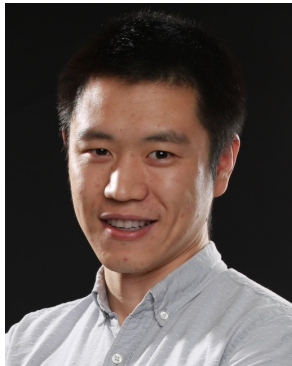


Congestion-aware Distributed Task Offloading in Wireless Multi-hop Networks using Graph Neural Networks



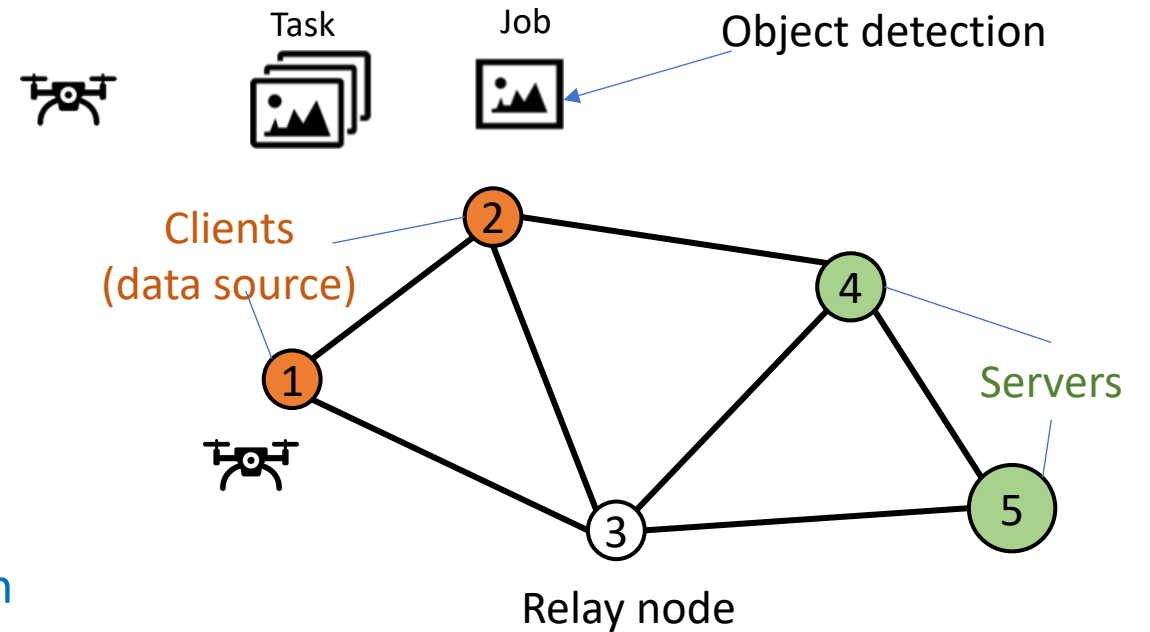
Zhongyuan Zhao*, ***Jake Perazzone†***, ***Gunjan Verma†***, ***Santiago Segarra****

*Rice University, USA

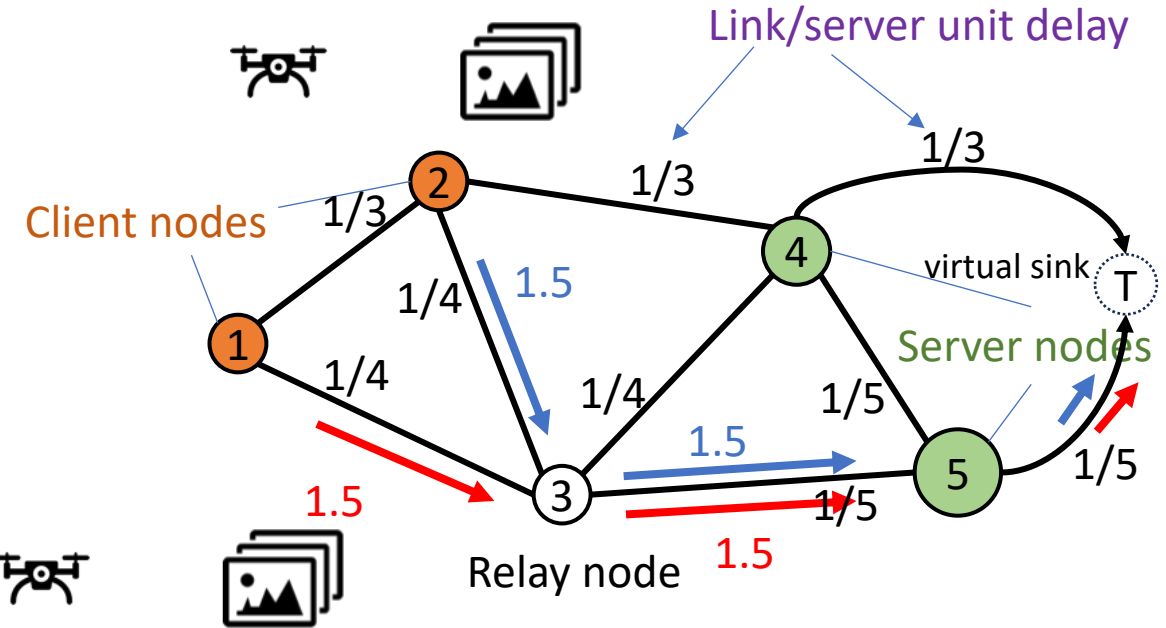
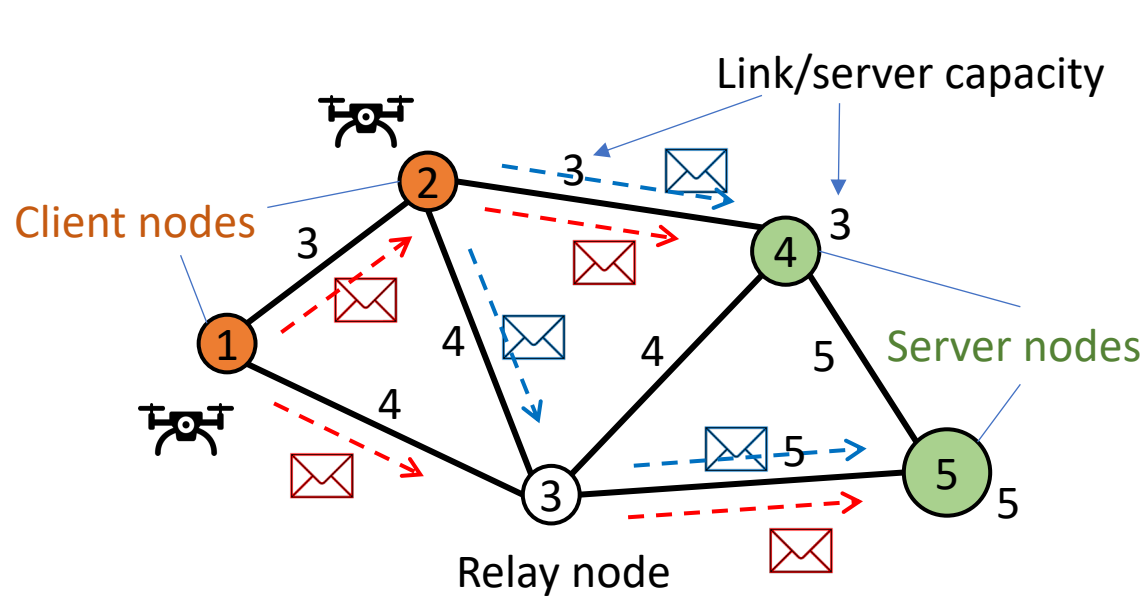
† US Army's DEVCOM Army Research Laboratory, USA

Distributed task offloading

- Wireless multihop networks
 - Clients: data source, resource-constrained
 - Relay: well-connected, no computing
 - Servers
- Task: a steady flow of similar jobs
 - Same job type (object detection)
 - Same job data size (a video frame)
- Client decision-making
 - Location: which server to do computing
 - Routing: path to the selected server
- Multiple clients make parallel decisions in a batch
 - Streaming based on per-task decisions
 - Minimize average job **response time**



Baseline: context-agnostic distributed offloading



Step 1: clients send out **probing messages**

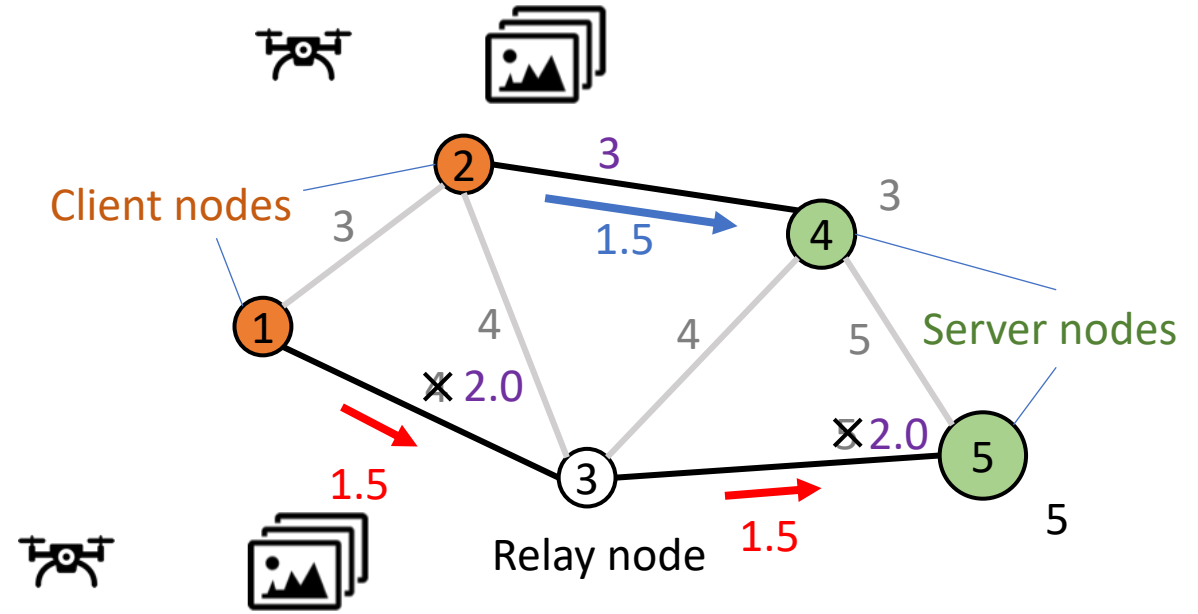
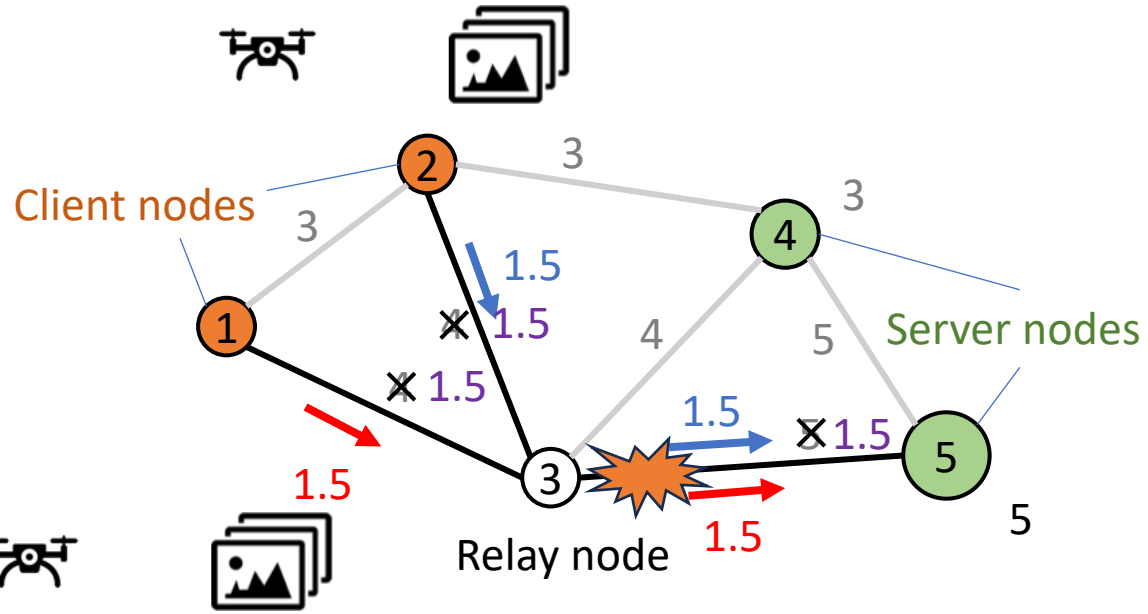
Step 2: clients send **task flows** via “the shortest path” to the virtual sink

Minimize task response time

$$\delta = \frac{1}{r}$$

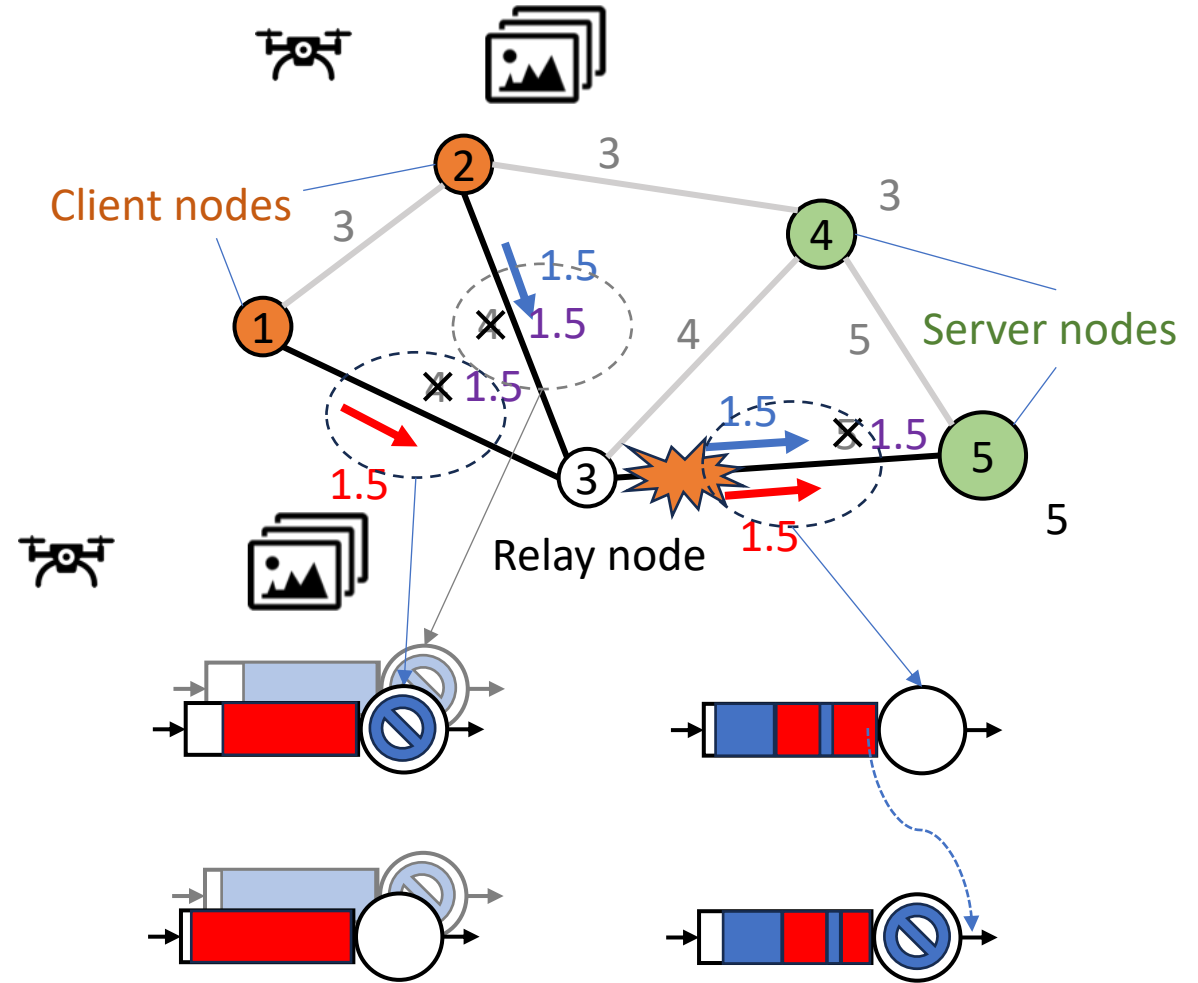
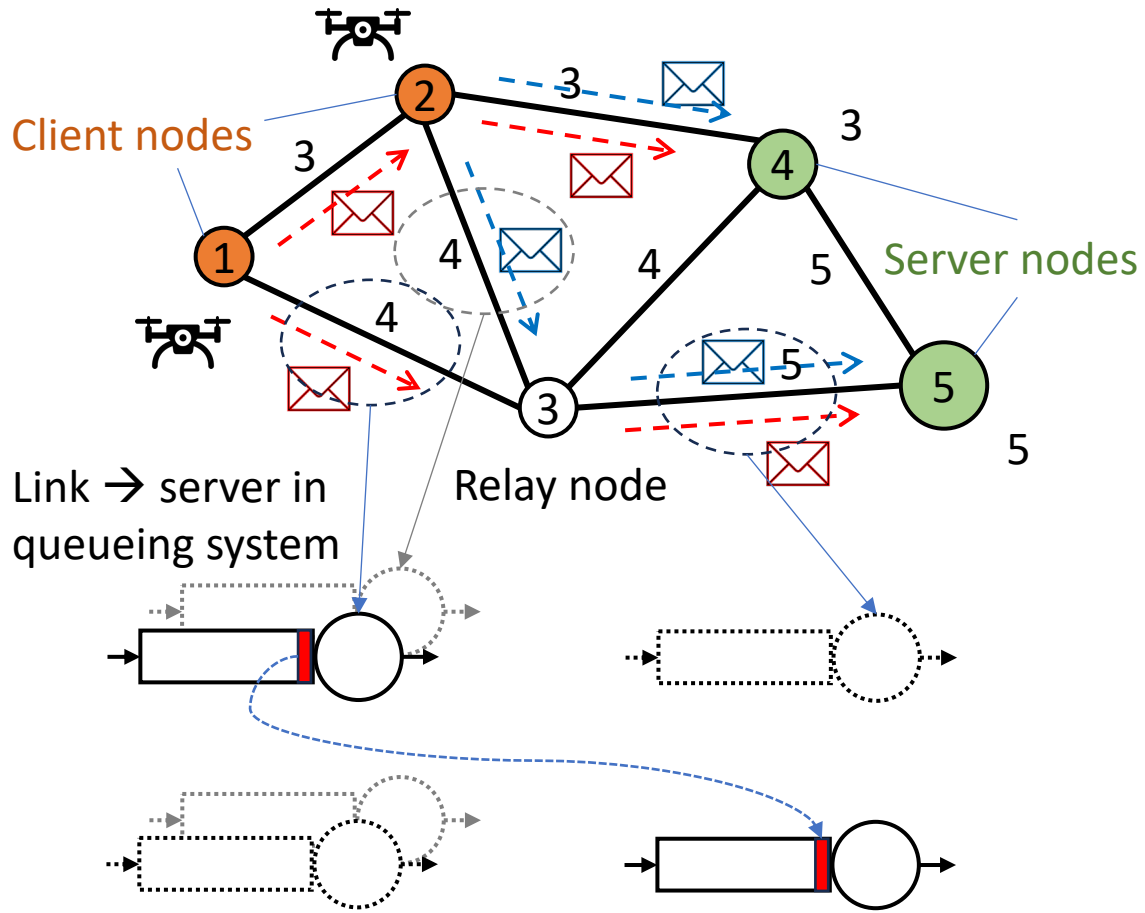
Consider **per-link unit delay** as edge weight

What could go wrong in wireless networks?



Link capacity changes once the streaming begins, depending on path selection & flow rate assignment

Queueing networks with interference constraints

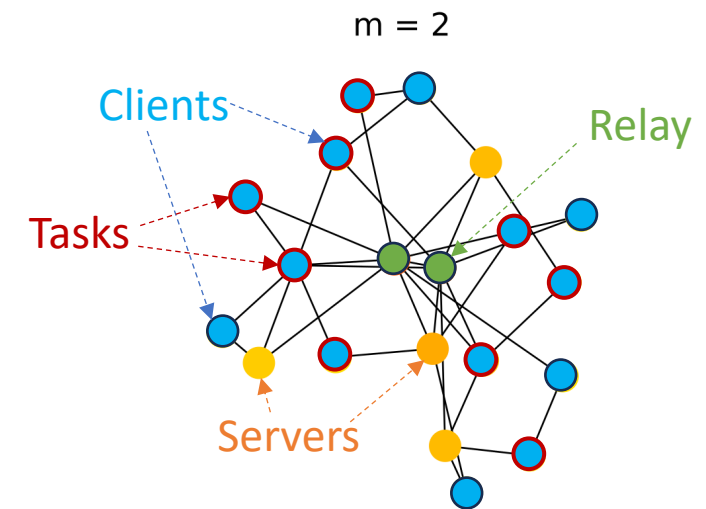


probing messages are **short-lived** flows

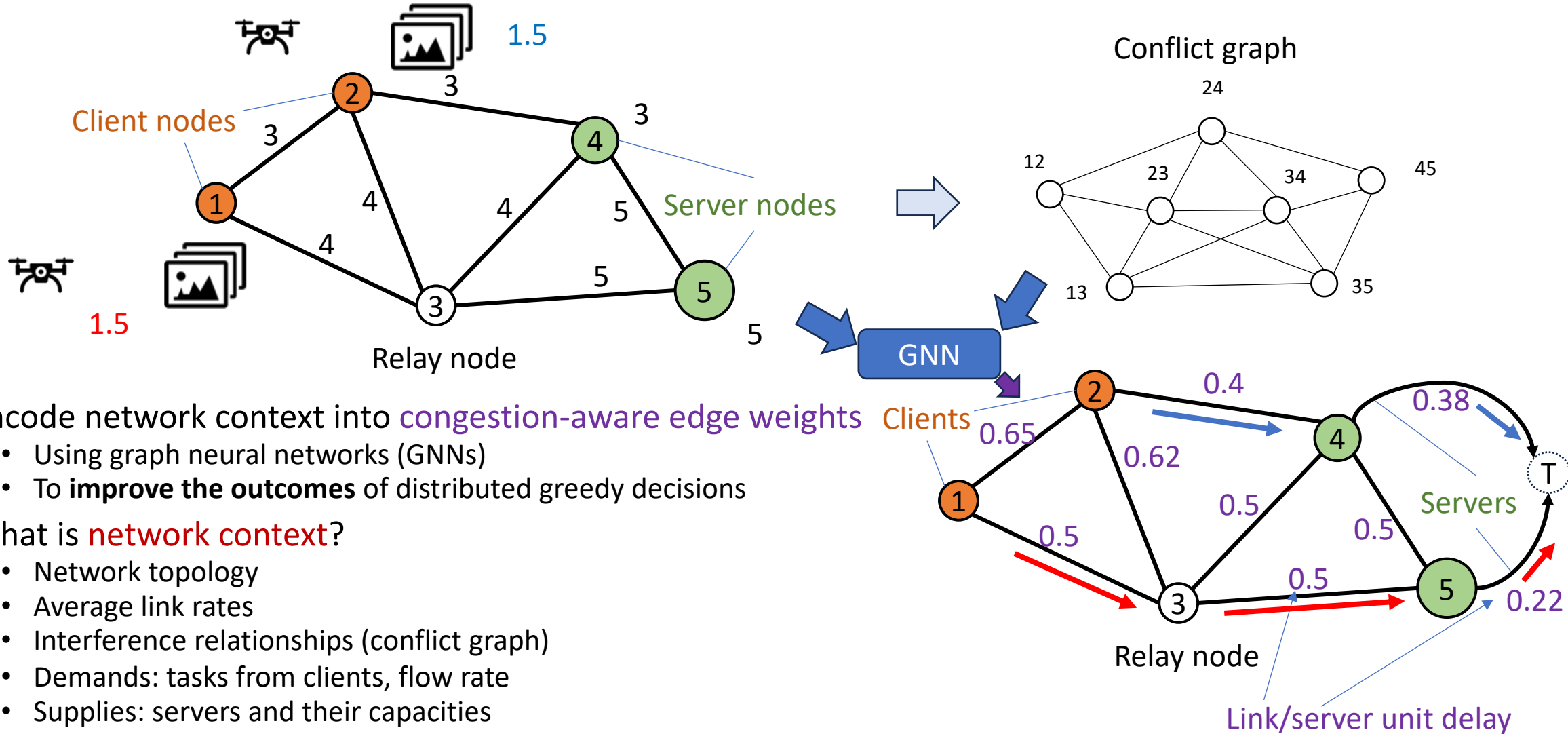
Links (1,3), (2,3), (3,5) **conflict** with each other, since node 3 has only one radio interface

Alternative decision frameworks

- Distributed greedy decision
 - Shortest path
 - Low communication overhead
 - Congestion/collision
- Centralized scheduler
 - High communication overhead
 - Single point of failure
- Peer coordination between clients
 - Difficult for large networks
 - High communication overhead

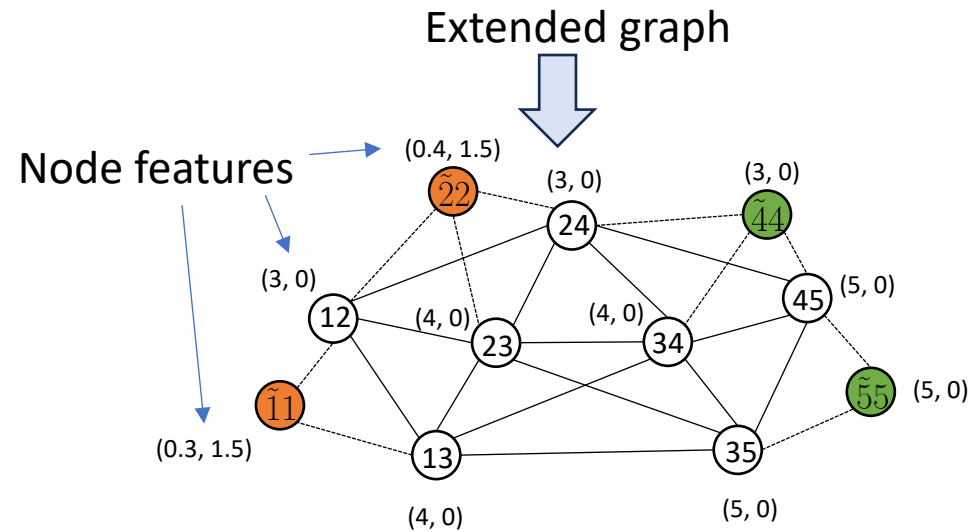
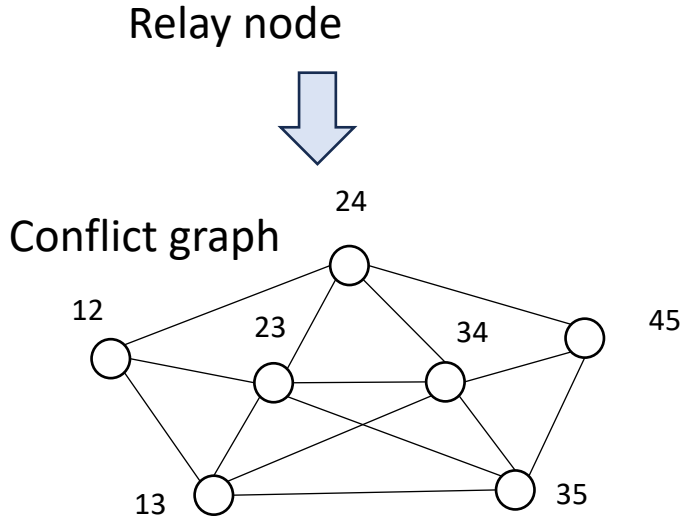
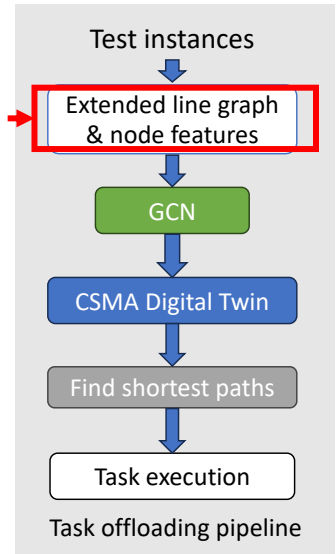
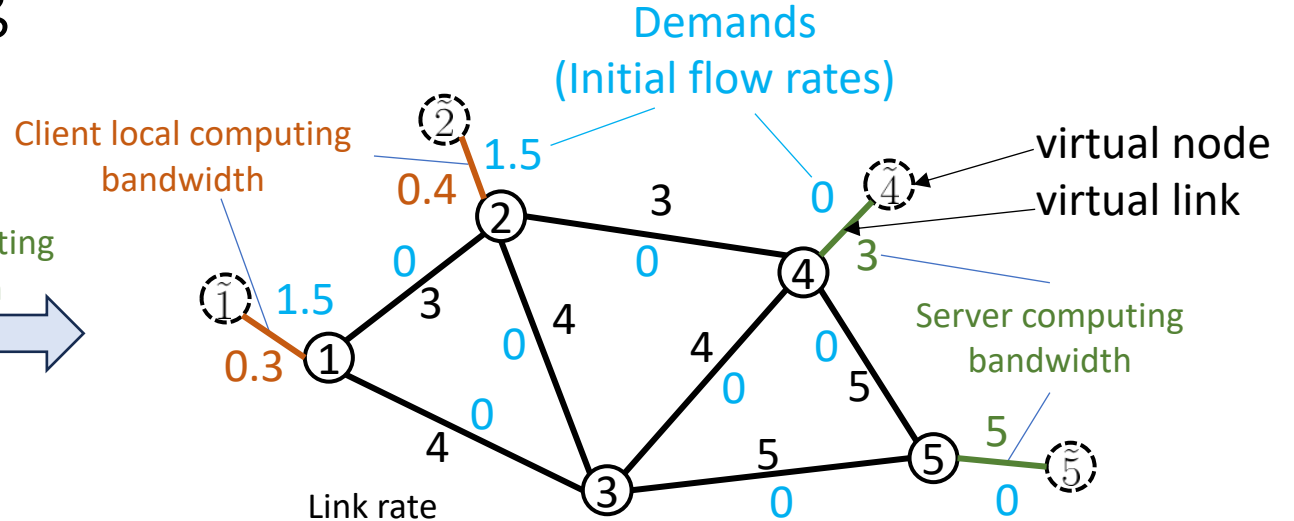
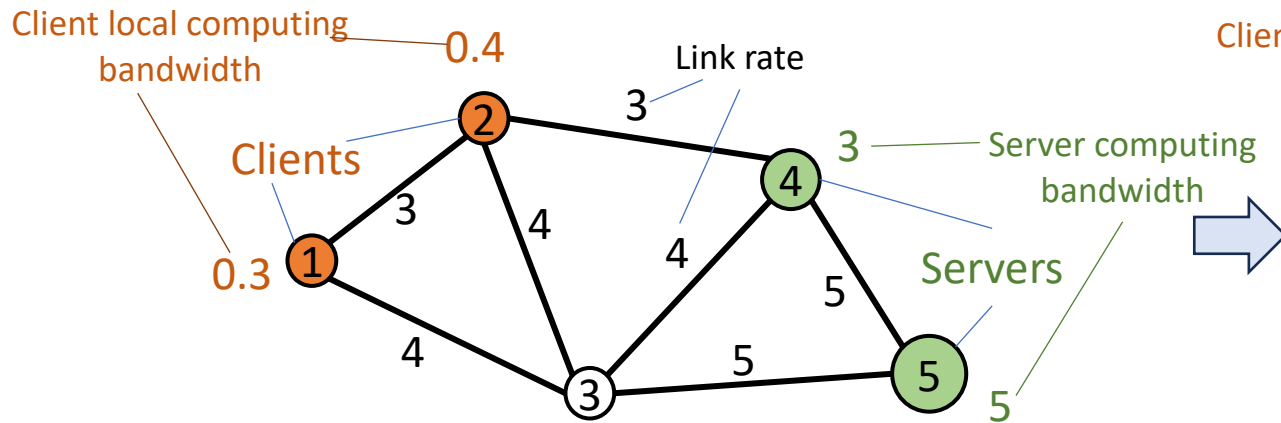


Our solution: keep shortest path decision, change *edge weights*



- Encode network context into *congestion-aware edge weights*
 - Using graph neural networks (GNNs)
 - To **improve the outcomes** of distributed greedy decisions
- What is **network context**?
 - Network topology
 - Average link rates
 - Interference relationships (conflict graph)
 - Demands: tasks from clients, flow rate
 - Supplies: servers and their capacities

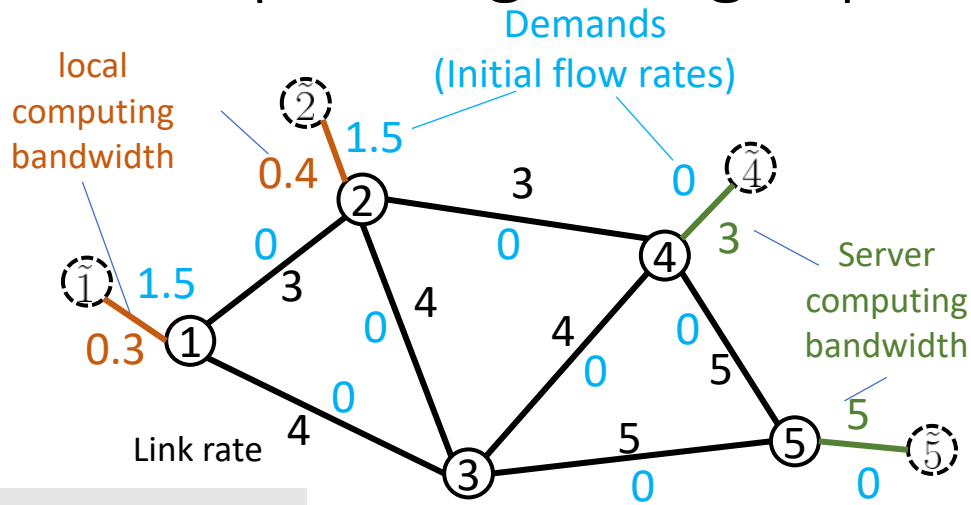
Our solutions: graph modeling



So that they can be processed by our GNN

Extended line graph

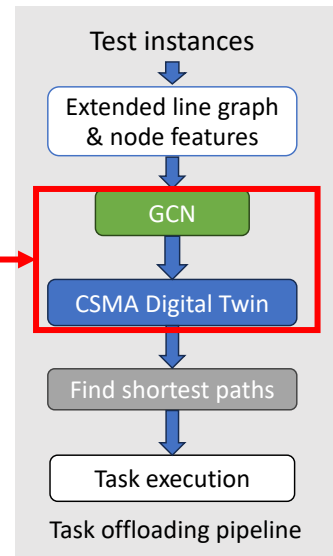
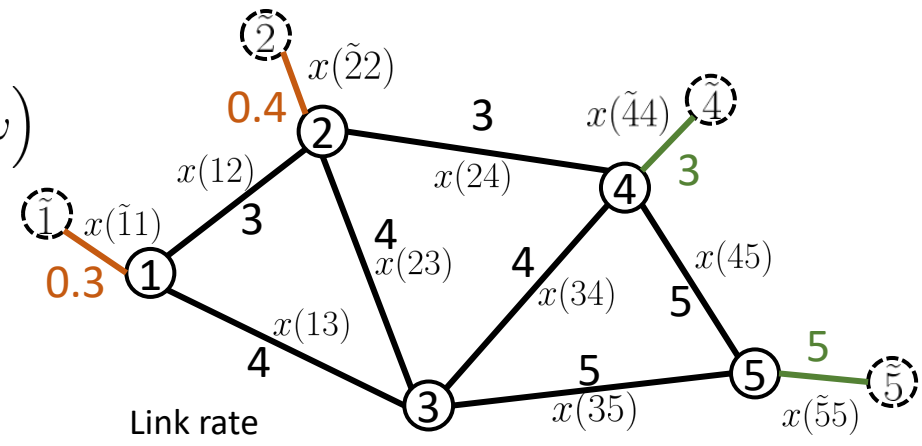
Step 1: edge weight prediction by GNN



$$\mathbf{x}^l = \Psi_{\mathcal{G}^l}([\mathbf{q}^l, \mathbf{w}^l, \lambda^l, \mathbf{r}^l]; \omega)$$



1.1 Traffic load prediction

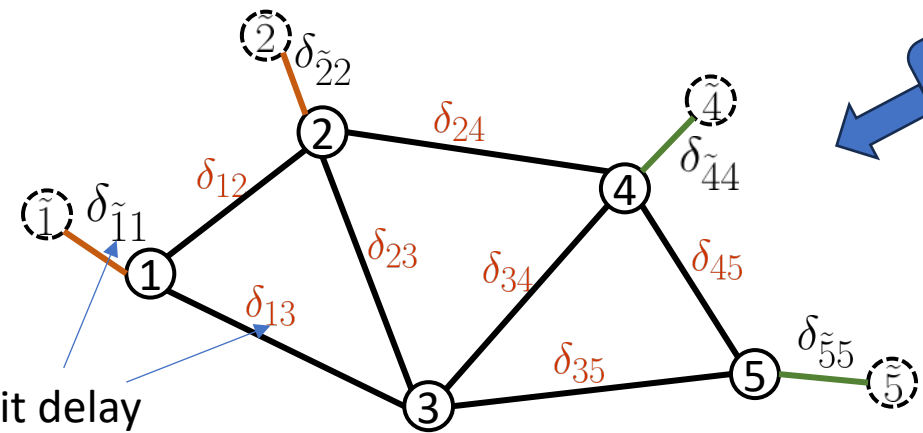


Extended graph

1.2 Link unit delay prediction



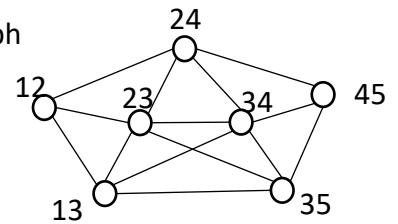
$$\hat{\delta}^l = \varphi(\mathcal{G}^c, \mathcal{G}^l, \mathbf{r}^l, \mathbf{x}^l, T, K)$$



Link unit delay

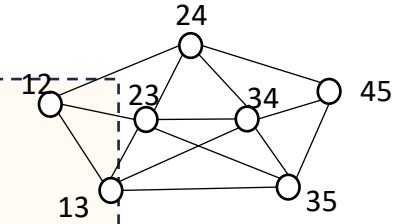
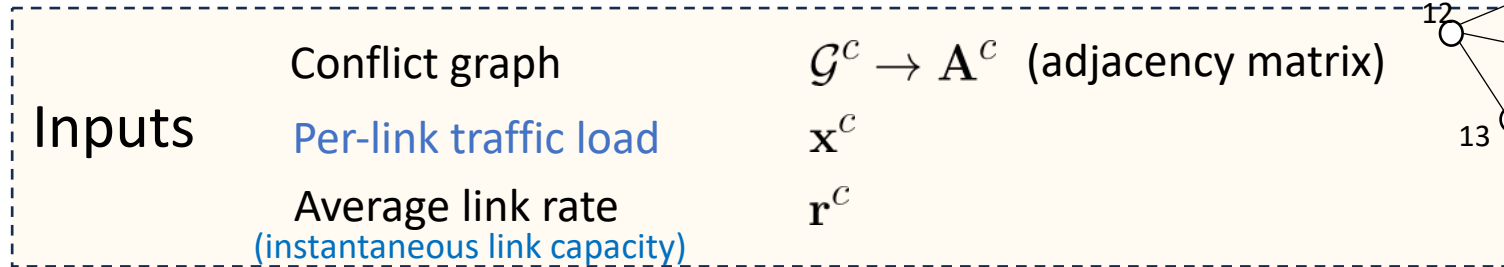
Extended graph

Conflict graph



CSMA Digital Twin

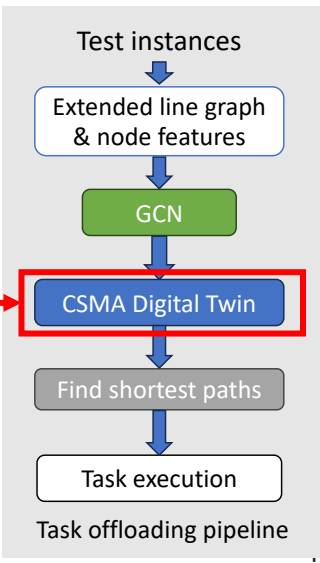
A **differentiable** graph process that predicts **link unit delay** under CSMA scheduling



Initially assume all links are busy

$$\mathbf{b}^c(0) = \mathbf{1}^c$$

CSMA: contending neighbors have **equal chances** of channel access



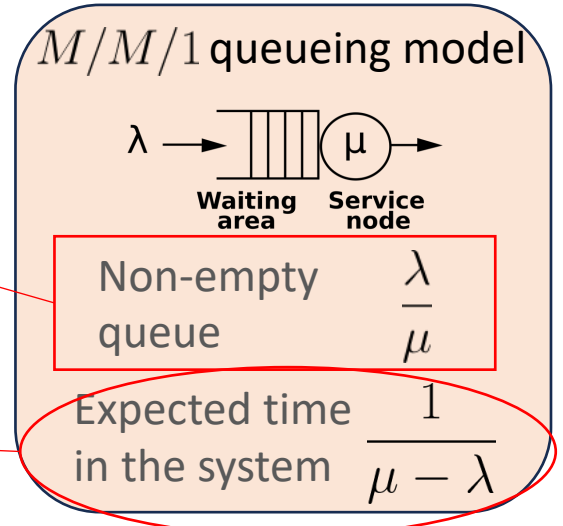
long-term link service rate

Expected number of contending neighbors

Contention probability

```

for  $k \in \{0, \dots, K\}$  do
     $\mu^c(k) = \frac{\mathbf{r}^c}{\mathbf{1} - \mathbf{A}^c \mathbf{b}^c(k)}$ 
     $\mathbf{b}^c(k+1) = \min \left\{ \frac{\mathbf{x}^c}{\mu^c(k)}, \mathbf{1}^c \right\}$ 
end for
    
```



Output Unit delay on physical links

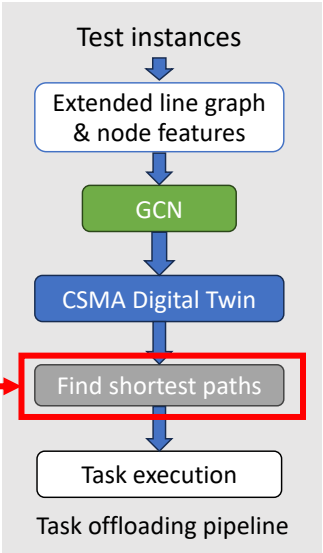
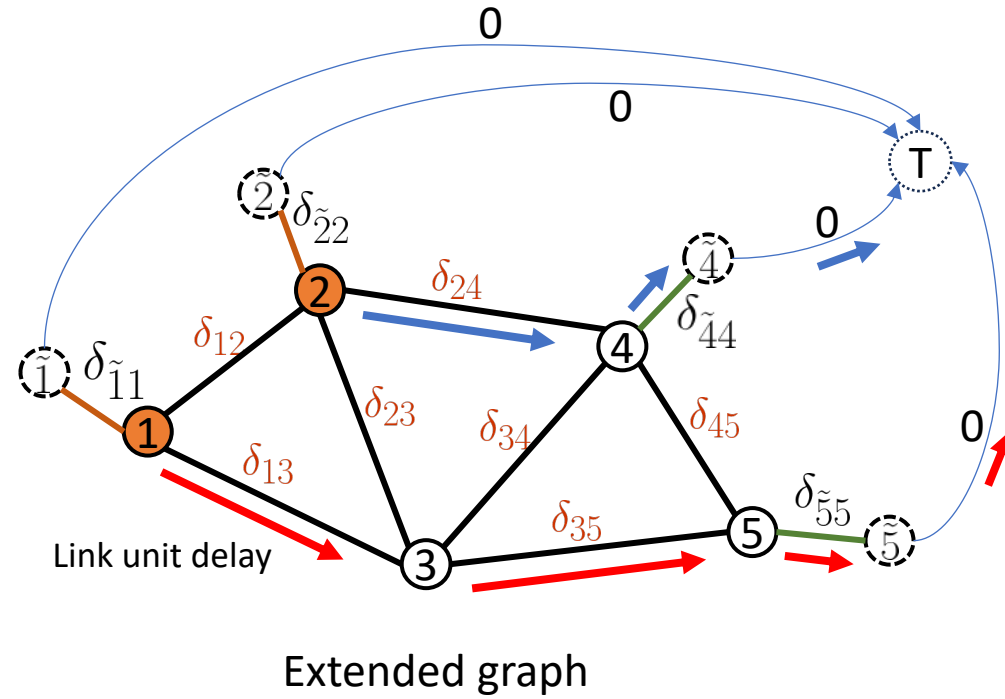
$$\delta^c = \frac{1}{\mu^c - \mathbf{x}^c}$$

long-term link service rate

Per-link traffic load

Virtual links are contention free

Step 2: distributed offloading & routing decisions



Clients (nodes 1, 2) find their own shortest paths to the virtual sink

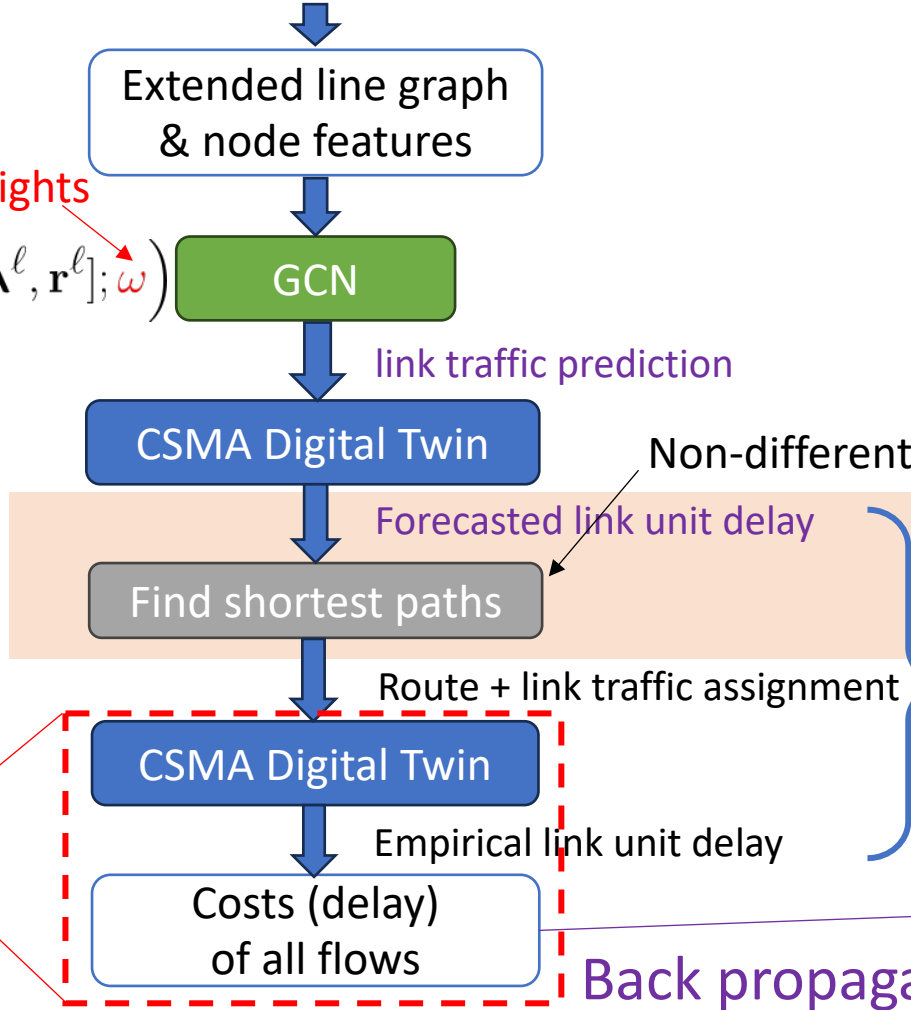
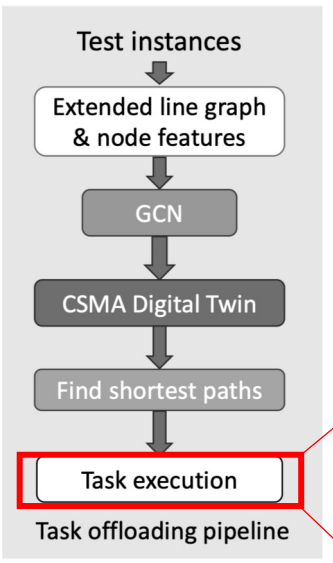
GCN training

Forward Draw random test instances

$$\min_{\omega} \mathbb{E} [J_1(\omega) + J_2(\omega)]$$

Intuition: Increasing the weight of a link can reduce its chance of being included in shortest path routing

$$\mathbf{x}^l = \Psi_{G^l}([\mathbf{q}^l, \mathbf{w}^l, \boldsymbol{\lambda}^l, \mathbf{r}^l]; \omega)$$



Non-differentiable Gradient Proxy

$$\nabla_{\delta_e} J_1(\omega) = \begin{cases} -\nabla_e J_1(\omega) & \text{if } e \text{ on the shortest paths} \\ 0 & \text{otherwise} \end{cases}$$

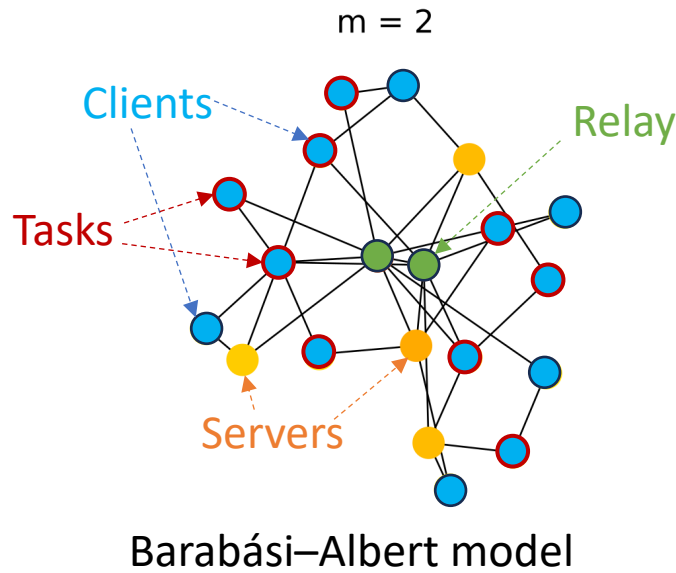
$J_2(\omega)$ Objective 2: mean squared error (MSE) regularization

$J_1(\omega)$ Objective 1: total cost of all flows (tasks)

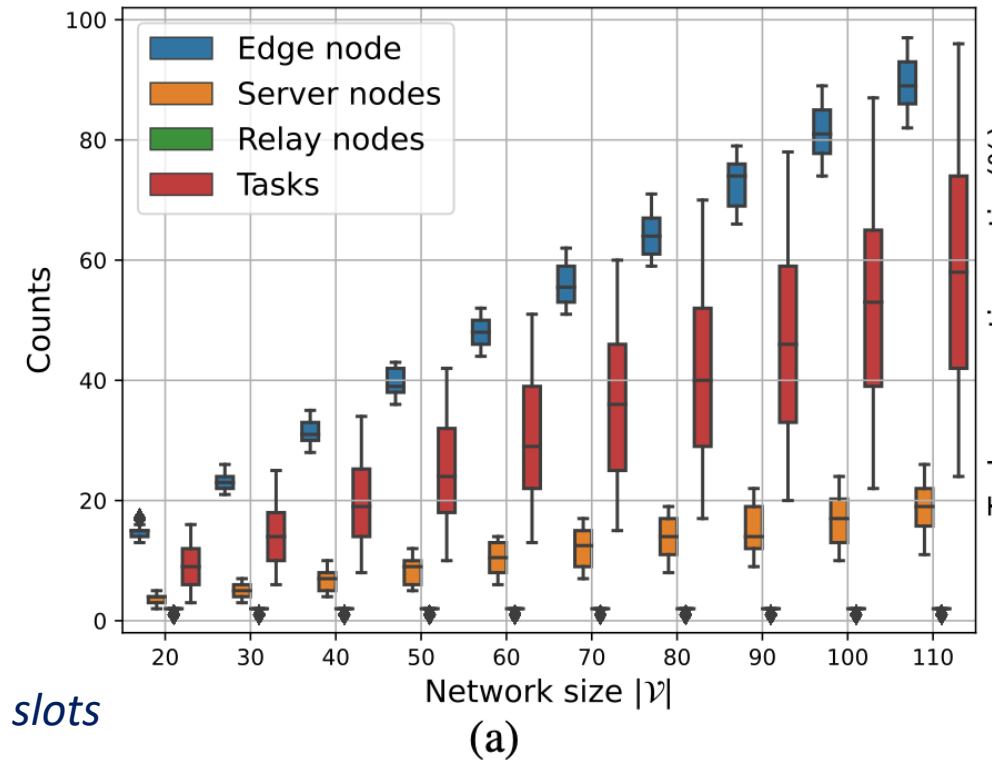
Back propagation + stochastic gradient descent

Numerical experiment

Random network topology and offloading instances



10 network sizes x 100 graphs x 10 offloading instances



$T = 1000$ time slots

Three offloading policies under test

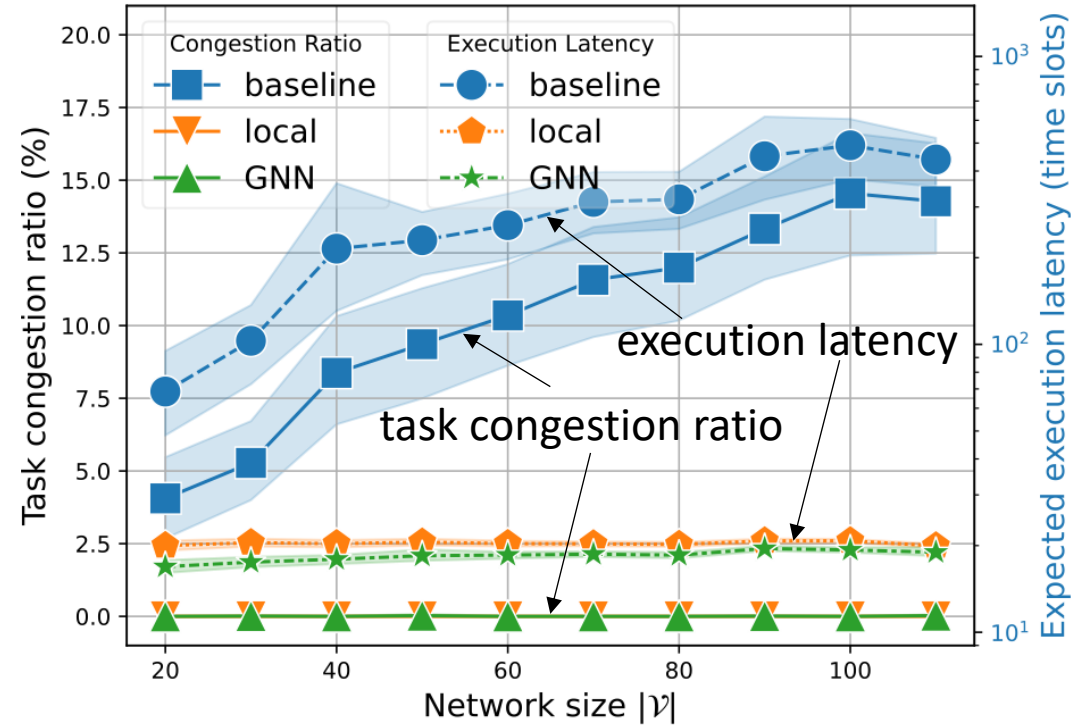
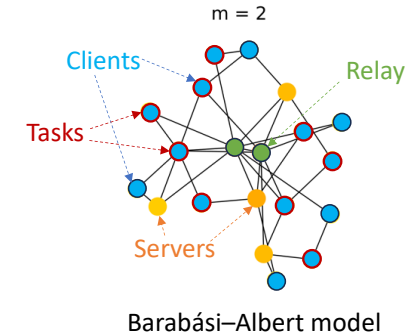
Local: every task is executed at its source client node

GNN: distributed greedy decisions based on link/server unit delay predicted by GNN

Baseline: distributed greedy decisions based on $(1/\text{link rate})$ – network context agnostic

$T = 1000$ time slots

Numerical results



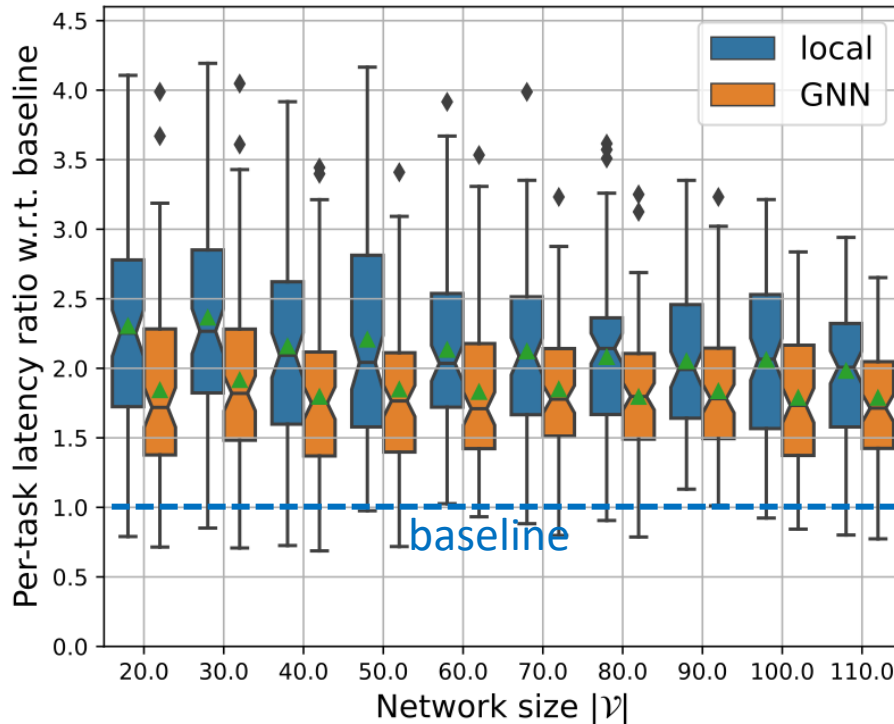
If a task is congested, its execution latency > 1000 time slots

- Local:** all clients can execute their own tasks without congestion
- GNN:** some tasks offloaded to remote servers without congestion, reducing average execution latency compared to the local policy
- Baseline:** 4%~15% congestion ratio, and high average execution latency (500)

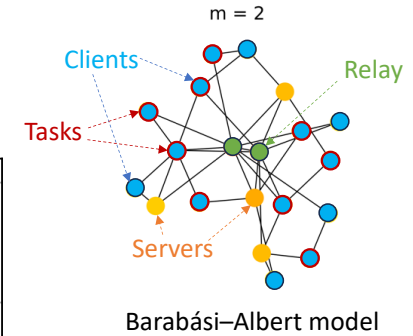
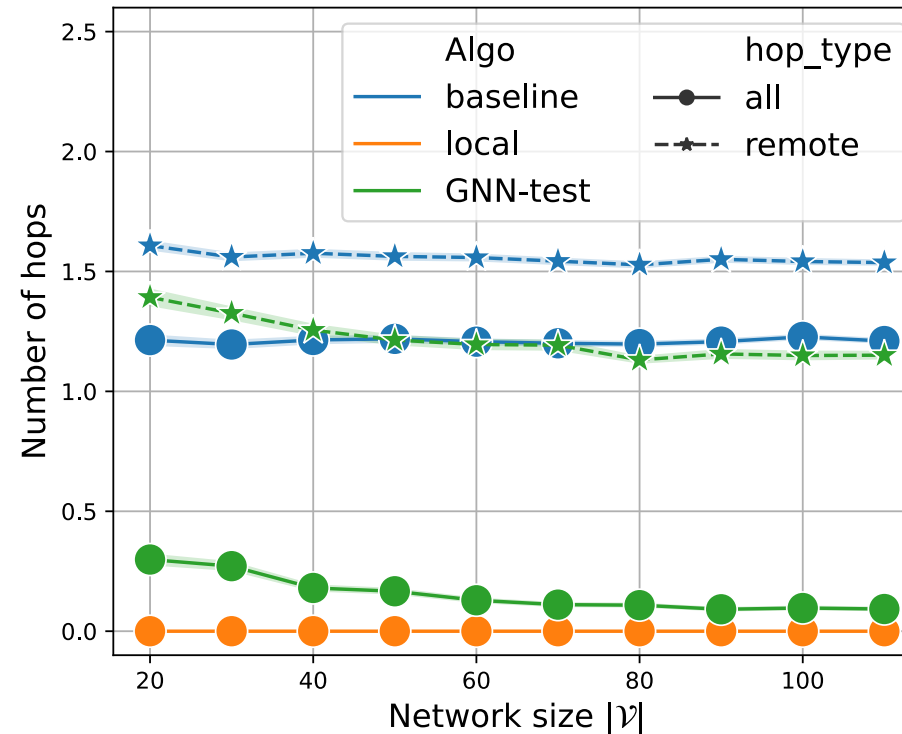
$T = 1000$ time slots

Individual task behaviors

Average per-task execution latency ratio



Average hop distance of servers



What about tasks that are NOT congested under the baseline policy?

- Baseline policy yields the lowest average execution time
- GNN is still better than local policy
- GNN favors nearby servers
- The result is still preliminary, GNN could be further improved!

Conclusions & future work

- Distributed task offloading + routing → shortest path routing
- Encode network context into edge weights
 - Graph convolutional neural networks
 - CSMA digital twin
- Mitigate congestion of concurrent flows of jobs
- **Future work**
 - Decision framework: iterative, probabilistic
 - Improve training approach
 - Trainable digital twin for other link schedulers
 - Evaluation on simulated queueing networks

