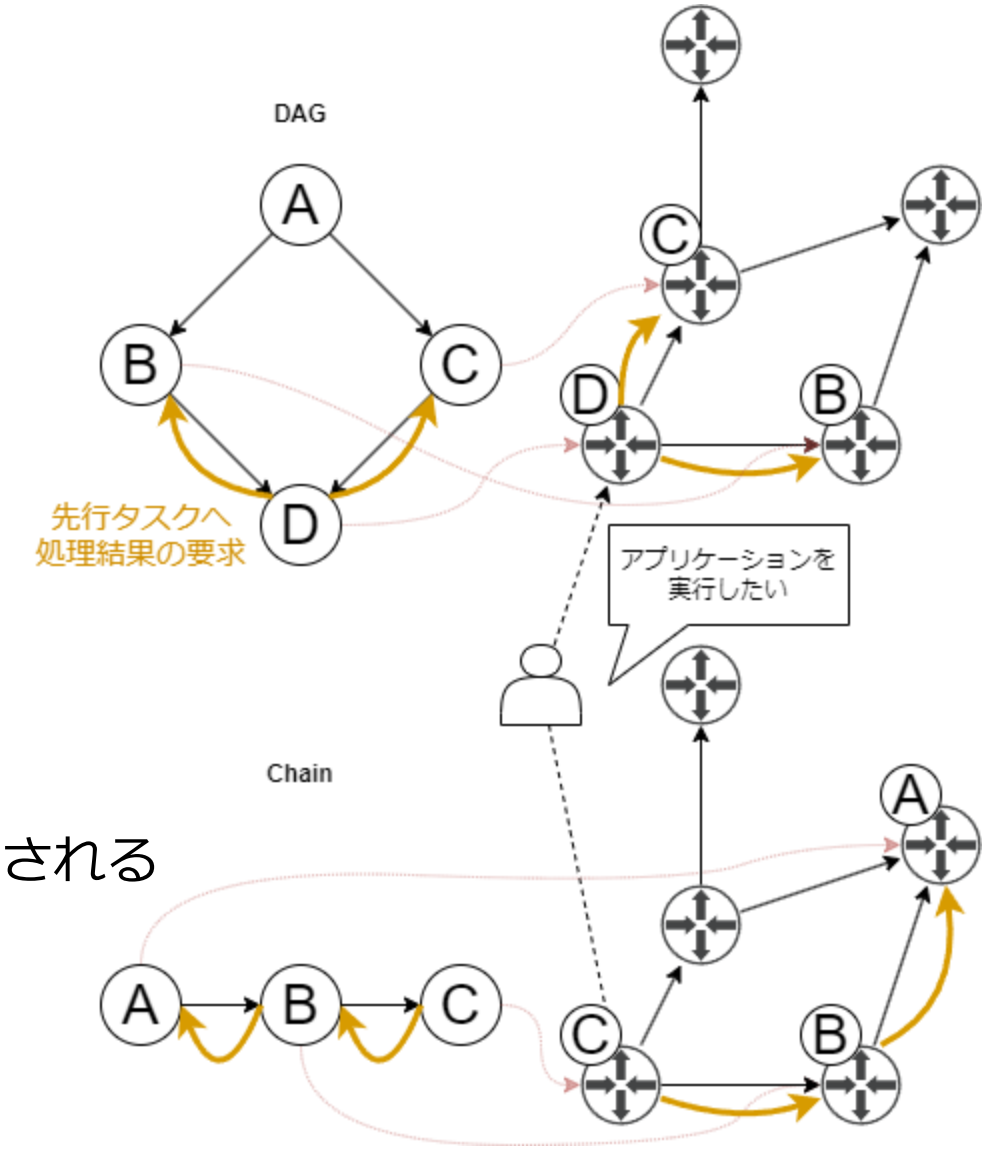
- ICN-SFCでの処理要求と
  タスクの割当
  - 後続タスクから先行タスクに向かって
    処理結果の要求を行う

➡タスクが順に割り当てられていく
先頭のタスクまで辿り着いたら実行開始



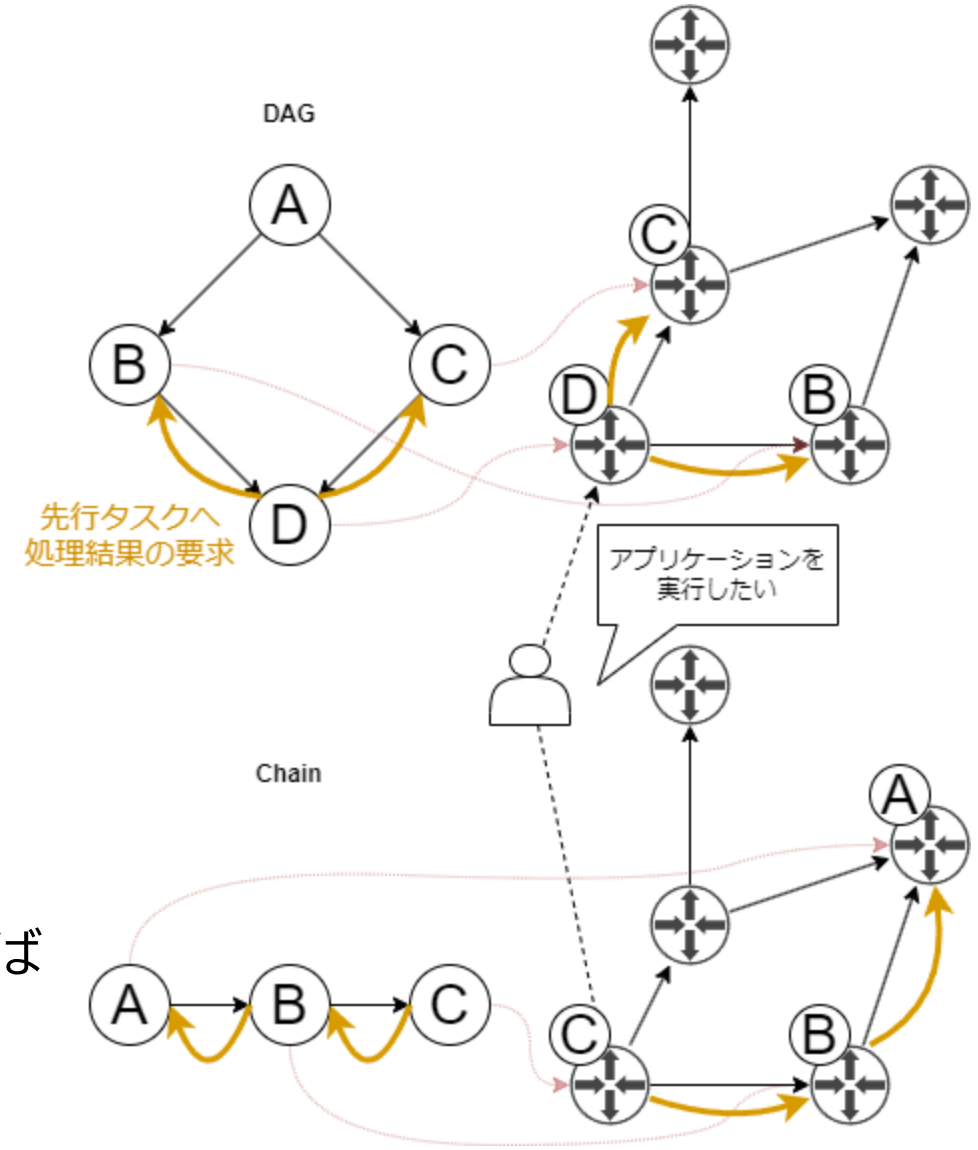
先行研究

- ICN-SFCでの処理要求と
  タスクの割当
  - タスクへの要求送信先は
    各ノードのFIBから選択される
    - FIBはそのタスクを保持するノードが
      ネットワーク全体に広告することで設定される



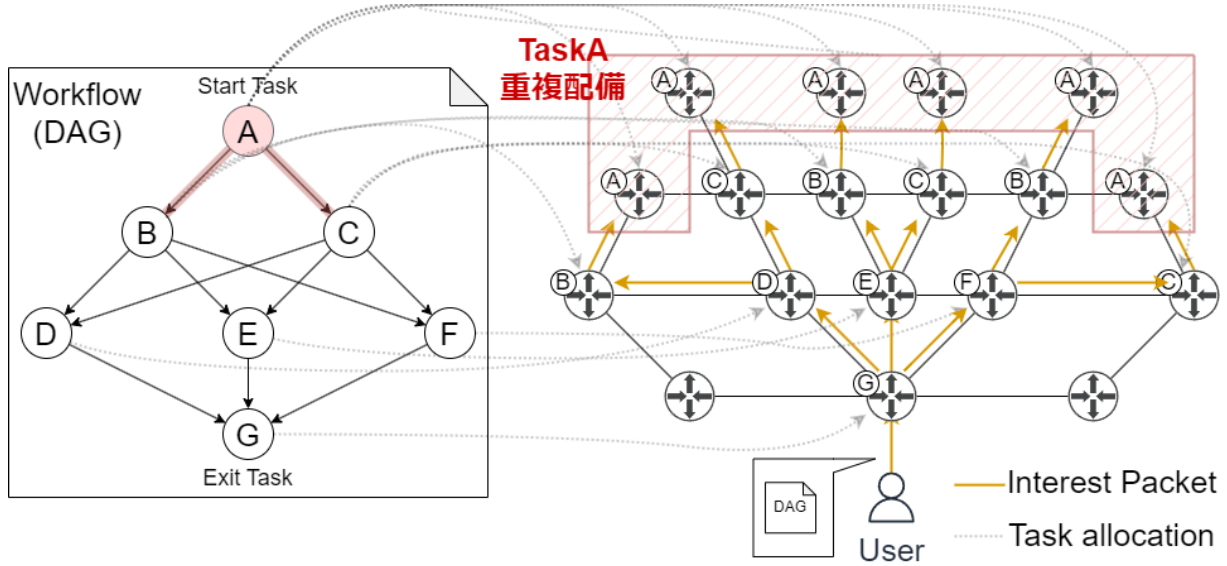
先行研究

- ICN-SFCでの処理要求と
  タスクの割当
  - タスクの割り当ては，そのタスクの
    実行場所として適しているかどうか
    毎回判断することで行われる
    - FIBに従って要求が転送されていく中で，
      タスクの実行に適したノードが見つかれば
      そこにタスクが割り当てられる



課題と研究目的

- ICN-SFCでワークフローを実現する際の問題点
  - タスクの重複配備・冗長実行が発生してしまう
    - 複数のタスクが
      同一の先行タスクを持つ
    - 各ノードの経路表から
      先行タスクの要求先を決定
    - 自立分散処理のため
      要求送信先を同期できない



➡異なるノードに要求が到達し，タスクが重複配備される

課題と研究目的

- 本研究の目的
  - タスクの重複割当を回避した，タスクスケジューリングを実現する
    - タスクの重複割当を防ぐことで，タスクの冗長実行を防ぎ
      無駄な計算資源の消費を減らす

提案手法

- タスクの重複割当を防ぐために
  - DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する
    - 既存手法では先行タスクに対して同時に要求を送信し，その先々で
      それぞれが割当を行っていた
      ➡重複割当
    - DAG内のタスクを段階的に順序付けしていくことで，
      一筆書きのような要求・割当パスを作成し，全てのタスクを必ず１つずつ割当
      ➡重複せずに割当

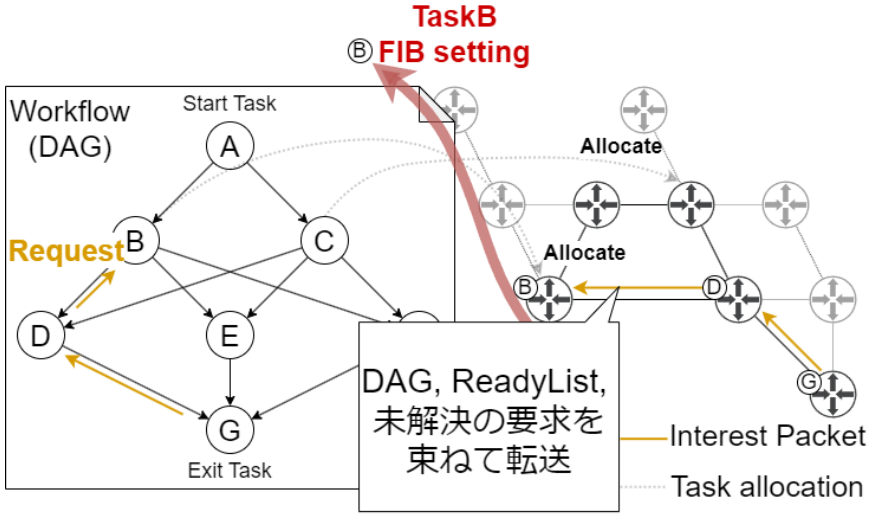
提案手法

- タスクの重複割当を防ぐために
  - DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する
    - **ReadyList**を導入
      - ReadyList：その時点で未割当のタスクのうち，全ての後続タスクが割当済となっているタスクの集合
    - ReadyListの作成と，その中から１つずつ要求し割当先の決定を繰り返す
      ➡必ずDAG内の全てのタスクをスケジュールできる

提案手法

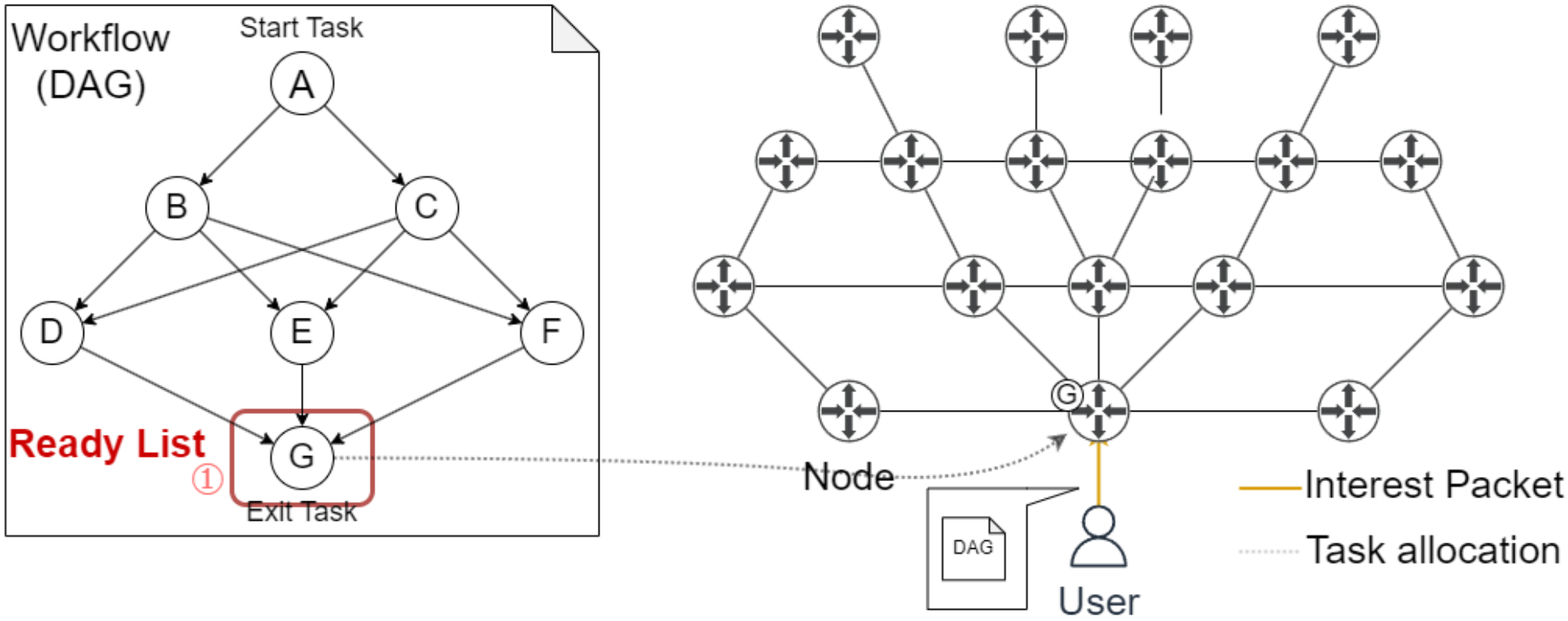
- タスクの重複割当を防ぐために
  - DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する
    - DAG(アプリケーション構造), ReadyList, 未解決の要求を全てInterestパケットに束ねて転送する

➡ICN-SFCのような自律分散処理でも，
各ノードで自律的にタスクの段階的な
順序付け・一筆書き要求パスの形成
を行うことができる



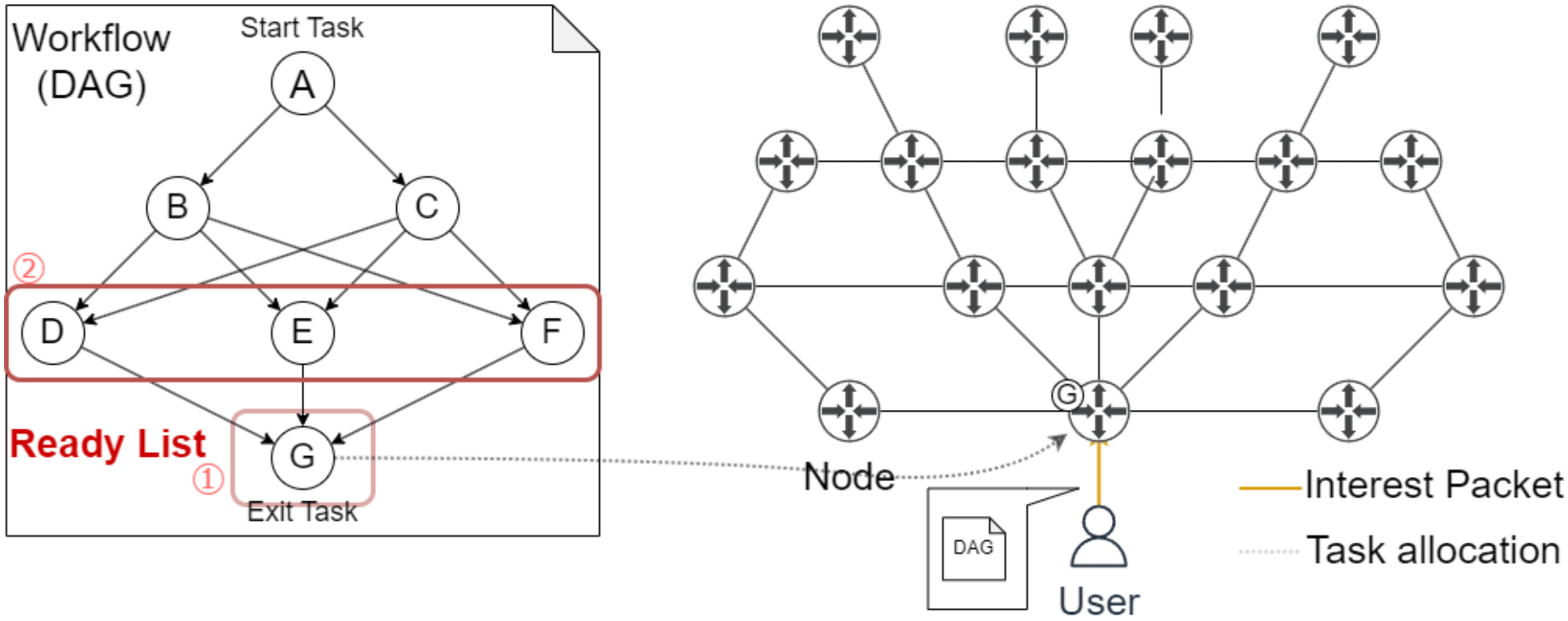
提案手法:動作

- DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する



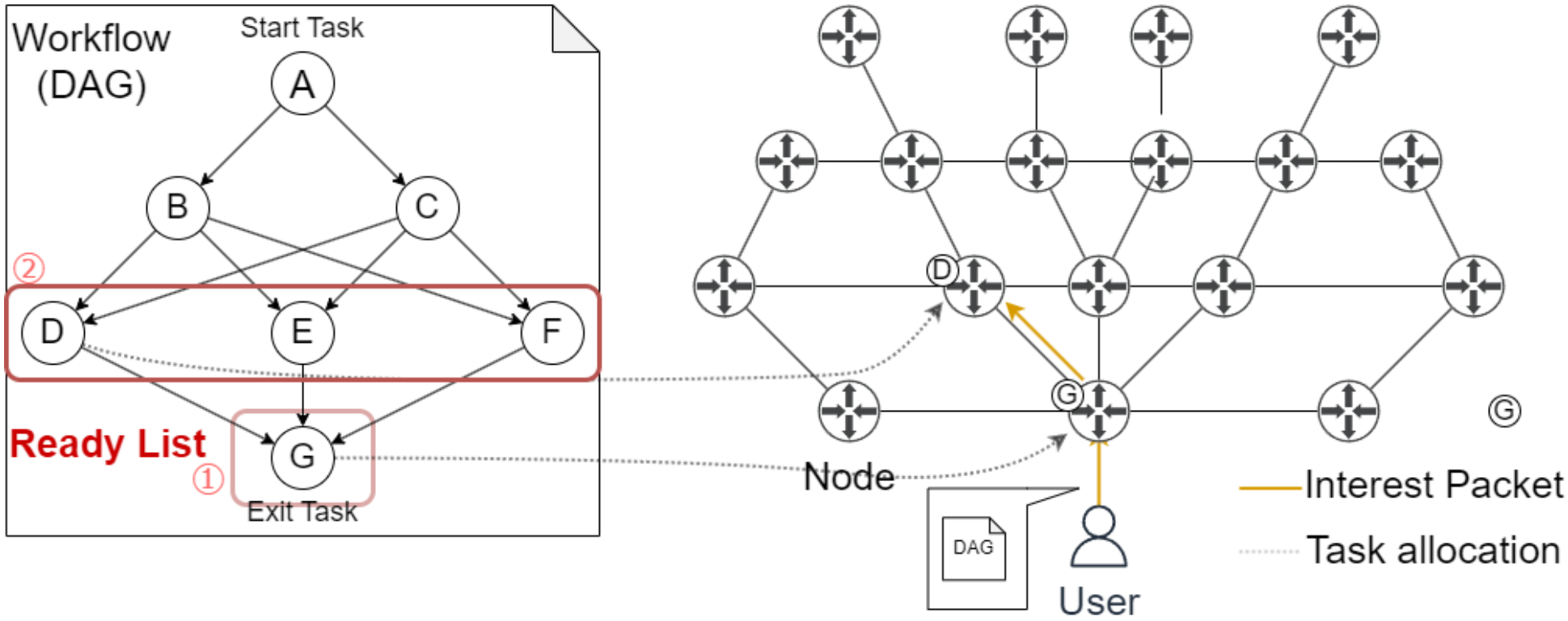
提案手法:動作

- DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する



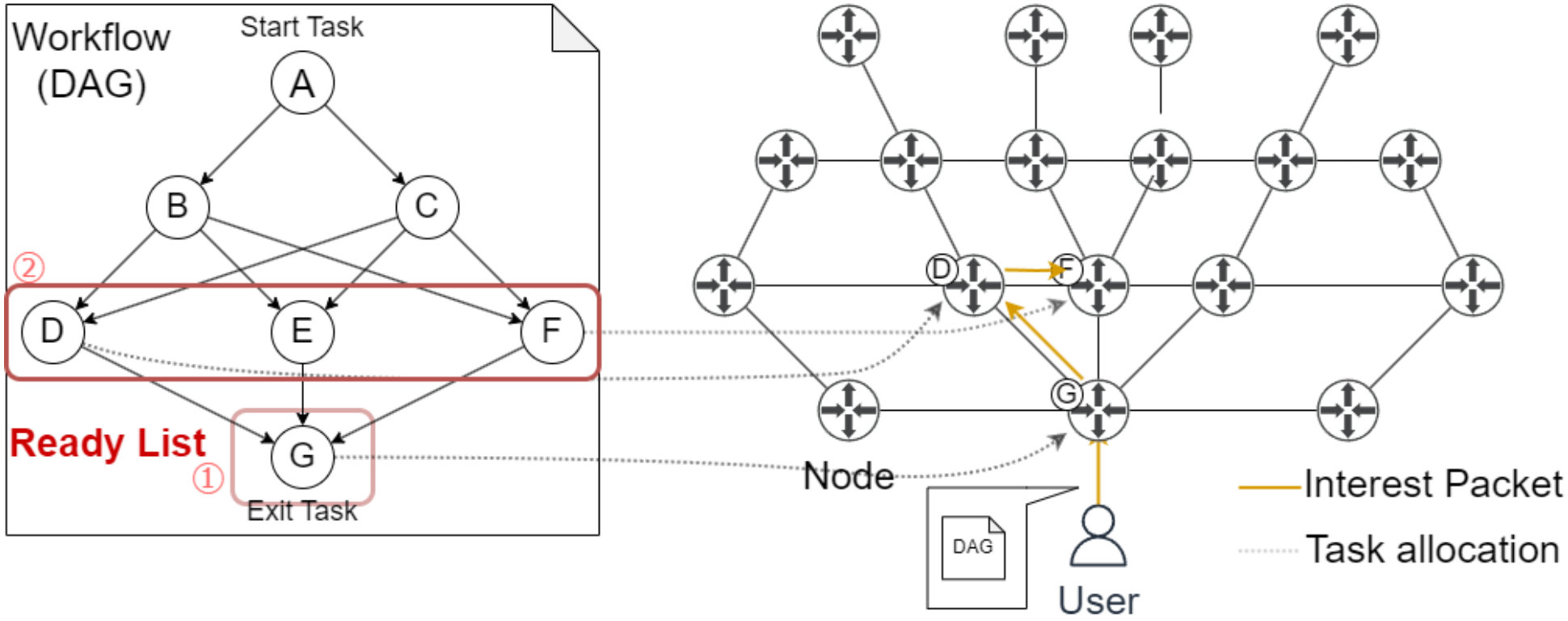
提案手法:動作

- DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する



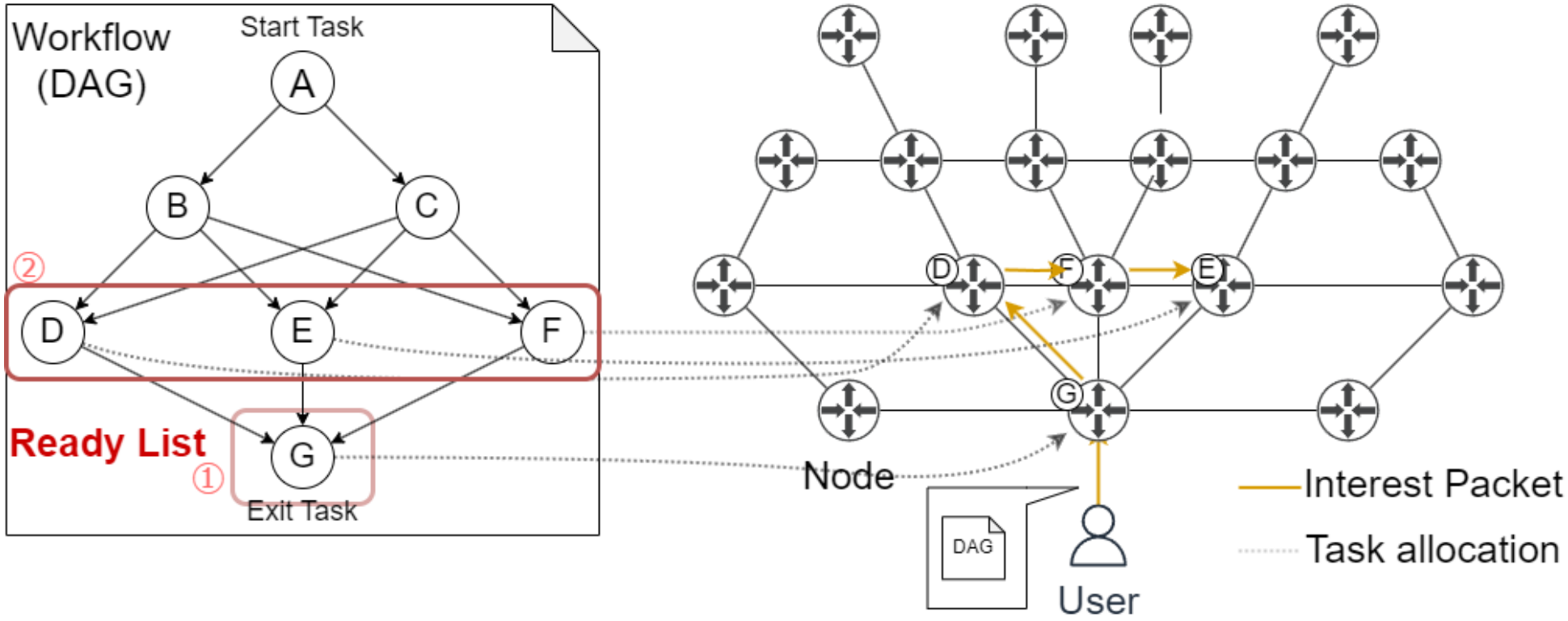
提案手法:動作

- DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する



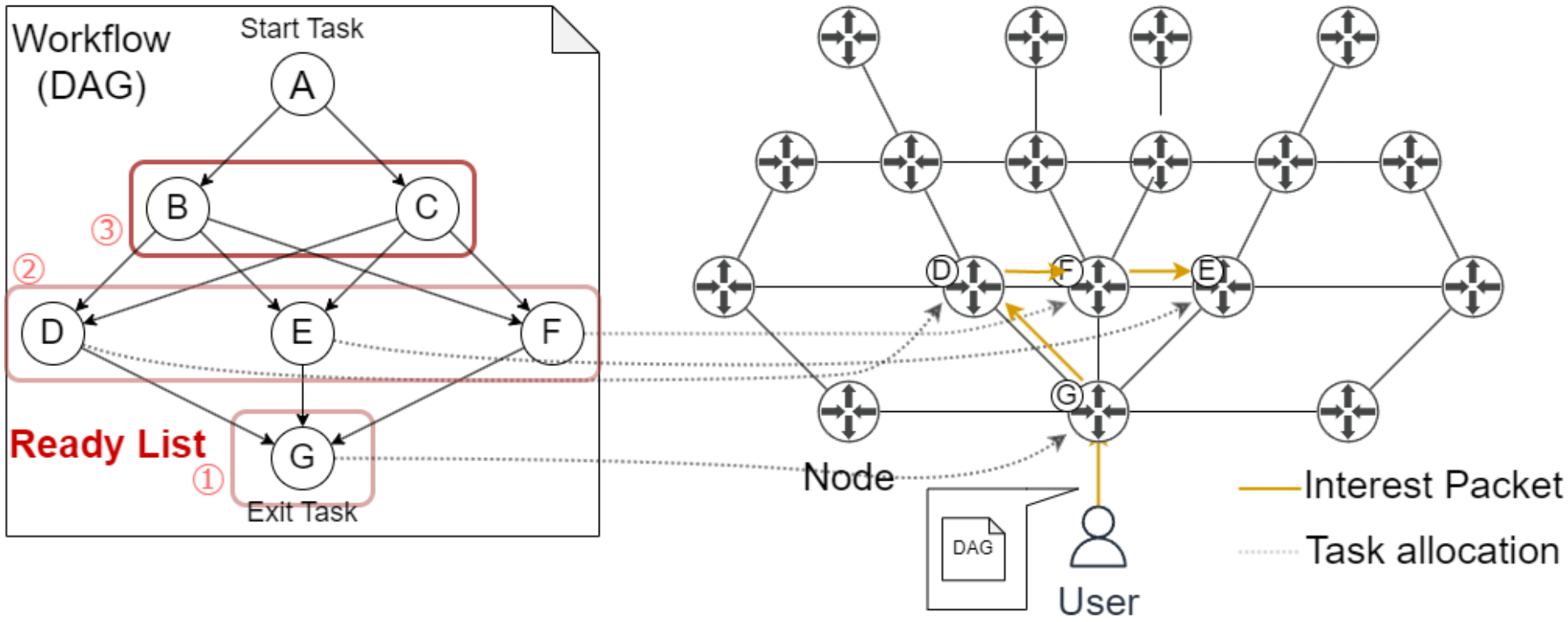
提案手法:動作

- DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する



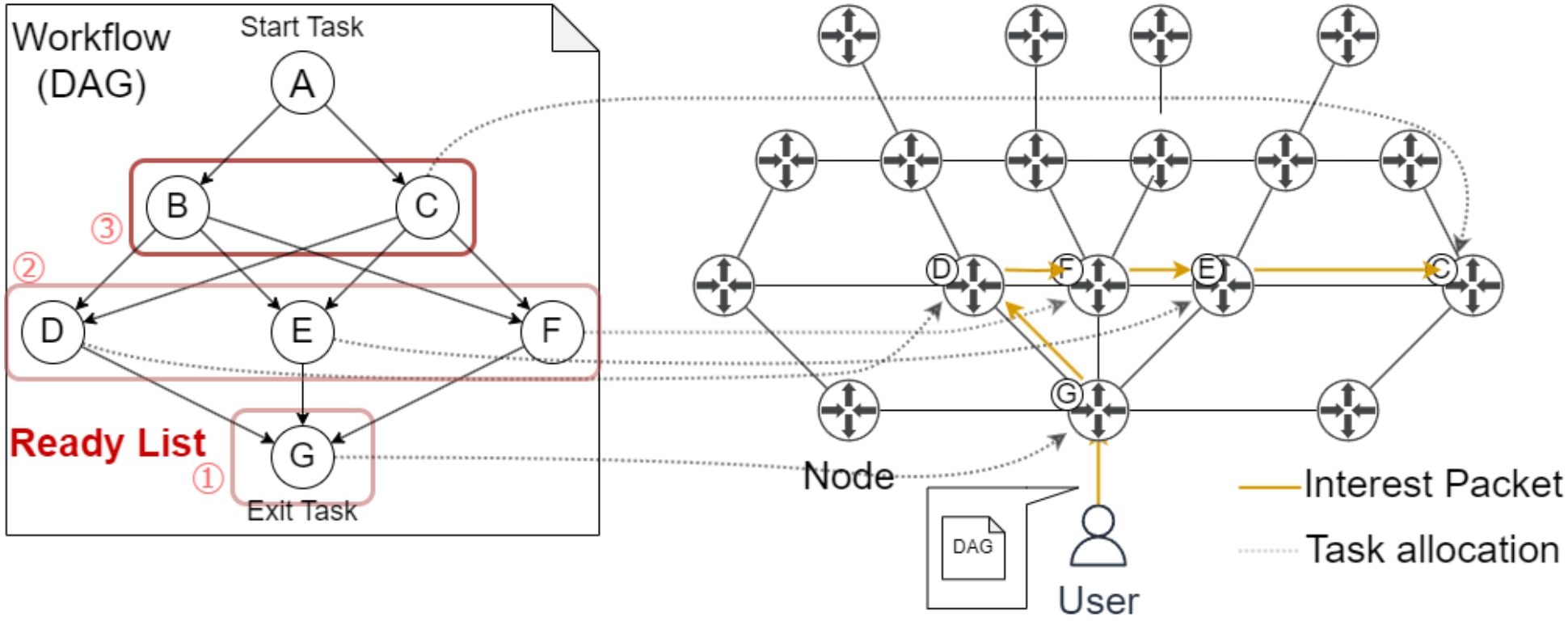
提案手法:動作

- DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する



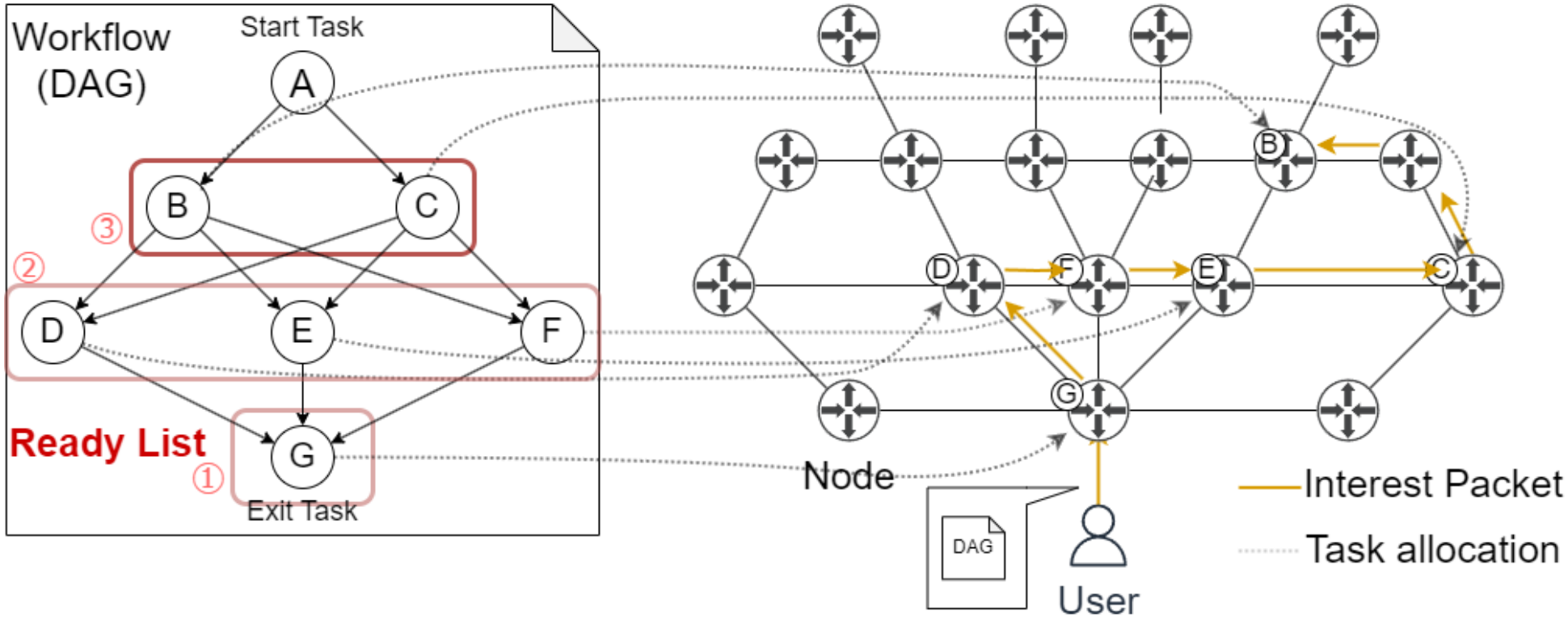
提案手法:動作

- DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する



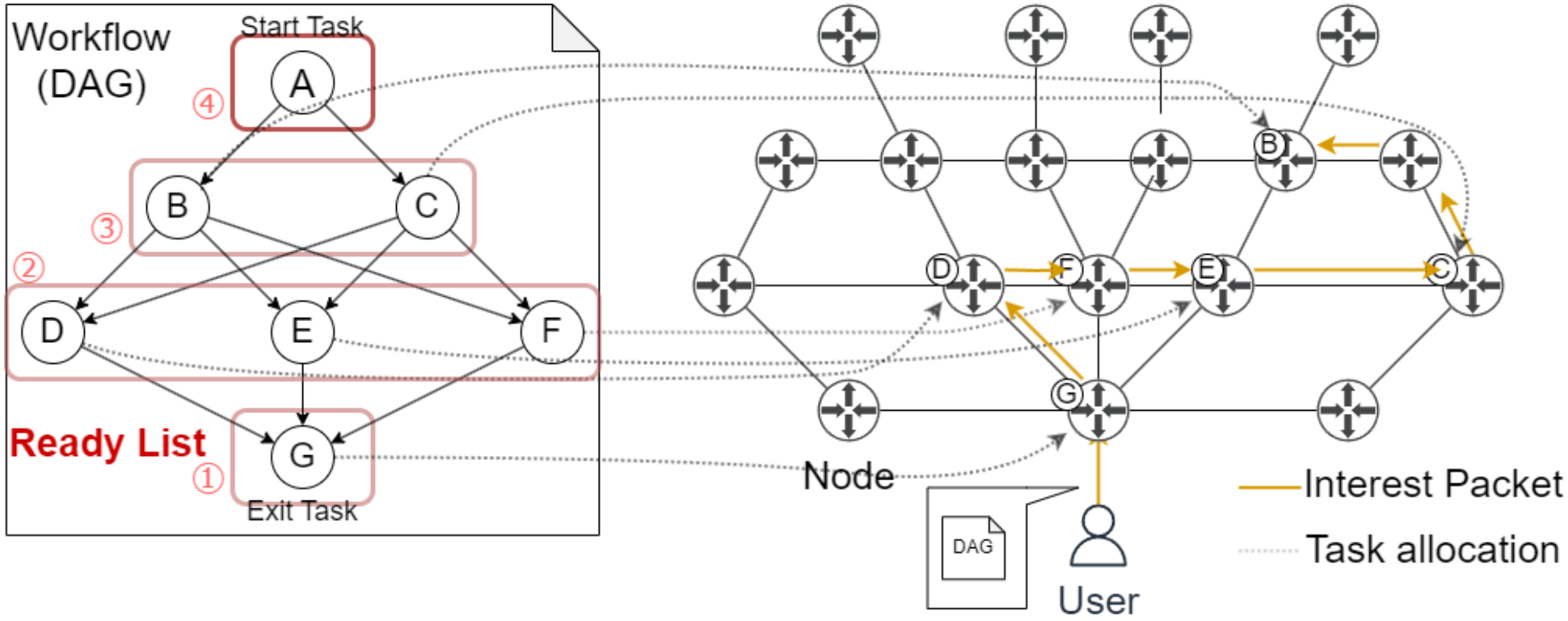
提案手法:動作

- DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する



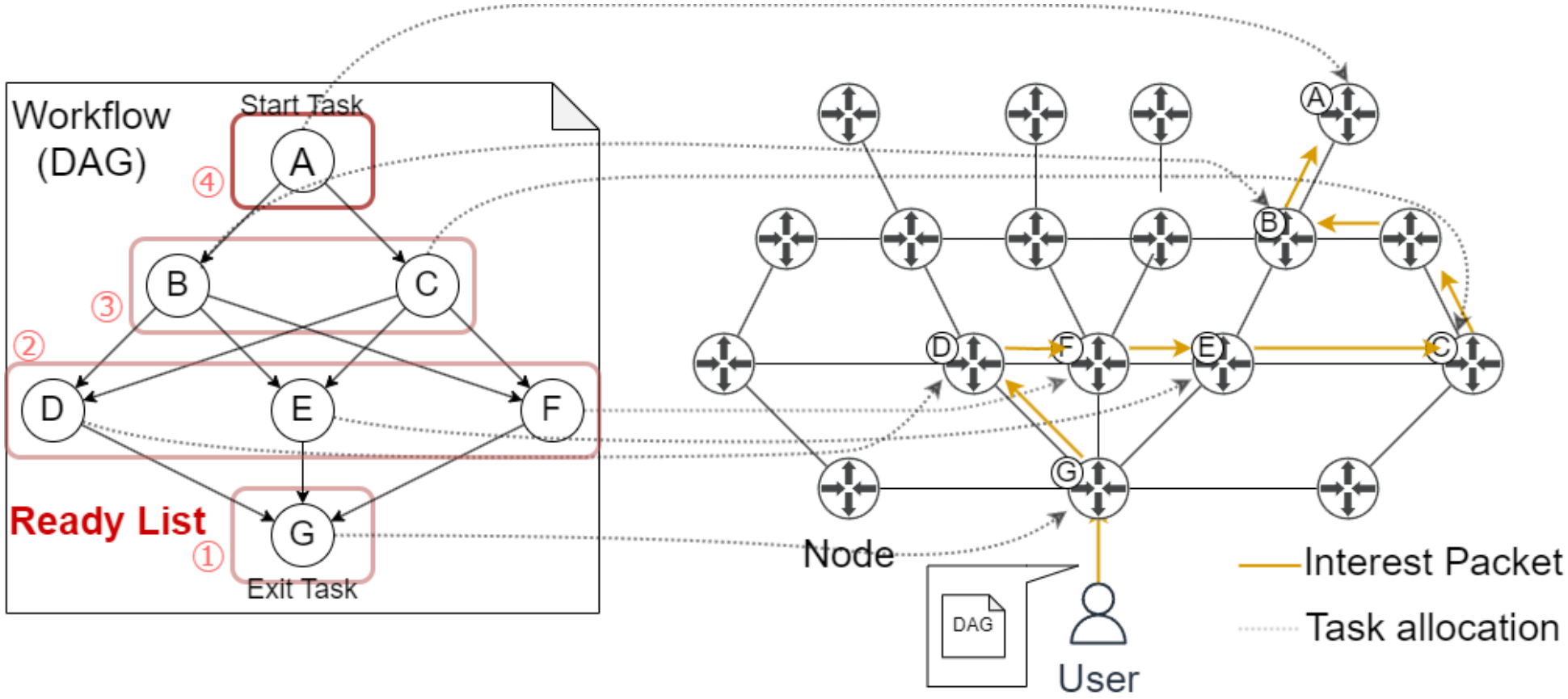
提案手法:動作

- DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する



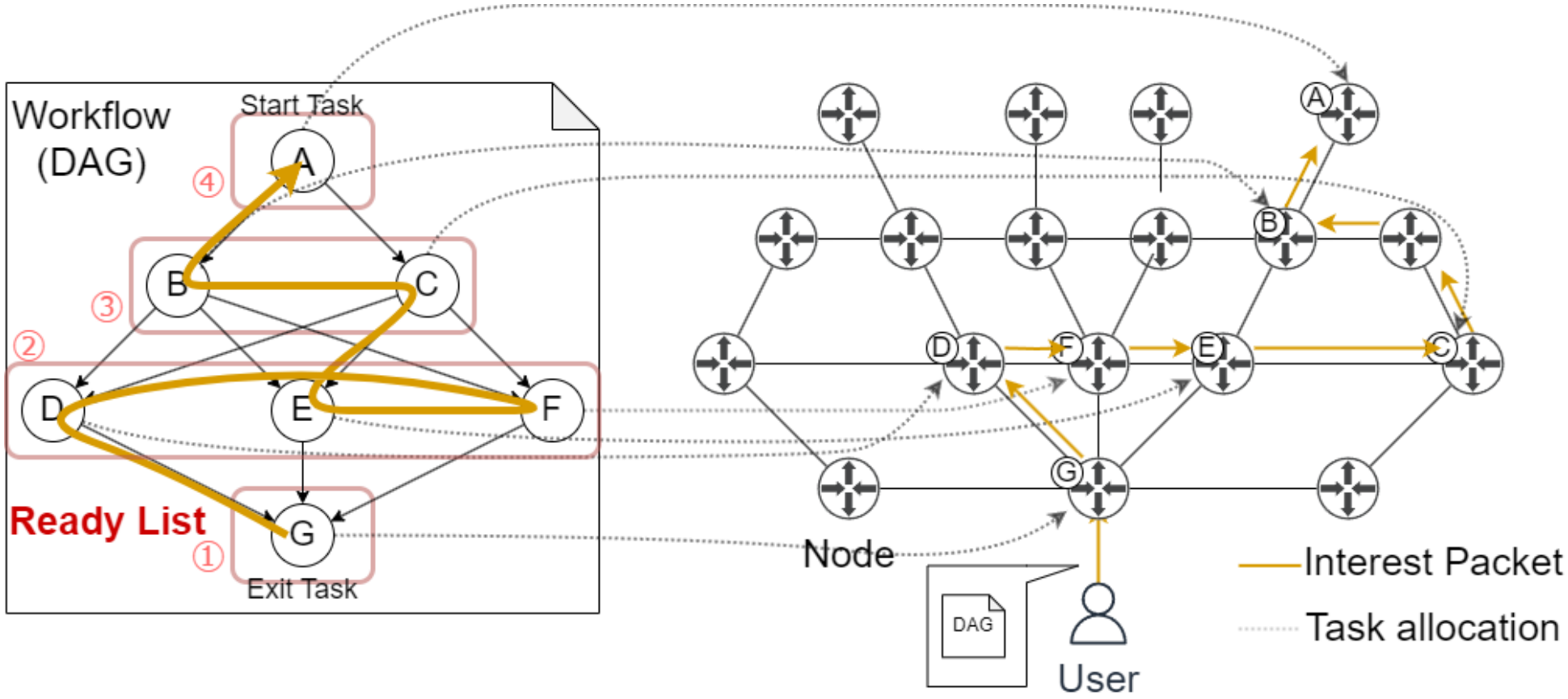
提案手法:動作

- DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する



提案手法:動作

- DAG構造上を一筆書きのように順序付け，１つ１つ割当先を決定する



実験

シミュレーション

- ランダムなDAGを生成し，ランダムにネットワークに投入する

- 複数回実行し平均することで，どのような実行結果になるか比較

  - タスクの要求回数
    - 提案手法によって重複割り当てが防げているか比較

  - アプリケーションのTurnaround timeを比較
    - Turnaround time内訳：要求フェーズ + 実行フェーズ
    - 既存手法と提案手法ではアプリケーションの実行時間にどのような変化があるか比較

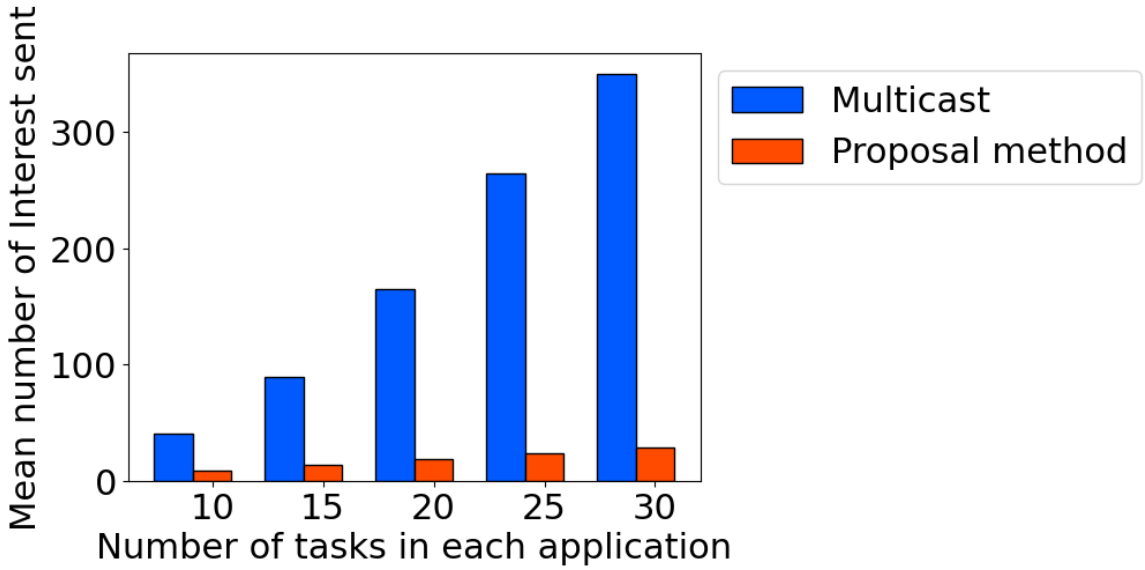
実験

- 実験環境
  - ICN Router 500, ユーザー 100
  - 同時投入するアプリケーション 5

- ランダムDAG
  - 各タスクの最大後続タスク数
  - アプリケーション内のタスク数 10~30
    - DAGの階層を変化することで，タスクの重複割当の変化とその影響を確認
  - 各タスクのCCR　　0.1~10.0
    - アプリケーションの特徴を変化させることで，どのような影響があるか確認

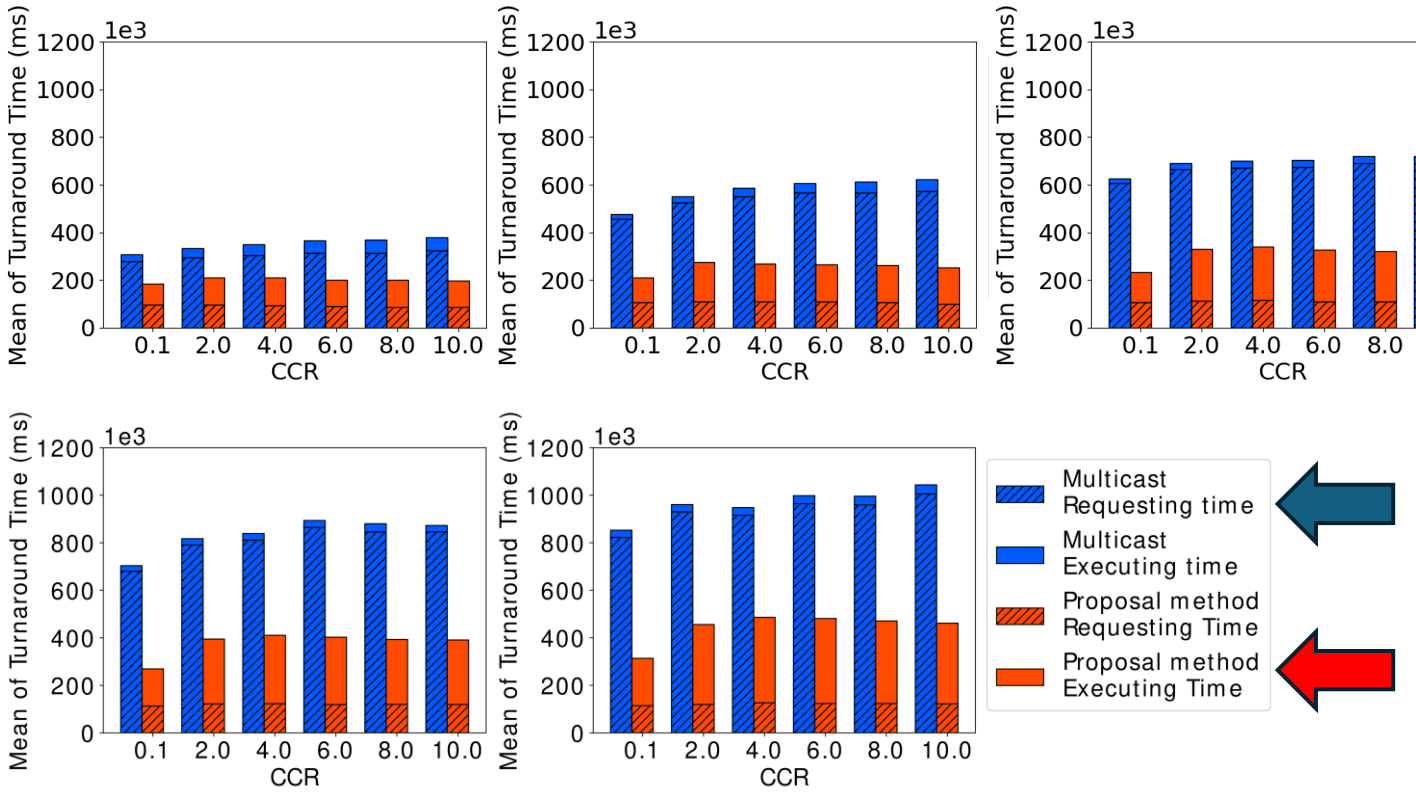
実験結果

- タスクへの要求回数 (=タスクの割当回数)
  - 既存手法はタスク数が増えるにつれて重複割当が大幅に増加
  - 提案手法は重複割当を防げている



実験結果

- Turnaround timeの変化
  - 全体的に提案手法の方が高速
  - 既存手法は要求フェーズの時間が支配的
    - 重複割当がネットワークに負荷を与えているため
  - 提案手法は実行フェーズの時間が支配的



評価

- 提案手法は重複割当を防げている
- 既存手法は重複割当/重複要求によってネットワークに大きな負荷を発生させる
  - アプリケーションの実行時間に大きな影響あり
- 提案手法は一筆書きによって実行フェーズにおいてデータの返送に遅延が発生

まとめ

- ICN-SFCでDAGアプリケーションを実行すると，タスクの重複割当が発生

- 提案手法によってタスクの重複割当を防げる
  - 提案手法の方が全体として高速
  - 一方で，実行フェーズが遅い
    - 実験環境やアプリケーションによっては良い結果にならない可能性

- 今後の展望
  - 提案手法と既存手法について，どのような環境/アプリケーションが適しているのか詳しく実験し，その境界を明らかにする

参考文献

1. Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh 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 (MCC ’12), New York, NY, USA, 2012, pp. 13 - 16
2. Van Jacobson, Diana K. Smetters, James D. Thornton, Michael F. Plass, Nicholas H. Briggs, and Rebecca L. Braynard. ”Networking named content.” In Proceedings of the 5th international conference on Emerging networking experiments and technologies (CoNEXT ’09). Association for Computing Machinery, New York, NY, USA, 2009, pp. 1 - 12.
3. Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Be- ichuan Zhang. 2014. Named data networking.
4. 金光永煥・花田真樹・金井謙治・中里秀則, ワークフローにおけ る ICN を用いたファンクションチェイニング, 信学技報, vol.119, no. 461, IN2019-120, pp. 249-254, 2020 年 3 月