Dynamic Flow Routing and Scheduling for Time-Critical Network Services

Xuanli Lin, Zhaofeng Zhang, Zunzheng Zhang, Guoliang Xue
Arizona State University, Tempe, AZ. Email: {xlin54, zzhan199, zzhan621, xue}@asu.edu

MILCOM 2025 - 2025 IEEE Military Communications Conference (MILCOM) | 979-8-3315-0292-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/MILCOM64451.2025.11310531

***Abstract*—A key challenge in next-generation networks is providing intelligent network services to support time-sensitive applications such as AR/VR and the Internet of Military Things (IoMT). This work aims to answer the question: "How can we design a network service scheduling and flow routing scheme to satisfy the stringent demand and deadline requirements of tactical network applications?" We take a time-expanded graph-based approach to this study. Specifically, given a time-expanded graph constructed from an original network graph, we propose an optimization problem to find a subset of network services to maximize their total completion rewards such that the data of all services in this subset can be transmitted to their destination nodes by their deadlines. Unfortunately, solving this problem directly is challenging since it is a mixed-integer program. We propose a heuristic algorithm based on a sequential rounding approach where a linear programming (LP) feasibility problem is iteratively solved to update the network service subset, guided by a relaxed reward maximization problem that prioritizes high-reward requests. To further improve efficiency, we fine-tune the above LP feasibility problem from a static flow perspective, enabling fast feasibility checks on the original graph, and design the corresponding efficient heuristic algorithm. The simulation results demonstrate the trade-off among different approaches and validate the effectiveness of the proposed algorithms.**

1. Introduction

Next-generation tactical edge networks are expected to support time-sensitive services such as augmented reality (AR), virtual reality (VR), and the Internet of Military Things (IoMT) [4, 7, 8], all of which demand stringent latency and reliability guarantees. In such environments, network services must be scheduled and routed promptly and resource-awarely to ensure end-to-end delivery within hard deadlines. Recent studies have explored deadline-aware scheduling frameworks to align network operations with service-level deadlines better [1, 3]. Nevertheless, many of these approaches rely on simplified models that ignore traversal delays, assume single-hop connectivity, or overlook bandwidth competition among multiple flows, which limits their applicability. Moreover, the joint problem of selecting a subset of services and routing them dynamically through a multi-hop network with time constraints remains computationally challenging due to its combinatorial nature and dynamic flow coupling. Motivated by these challenges, in this paper, we aim to address the following critical question: *"How can we design a dynamic flow routing and scheduling strategy that selects and fulfills the most valuable network services under strict deadlines and bandwidth constraints?"*

To answer the above question, we propose a novel dynamic routing and scheduling (**DRS**) framework based on the time-expanded graph representation. In this framework, a feasible offloading strategy of each service is modeled as a dynamic flow constrained by deadline, demand, and network capacities. The **DRS** objective is to maximize the total reward, subject to various constraints, by admitting the most valuable subset of services that can be completed on time. We formulate **DRS** as a mixed-integer program, and to overcome its computational intractability, we design a sequential rounding algorithm, where a linear programming (LP) feasibility problem is iteratively solved to update the network service subset, guided by the solution of the relaxed reward maximization problem. To further reduce complexity, we refine the LP feasibility problem from a static flow perspective, enabling fast feasibility checking using only the original network topology. This leads to a hybrid rounding algorithm that achieves near-optimal performance while offering significant runtime reduction. The main contributions of this paper are the following:

- We introduce a novel dynamic flow routing and scheduling framework for time-critical network services, based on a time-expanded graph model.
- We formulate the **DRS** problem as a mixed-integer program and develop an LP-based sequential rounding algorithm to address its computational intractability.
- We propose an efficient heuristic based on a static flow surrogate model to reduce the complexity further.
- We conduct numerical evaluations, demonstrate the trade-offs among various approaches, and highlight our proposed algorithms' advantages.

We present the system model in §2 and formulate the **DRS** problem in §3. We present an LP-based sequential rounding algorithm and reduce its complexity in §4 and §5, respectively. We present evaluation results in §6, introduce related work in §7, and conclude the paper in §8.

2. System Model

In this section, we first introduce the model of the network and transmission requests. We use an example to show the difficulties in presenting dynamic flow routing and service scheduling based on an original network graph. Then, we present a time-expended graph approach to model a dynamic routing strategy and its throughput.

A. Network Model

We consider a network represented by an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes (e.g., routers, edge

servers, mobile devices), and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of communication links. We assume that the network has $|\mathcal{V}| = n$ nodes and $|\mathcal{E}| = m$ links in total. Each link $e = (x, y) \in \mathcal{E}$ is associated with a fixed capacity, denoted by $c(x, y) > 0$, indicating the maximum amount of data that can be transmitted per time unit; and a travel delay, denoted by $\mu(x, y) \in \mathbb{Z}_{>0}$, which is the number of time units needed to traverse the link.

B. Transmission Request Scheduling and Dynamic Routing

The network accepts multiple data transmission requests from network users. We assume a global time horizon of T discrete time slots. Each time slot corresponds to the smallest interval for scheduling strategies. Each network request k is characterized by a commodity, which can be represented by a five-tuple $(s_k, t_k, \delta_k, r_k, d_k)$, $k \in \mathbb{K} = \{1, ..., K\}$, where:

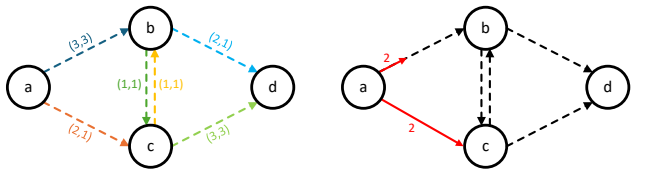
- $s_k, t_k \in \mathcal{V}$ are the request's source and destination nodes, respectively;
- $\delta_k \in \mathbb{Z}_{>0}$ is the completion deadline by which all the data must arrive at t_k, where $\delta_k \leq T$;
- $r_k \in \mathbb{R}_{>0}$ is the reward value (if completed on time);
- $d_k \in \mathbb{R}_{>0}$ is the demand, i.e, the amount of data to transmit.

We further define a feasible request subset $\mathcal{K} \subseteq \mathbb{K}$ such that we can find a dynamic routing strategy for all requests in $\mathcal{K}$ that can be transmitted to their destinations by their deadlines. Given a network topology, we use the following example to illustrate a feasible scheduling and dynamic routing strategy.

Example 1 (A feasible scheduling and dynamic routing strategy)**.** *Consider a network topology with three transmission requests, the first is from a to d, with a deadline of 5, a reward of 3, and a demand of 8. The second is from a to c, with a deadline of 4, a reward of 2, and a demand of 5. The third is from a to b, with a deadline of 3, a reward of 1, and a demand of 6. As shown in Fig. 1a, the first number on an edge is its capacity, and the second is its edge traversal time. Lastly, the total time horizon T is 5.*

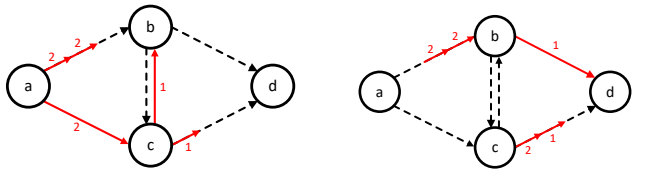
A feasible scheduling strategy is to schedule only the first commodity. We use the following time-sequential Fig. 1b-1f to illustrate a corresponding dynamic routing strategy. The flow values are illustrated in solid red arrows. Fig. 1b represents the time step $t = 1$. Specifically, 4 units leave node a at time 0, 2 units go to node b, the other 2 units go to node c; at $t = 1$, the 2 units going to node b have traversed $1/3$ of the link (a, b), and the other 2 units have arrived at node c. At time 1, the destination node d receives nothing. Fig. 1c shows that, at $t = 2$, all the flows are "on the road". Node d receives nothing as well. Fig. 1d shows that, at time 3, node d receives 1 unit, and the cumulative reward is 1. Fig. 1e shows that, at time 4, node d receives 3 unit, and the cumulative reward is 4. Fig. 1f shows that, at time 5, node d receives 4 unit, and the cumulative reward is 8.

The above scheduling and dynamic routing strategy is feasible, but can we achieve higher total rewards? To this



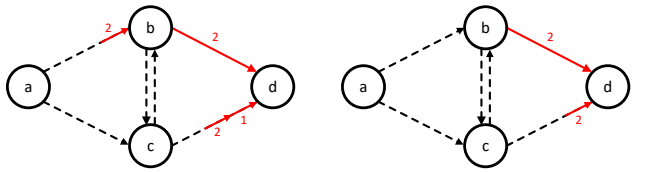
(a) Network topology. The tuple on each edge indicates its (capacity, delay). Note: the coloring is to aid the presentation of Fig. 2a and has no meaning otherwise.

(b) Flow values at $t = 1$. Node a sent 4 units of commodity total, 2 of which was sent along (a, b), and 2 units was sent via (a, c). Node c receives 2 units of commodity.



(c) Flow values at $t = 2$. 4 additional units of commodities have left node a, moving down (a, b) and (b, c). Another 2 units on (a, b) are still en route. Of 2 units received by c in the previous time step, 1 unit was sent along (c, b) and is received at b, while 1 unit was sent via (c, d).

(d) Flow values at $t = 3$. Node b receives 2 units, and sent 1 unit along (b, d), which is received at d. Node c sent 2 units via (c, d), and the 1 unit already sent is still en route.



(e) Flow values at $t = 4$. Node b receives 2 units and sends 2 units via (b, d). 2 units are en route (c, d). Node d receives 1 unit via (c, d), and 2 units via (b, d).

(f) Flow values at $t = 5$. Node b sent 2 units of flow via (b, d), which is received at d. Additionally, node d receives 2 units via (c, d), completing the network request.

Fig. 1: A step-by-step example showing Example 1 over time.

end, our network transmission request scheduling and dynamic routing objective is to achieve the following key properties:

- **Maximizing Total Rewards:** We aim to find a subset of commodities $\mathcal{K} \subseteq \mathbb{K}$ such that every commodity in $\mathcal{K}$ can be fulfilled, and the total reward is maximized.
- **Adhering to Completion Deadlines:** The scheduling strategy must ensure that the deadline of each commodity in $\mathcal{K}$ is not exceeded.
- **Satisfying Network Constraints:** The corresponding dynamic routing strategy must satisfy the network's physical constraints at all time.

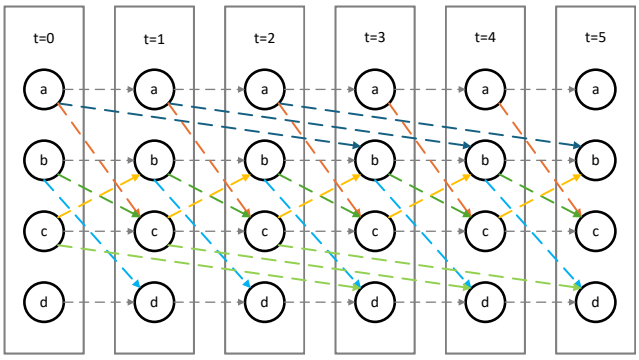
It is non-trivial to achieve the above objectives using the original graph $\mathcal{G}$. In the following, we introduce the time-expanded graph model. Then, given a $\mathcal{K}$, we formally define a routing strategy based on the time-expanded graph.

C. Dynamic Flow based on Time-expanded Graph

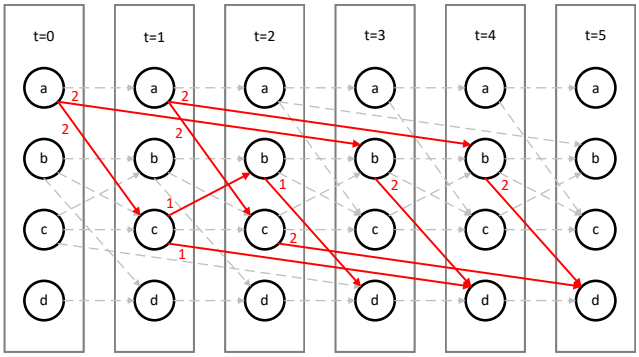
To model the temporal evolution of flows, we define the *time-expanded graph* of $\mathcal{G}$ as follows.

Definition 1 (Time-expanded graph)**.** *Given a time horizon T, we define the time-expanded graph $L\mathcal{G}(T)$, which is constructed from $\mathcal{G}$ as follows. For each node $x \in \mathcal{V}$, $L\mathcal{G}(T)$ has $T+1$ nodes $x(\tau)$, $\tau = 0, ..., T$, and the set of these nodes is denoted by $L\mathcal{V}(T)$; for each edge $(x, y) \in \mathcal{E}$, $L\mathcal{G}(T)$ has edges $(x(\tau), y(\tau + \mu(x, y)))$, $0 \le \tau \le T-1$, and the set of these edges is denoted by $L\mathcal{E}(T)$; in addition, there are edges $(x(\tau), x(\tau+1))$, $0 \le \tau \le T-1$, to represent holdovers at node x. A replica $(x(\tau), y(\tau+\mu(x, y)))$ of (x, y) has capacity $c(x, y)$, whereas holdover edges are assumed to have infinite capacity.*

Fig. 2a shows a time-expanded version of the network graph ($T = 5$) shown in Fig. 1a. Fig. 2b shows a flow in this network that corresponds to the dynamic routing shown in Fig. 1b-1f.



(a) Construction of time-expanded graph. Notice the color of edges corresponds to Fig. 1a. Light gray holdover edges are added between the same nodes over time.



(b) Corresponding feasible network request scheduling and routing strategy.

Fig. 2: Network flow of Example 1 in the time-expanded graph. Note that the flow is identical to Fig. 1.

Based on the definition of the time-expanded graph, each data flow is represented as a set of path segments with the amount of data transfer in $L\mathcal{G}(T)$, constrained by the deadline and bandwidth capacities. We formally define the *dynamic flow* model to represent a routing strategy:

Definition 2 (Dynamic flow)**.** *Given a set of network requests $\mathcal{K}$ and a time horizon T, we define a dynamic flow $\mathbf{W} = (W_1, W_2, \ldots, W_{|\mathcal{K}|})$, where for $k \in \mathcal{K}$, $W_k : L\mathcal{V}(T) \times L\mathcal{V}(T) \mapsto \mathbb{R}$, which satisfies the following constraints:*

1) Flow conservation:
$$\sum_{(v(\tau), y(\tau+\mu(v,y))) \in L\mathcal{E}(T)} W_k(v(\tau), y(\tau+\mu(v,y))) - \sum_{(x(\tau-\mu(x,v)), v(\tau)) \in L\mathcal{E}(T)} W_k(x(\tau-\mu(x,v)), v(\tau)) = 0, v \notin \{s_k, t_k\}; \quad \text{(a)}$$

2) Non-negativity:
$$W_k(x(\tau), y(\tau+\mu(x,y))) \ge 0, \quad \forall (x(\tau), y(\tau+\delta(x,y))) \in L\mathcal{E}(T); \quad \text{(b)}$$

3) Capacity:
$$\sum_{k \in \mathcal{K}} W_k(x(\tau), y(\tau+\mu(x,y))) \le c(x,y), \quad \forall (x(\tau), y(\tau+\mu(x,y))) \in L\mathcal{E}(T). \quad \text{(c)}$$

Moreover, given a network request $k \in \mathcal{K}$ with completion deadline $\delta_k \le T$, we have the following defined *deadline-constrained time-expanded subgraph*:

Definition 3 (Deadline-constrained subgraph)**.** *For a network request $k \in \mathcal{K}$ with completion deadline $\delta_k \le T$, we define a subgraph $L\mathcal{G}(\delta_k) = (L\mathcal{V}(\delta_k), L\mathcal{E}(\delta_k))$ as a* deadline-constrained time-expanded subgraph *of $L\mathcal{G}(T)$, which includes only the temporal nodes and edges that lie within the time window $[0, \delta_k]$. Here:*

- *$L\mathcal{V}(\delta_k) = \{v(\tau) \mid v \in \mathcal{V},\ \tau \in \{0, 1, \ldots, \delta_k\}\}$ is the node set which includes all temporal copies of nodes from the original graph up to time δ_k.*
- *$L\mathcal{E}(\delta_k) = \{(x(\tau), y(\tau+\mu(x,y))) \mid \tau + \mu(x,y) \le \delta_k\}$, where $(x, y) \in \mathcal{E}$, is the edge set which includes edges whose arrival time does not exceed δ_k.*

Based on Definition 3, we further define the throughput of a network request as follows:

Definition 4 (Network request throughput)**.** *For a network request $k \in \mathbb{K}$, its throughput is defined as the flow leaving entering destination t_k before the deadline δ_k:*

$$\Delta_k = \sum_{(x(\tau-\mu(x,v)), t_k(\tau)) \in L\mathcal{E}(\delta_k)} W_k(x(\tau-\mu(x,v)), t_k(\tau)) - \sum_{(t_k(\tau), y(\tau+\mu(v,y))) \in L\mathcal{E}(\delta_k)} W_k(t_k(\tau), y(\tau+\mu(v,y))).$$

3. Problem Formulation

Recall that we aim to select a feasible subset of transmission requests and compute a corresponding dynamic flow over the time-expanded graph to maximize the total reward. Formally, we formulate the dynamic flow routing and scheduling (**DRS**) as the following optimization problem:

$$\textbf{DRS}: \max_{\mathcal{K}, \mathbf{W}} \sum_{k \in \mathcal{K}} r_k \quad (1)$$

$$\text{s. t. } \Delta_k = d_k, \forall k \in \mathcal{K}; \quad (2)$$

$$\text{(a), (b)}, \forall k \in \mathcal{K}, \text{(c)}. \quad (3)$$

Here, constraint (2) ensures that for each network request $k \in \mathcal{K}$, the total flow received at t_k by deadline δ_k satisfies its demand d_k. Constraint (a) ensures the flow conservation. Constraint (b) ensures that Flow values are non-negative.

Constraint (c) ensures that the flow across all commodities cannot exceed physical link capacities at any time. This mixed-integer program is challenging to solve directly due to the combinatorial nature of selecting $\mathcal{K}$ and satisfying the underlying flow constraints over time. In the following sections, we introduce an LP-based sequential rounding approach to overcome **DRS**'s computational intractability and reduce the time complexity further in a static flow perspective.

4. LP-based Sequential Rounding Approach

A. LP-based Feasibility Checking

A key subroutine required in our proposed rounding scheme is a feasibility-checking step that determines whether a given subset of requests can be served before their deadlines without violating link capacity constraints. We formulate this subroutine as an LP feasibility problem over the time-expanded graph. Specifically, given a candidate network request set $\mathcal{K} \subseteq \mathbb{K}$, we aim to determine whether there exists a dynamic flow routing strategy that satisfies the deadline, flow conservation, and bandwidth constraints for all $k \in \mathcal{K}$. This can be formulated as the following problem over the deadline-constrained time expanded subgraph $L\mathcal{G}(\delta_k)$ for each $k \in \mathcal{K}$:

$$\textbf{P1}(\mathcal{K})\textbf{:}\ \texttt{Find}\ W_k\ \texttt{for each}\ k \in \mathcal{K} \tag{4}$$
$$\text{s. t. } \Delta_k = d_k, \forall k \in \mathcal{K}; \tag{5}$$
$$\text{(a), (b)}, \forall k \in \mathcal{K}, \text{(c)}. \tag{6}$$

This LP-feasibility problem ensures that each flow obeys flow conservation at intermediate nodes, satisfies its demand before its deadline, respects the link capacity constraints at each time slot, and maintains non-negativity of all flows. The feasibility of **P1**($\mathcal{K}$) determines whether a given subset is admissible. Based on the above, we introduce a relaxation-based heuristic and later a method to reduce complexity via static flows.

B. LP Relaxation for Reward Maximization

While solving **P1**($\mathcal{K}$) helps validate whether a given subset of requests can be supported, it does not directly guide the selection of high-reward request subsets. To this end, we propose an LP relaxation that enables global reasoning about request priorities based on their reward values and feasibility potential. Specifically, we define **P2**($\mathbb{K}$) of the original **DRS** problem by relaxing the equality constraint $\Delta_k = d_k$, and maximizing the weighted sum of rewards where the each weight is the fractional reward ratio $\frac{\Delta_k}{d_k}$:

$$\textbf{P2}(\mathbb{K})\textbf{:}\ \max_{\mathbf{W}} \sum_{k\in\mathbb{K}} \frac{\Delta_k}{d_k} r_k \tag{7}$$
$$\text{s. t. } \Delta_k \le d_k, \forall k \in \mathbb{K}; \tag{8}$$
$$\text{(a), (b)}, \forall k \in \mathbb{K}; \tag{9}$$
$$\sum_{k\in\mathbb{K}} W_k(x(\tau), y(\tau + \mu(x,y))) \le c(x,y), \quad \forall (x(\tau), y(\tau+\mu(x,y))) \in L\mathcal{E}(T). \tag{10}$$

Solving **P2**($\mathbb{K}$) yields a fractional solution that helps prioritize which requests to consider in a feasible final subset. The next step is to apply a sequential rounding approach using this global view.

C. LP-based Sequential Rounding Algorithm

Our proposed LP-based sequential rounding (LPSR) algorithm constructs a feasible request subset by initializing the solution set $\mathcal{K}$ with *fully satisfied requests* in the solution of **P2**($\mathbb{K}$) (i.e., commodities where the reward ratio $\frac{\Delta_k}{d_k} = 1$, for $k \in \mathbb{K}$) and then greedily augmenting it based on the descending order of fractional reward ratios. Each new candidate is admitted only if it passes the feasibility check defined by **P1**($\mathcal{K}$). The proposed LPSR algorithm is outlined in Alg. 1.

Algorithm 1: LPSR$(T, \mathcal{G}, \mathcal{K})$

Input: T: time horizon; $\mathcal{G}$: graph; $\mathbb{K}$: request set
Output: $\mathbf{W}$: dynamic flow; $\mathcal{K}$: feasible request set
Construct $L\mathcal{G}(T)$;
Solve **P2**($\mathbb{K}$);
$\mathcal{K} \leftarrow \left\{k | \frac{\Delta_k}{d_k} = 1, k \in \mathbb{K}\right\}$;
$\hat{\mathcal{K}} \leftarrow \mathbb{K} \setminus \mathcal{K}$;
for $k \in \hat{\mathcal{K}}$ *in descending order of* $\frac{\Delta_k}{d_k}$ **do**
  if ***P1***$(\mathcal{K} \cup k)$ *is feasible* **then**
    $\mathcal{K} \leftarrow \mathcal{K} \cup k$;
Output $\mathbf{W}$, $\mathcal{K}$

The LP-based sequential rounding algorithm provides a principled trade-off between partial reward maximization and strict feasibility. Nevertheless, LPSR requires repeatedly solving **P1** over a growing subset, which incurs high computational overhead due to the size of the time-expanded graph. In the next section, we present a technique to replace this with a much more efficient static-flow-based feasibility check.

5. Reducing Complexity: Static Flow Perspective

This section proposes a computationally efficient surrogate based on static flows over the original graph.

A. Static Flow LP for Feasibility Checking

We first define a static flow and its throughput and describe how data is routed from source to destination under steady-state assumptions.

Definition 5 (Static flow)**.** *Given a network requests set $\mathcal{K}$, we define a static flow by $\mathbf{f} = (f_1, f_2, \ldots, f_{|\mathcal{K}|})$, where for $k \in \mathcal{K}$, $f_k : \mathcal{V} \times \mathcal{V} \mapsto \mathbb{R}$, which satisfies the following constraints:*

1) Flow conservation: $\sum_{(v,y)\in\mathcal{E}} f_k(v,y) - \sum_{(x,v)\in\mathcal{E}} f_k(x,v) = 0, \quad v \notin \{s_k, t_k\};$ (d)

2) Non-negativity: $f_k(x,y) \ge 0\ \forall (x,y) \in \mathcal{E};$ (e)

3) Capacity: $\sum_{k\in\mathcal{K}} f_k(x,y) \le c(x,y),\ \forall (x,y) \in \mathcal{E}.$ (f)

Definition 6 (Static flow throughput)**.** *The static throughput of request $k \in \mathbb{K}$ under a static flow f_k is defined as the net amount of flow reaching the destination node:*

$$V_k = \sum_{(x,t_k)\in\mathcal{E}} f_k(x,t_k) - \sum_{(t_k,y)\in\mathcal{E}} f_k(t_k,y). \tag{11}$$

This definition captures the net flow delivery from s_k to t_k, analogous to its dynamic version Δ_k in the time-expanded graph. However, this static model does not directly incorporate deadlines or link traversal delays. To address this, we leverage a key result originally introduced by Ford and Fulkerson [2], which relates the dynamic throughput of a request to its corresponding static flow.

Lemma 1 (Ford and Fulkerson [2])**.** *Let f_k be a feasible static flow for request $k \in \mathbb{K}$, with static throughput V_k, and let $\mu(u,v)$ denote the traversal delay of link $(u,v) \in \mathcal{E}$. Then, the dynamic throughput Δ_k achieved under the deadline of δ_k is given by:*

$$\Delta_k = (\delta_k + 1)V_k - \sum_{(u,v)\in\mathcal{E}} \mu(u,v) f_k(u,v). \tag{12}$$

Remark 1. *The term $(\delta_k + 1)V_k$ in (12) represents the maximum cumulative flow that could be delivered if every unit of flow travels instantly within a time horizon of $\delta_k + 1$. The second term deducts the "in-transit" delay penalty caused by link traversal times. Intuitively, the dynamic flow is a time-weighted transformation of the static flow. This relation enables us to check the feasibility of serving a request k over the original graph, by ensuring that $\Delta_k = d_k$.*

Based on Lemma 1, we formulate a static LP feasibility problem to determine whether a candidate request subset can be satisfied under bandwidth and deadline constraints using only the *original graph*:

$$\textbf{P3}(\mathcal{K})\textbf{:}\ \texttt{Find}\ f_k\ \texttt{for each}\ k \in \mathcal{K} \tag{13}$$

$$\text{s. t. } (\delta_k + 1)V_k - \sum_{(u,v)\in\mathcal{E}} \mu(u,v) f_k(u,v) = d_k, \forall k \in \mathcal{K}; \tag{14}$$

$$\text{(d), (e)}, \forall k \in \mathcal{K}, \text{(f)}. \tag{15}$$

Remark 2. ***P3***$(\mathcal{K})$ *serves as a fast and delay-aware surrogate for the more expensive* ***P1***$(\mathcal{K})$*, operating entirely on the original graph.* ***P3***$(\mathcal{K})$ *is more conservative than* ***P1***$(\mathcal{K})$ *because it checks feasibility using static flows that must satisfy all demands within a fixed deadline using average delay penalties, without accounting for how flows can be scheduled over time to share link capacity more efficiently. While slightly conservative, solving* ***P3***$(\mathcal{K})$ *significantly reduces the computational burden when evaluating large subsets of requests.*

B. Efficient Rounding with Static Feasibility Checking

We now incorporate the static LP feasibility problem **P3**$(\mathcal{K})$ into the proposed LP-based sequential rounding scheme to reduce its computational complexity. The key idea is to first apply the cheaper static feasibility checking (solving **P3**$(\mathcal{K})$) to screen new request candidates. Only if **P3**$(\mathcal{K})$ fails do we invoke the full dynamic check using **P1**$(\mathcal{K})$, which operates on the time-expanded graph. This results in a hybrid rounding algorithm, referred to as Efficient LP-based Sequential Rounding (`E-LPSR`), which retains the reward prioritization of the LP relaxation while avoiding expensive graph expansions. We summarize `E-LPSR` as Alg. 2.

Algorithm 2: `E-LPSR`$(T, \mathcal{G}, \mathcal{K})$

Input: T: time horizon; $\mathcal{G}$: graph; $\mathbb{K}$: request set
Output: $\mathbf{W}$: dynamic flow; $\mathcal{K}$: feasible request set
Construct $L\mathcal{G}(T)$;
Solve **P2**$(\mathbb{K})$;
$\mathcal{K} \leftarrow \left\{k | \frac{\Delta_k}{d_k} = 1, k \in \mathbb{K}\right\}$;
$\hat{\mathcal{K}} \leftarrow \mathbb{K} \setminus \mathcal{K}$;
for $k \in \hat{\mathcal{K}}$ *in descending order of* $\frac{\Delta_k}{d_k}$ **do**
  if ***P3***$(\mathcal{K} \cup k)$ *is feasible* **then**
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{k\}$;
  else if ***P1***$(\mathcal{K} \cup k)$ *is feasible* **then**
    $\mathcal{K} \leftarrow \mathcal{K} \cup \{k\}$;
return $\mathbf{W}$, $\mathcal{K}$

6. Performance Evaluation

A. Evaluation Setup

We evaluate the performance of our proposed Efficient LP-based Sequential Rounding (`E-LPSR`) heuristic against four baselines: the regular LP-based Sequential Rounding (`LPSR`), two greedy heuristics, Earliest Deadline First (`Greedy-EDF`) and Largest Reward First (`Greedy-LRF`), and the exact exhaustive-search solution. All experiments are conducted on a synthetic 8×8 grid network, where each of the 64 nodes is connected to its four orthogonal neighbors by unit-capacity, unit-delay links. We generate 8 commodities, each originating at the upper-left node $(1,1)$ and destined for the lower-right node $(8,8)$. To introduce heterogeneity, each commodity is assigned a distinct demand, deadline, and reward. The global time horizon is set to $T = 64$. For each algorithm, we measure:

- *Total Reward*: Sum of rewards of all commodities delivered by their deadlines;
- *Computation Time*: End-to-end runtime of the scheduling and routing algorithm (in milliseconds).

B. Evaluation Results

Table I summarizes the total reward achieved and the computation time required by each algorithm.

From Table I, we observe:

- **Reward Quality.** Both LP-based rounding schemes (`E-LPSR` and `LPSR`) achieve a near-optimal total reward of 23, i.e., 98.5% of the exact solution's reward of 24. In contrast, two greedy heuristics suffer a significant drop in their rewards (15 and 16), indicating their inability to balance deadlines and rewards effectively.

TABLE I: Comparison of Algorithms on the 8×8 Grid ($T = 64$, 8 commodities)

Algorithm	Reward	Time (ms)
E-LPSR	23	1,287
LPSR	23	3,909
Greedy-EDF	15	3,264
Greedy-LRF	16	2,711
Exact (Exhaustive)	24	15,634

- **Computational Efficiency.** E-LPSR reduces runtime by approximately 67% compared to LPSR ($1,287$ ms vs. $3,909$ ms), demonstrating the efficiency of our static-flow approach. Although Greedy-LRF and Greedy-EDF are faster than LPSR, they still incur higher runtime than E-LPSR while delivering a lower reward. The exhaustive search achieves the highest reward but costs prohibitive computation time (15.6 s), rendering it impractical for larger networks or real-time applications.

These results indicate that E-LPSR strikes the best trade-off between solution quality and efficiency, making it well suited for time-critical network services in practical dynamic and delay-sensitive environments.

7. Related Work

A. Dynamic Flow Scheduling and Routing

The study of dynamic flow scheduling and routing originates from classical flow network theory, with the time-expanded graph model introduced in the foundational work of Ford and Fulkerson [2]. However, most existing subsequent works focus on maximizing throughput or minimizing latency without incorporating application-level utility metrics such as task reward or deadline satisfaction. Moreover, solving dynamic flow problems over time-expanded graphs can be computationally intensive, prompting recent work to explore efficient heuristics and LP relaxations [5, 6]. Our work extends this line of research by integrating reward maximization with deadline-aware feasibility checks and proposing efficient approximations using static flow surrogates.

B. Deadline-aware Networking

Prior works have studied deadline-constrained task scheduling and routing in wireless networks [1, 3], typically assuming fixed paths or single-hop communication. These approaches often simplify the network dynamics by omitting traversal delays, which limits their applicability in tactical edge environments where both link delay and bandwidth heterogeneity are prevalent. In contrast, our work captures the full temporal dynamics of network flows, explicitly accounts for path-dependent delays and bandwidth constraints, and provides scalable algorithms for selecting the most valuable subset of requests under strict deadline constraints.

8. Conclusion

This paper addresses the problem of dynamic flow routing and scheduling for time-critical network services in next-generation networks. We formulate the problem as a mixed-integer program over a time-expanded graph, aiming to maximize the total reward of completed services under demand, deadline, and bandwidth constraints. To overcome the computational intractability of exact solutions, we propose an LP-based sequential rounding algorithm guided by the solution of a relaxed reward maximization problem. To further reduce complexity, we introduce a delay-aware static flow surrogate model that enables fast feasibility checks using only the original network topology, and propose an efficient heuristic. The simulation results demonstrate the trade-off among different approaches and validate the effectiveness of the proposed algorithms. Future work will explore online algorithms and robustness under dynamic topologies and uncertain network conditions.

9. Acknowledgementys

The research was sponsored by DEVCOM US Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-23-2-0225. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies either expressed or implied, of DEVCOM Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] S. ElAzzouni, E. Ekici, and N. Shroff, "Is deadline oblivious scheduling efficient for controlling real-time traffic in cellular downlink systems?" in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 49–58.

[2] D. R. Ford and D. R. Fulkerson, *Flows in Networks*. USA: Princeton University Press, 2010.

[3] M. Jain, A. Mishra, A. Wiese, S. Das, A. Bhattacharya, and M. Maity, "A deadline-aware scheduler for smart factory using wifi 6," in *Proceedings of the Twenty-fifth International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2024, pp. 221–230.

[4] S. Lin, Z. Zhou, Z. Zhang, X. Chen, and J. Zhang, *Edge intelligence in the making: Optimization, deep learning, and applications*. Springer, 2021.

[5] C. Tsanikidis and J. Ghaderi, "Online scheduling and routing with end-to-end deadline constraints in multihop wireless networks," in *Proceedings of the Twenty-Third International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*, 2022, pp. 11–20.

[6] C. Tsanikidis and J. Ghaderi, "Scheduling stochastic traffic with end-to-end deadlines in multi-hop wireless networks," in *IEEE INFOCOM 2024-IEEE Conference on Computer Communications*. IEEE, 2024, pp. 651–660.

[7] Z. Zhang, S. Lin, M. Dedeoglu, K. Ding, and J. Zhang, "Data-driven distributionally robust optimization for edge intelligence," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 2619–2628.

[8] Z. Zhang, X. Lin, G. Xue, Y. Zhang, and K. S. Chan, "Joint optimization of task offloading and resource allocation in tactical edge networks," in *MILCOM 2024-2024 IEEE Military Communications Conference (MILCOM)*. IEEE, 2024, pp. 703–708.