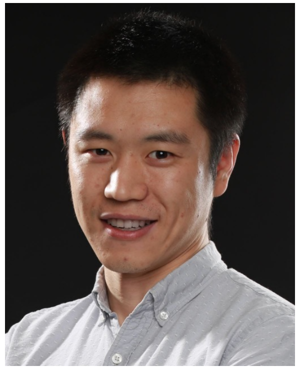
RICE UNIVERSITY

DEVCOM ARMY RESEARCH LABORATORY

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

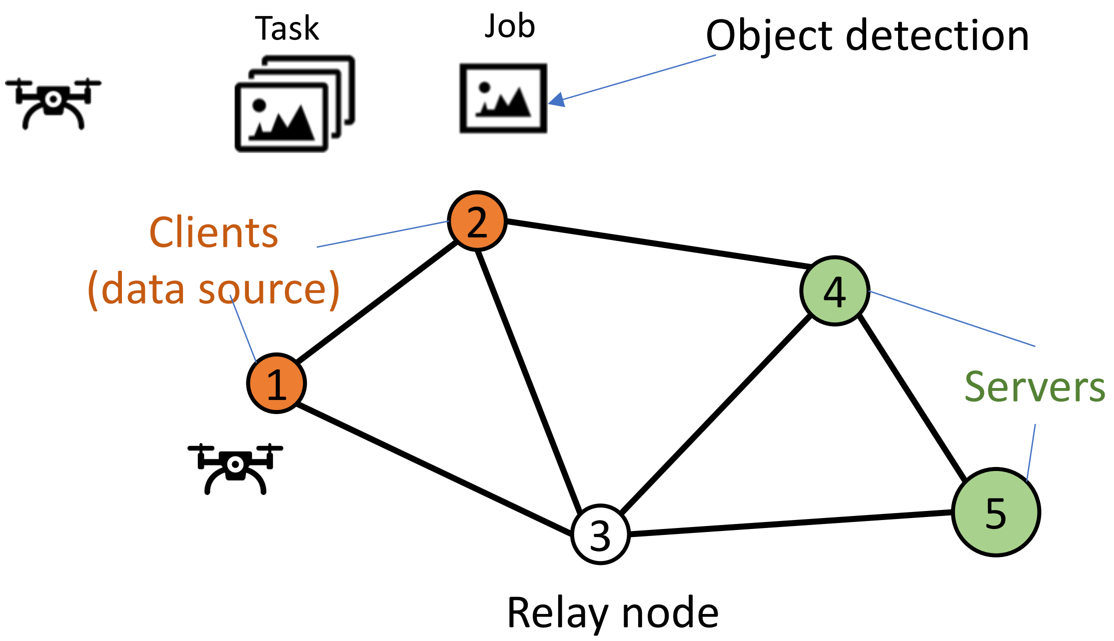
ICASSP 2024 KOREA

Seoul, Korea,

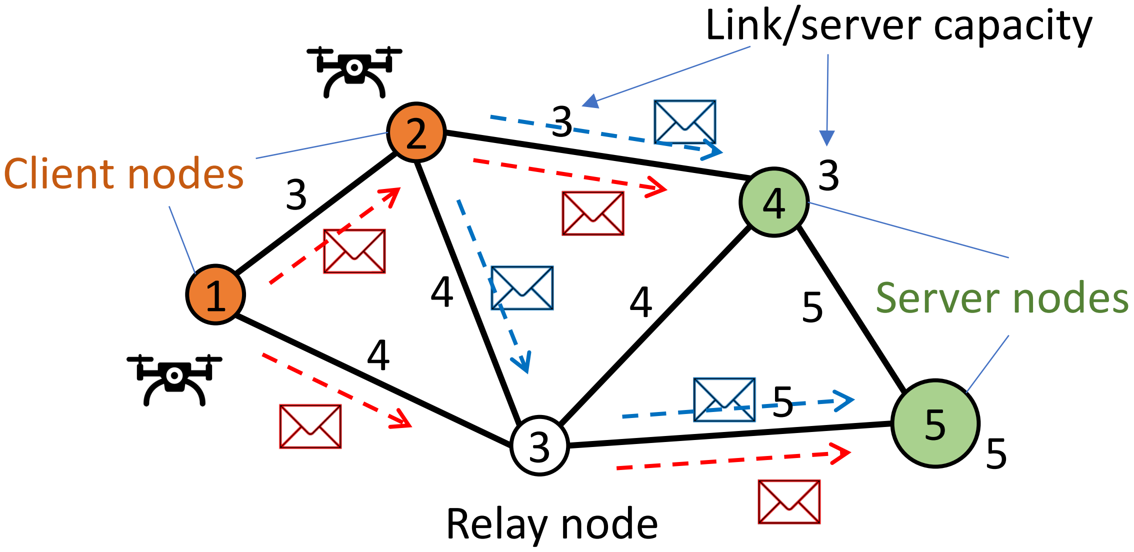
April 14-19, 2024

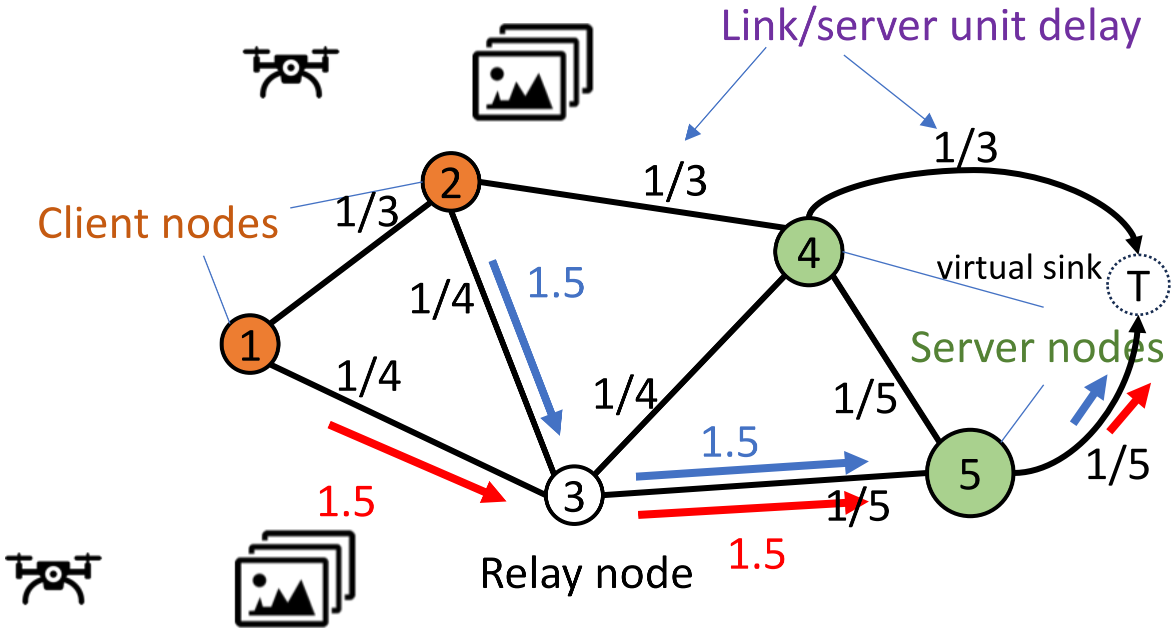
Distributed task offloading

- 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**

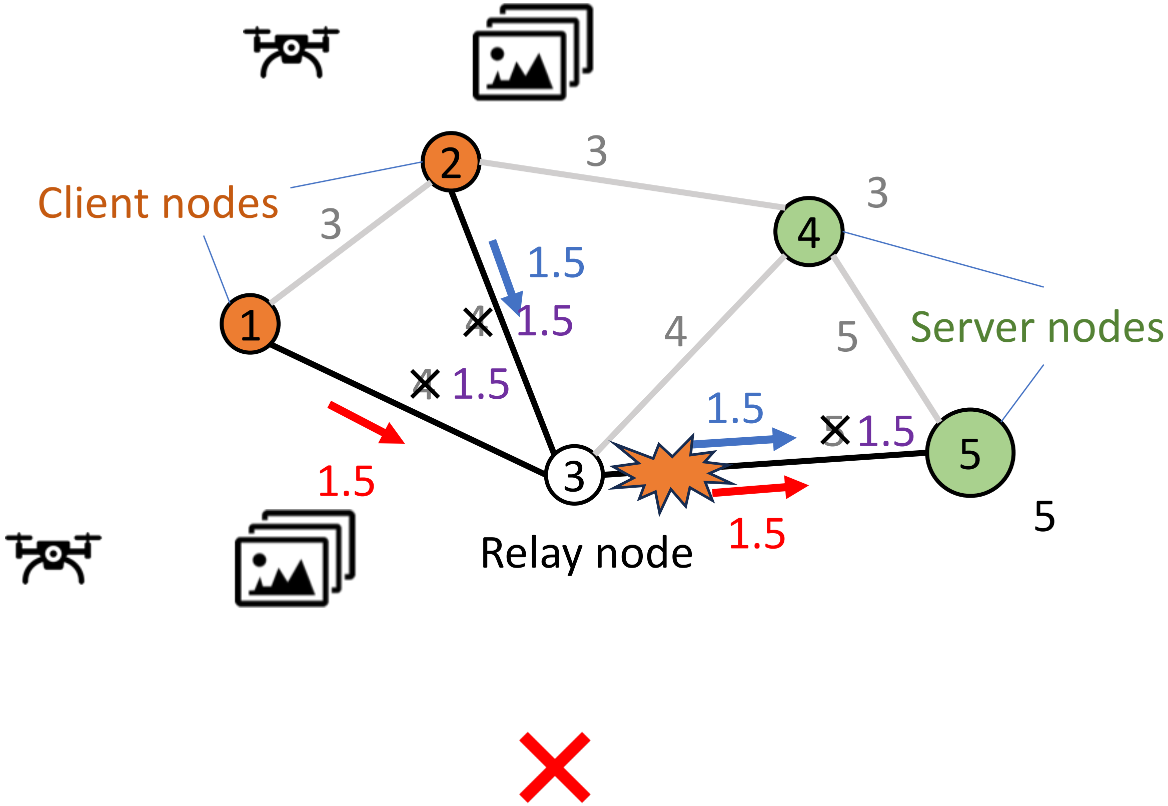
Step 2: clients send **task flows** via "the shortest path" to the virtual sink

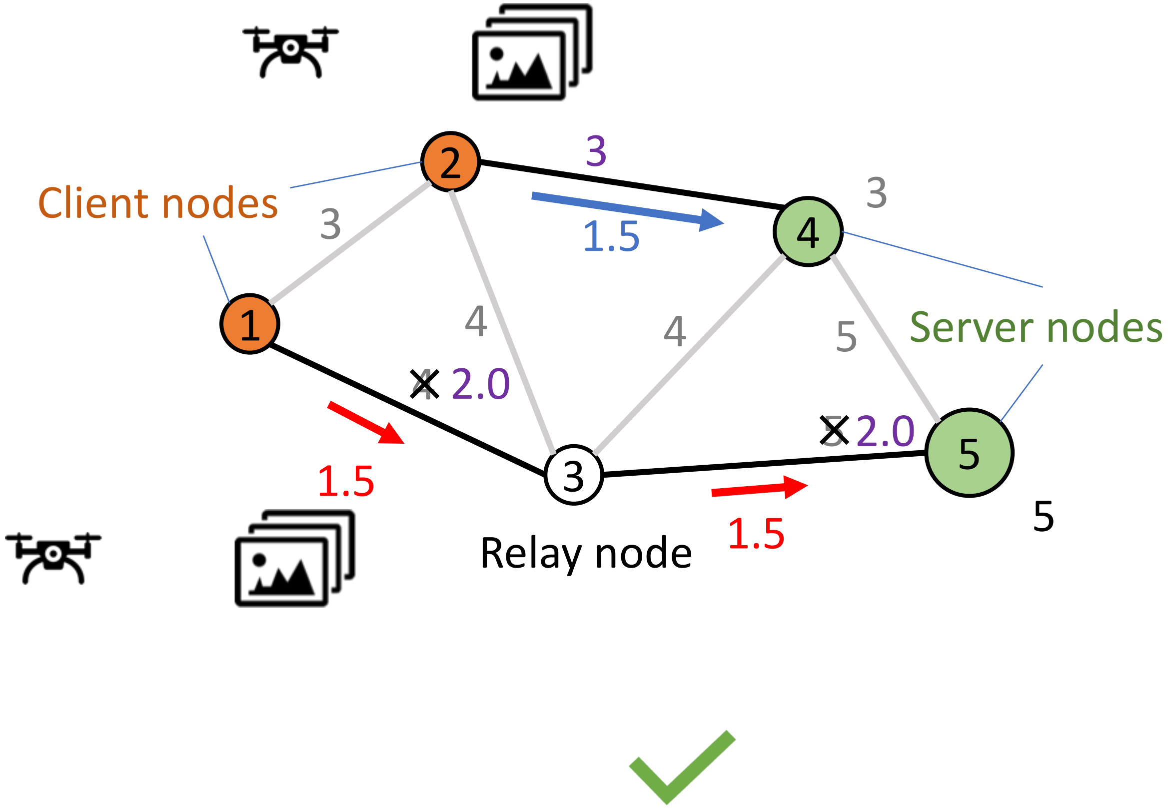
Minimize task response time

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

Consider per-link unit delay as edge weight

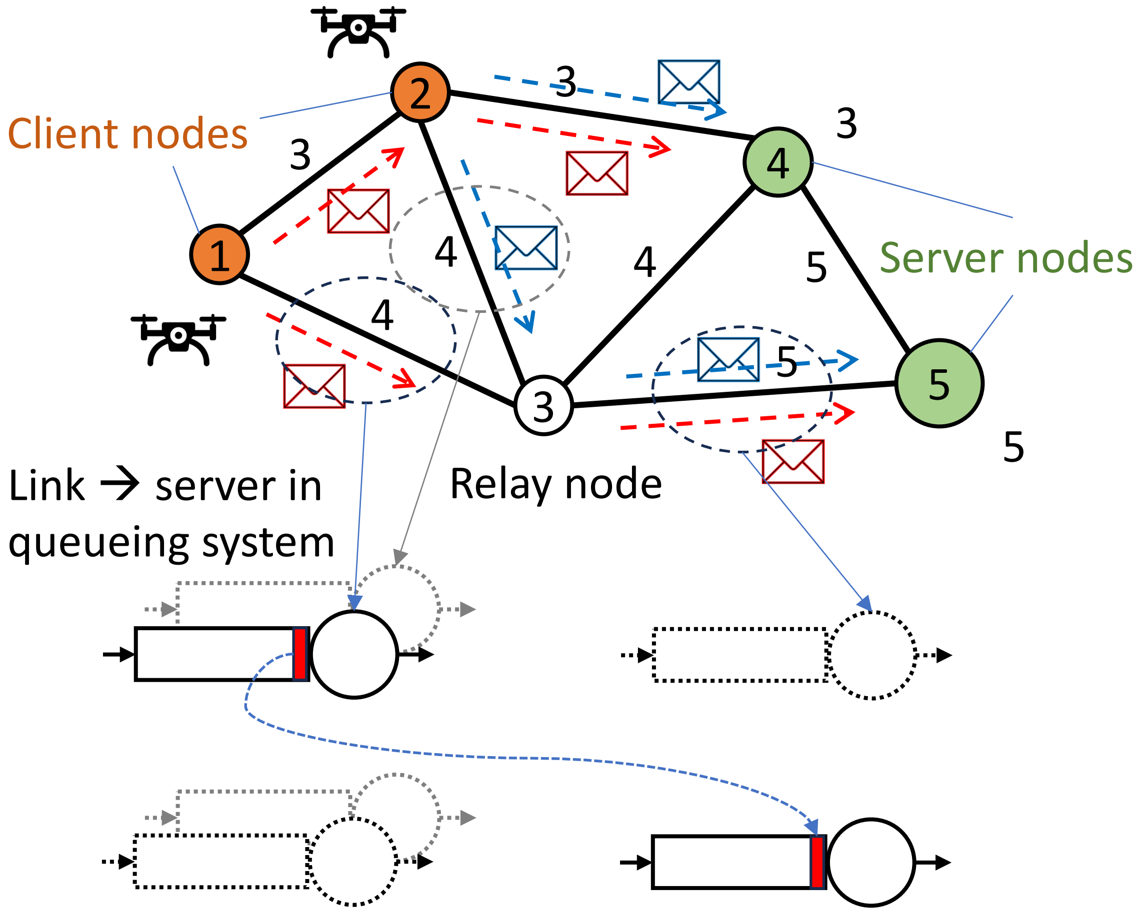
What could go wrong in wireless networks?



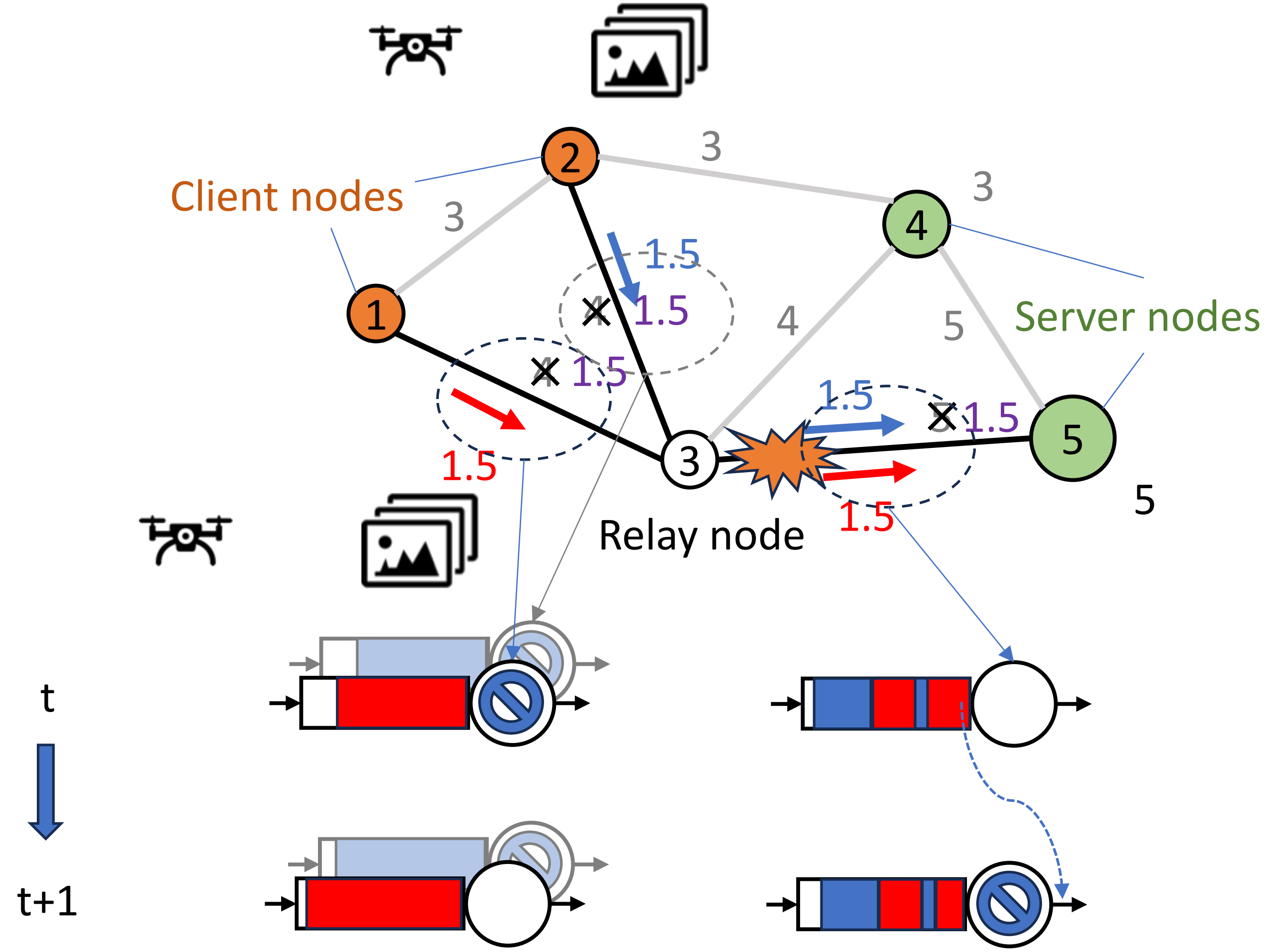


Link capacity changes once the streaming begins, depending on path selection & flow rate assignment

Queueing networks with interference constraints



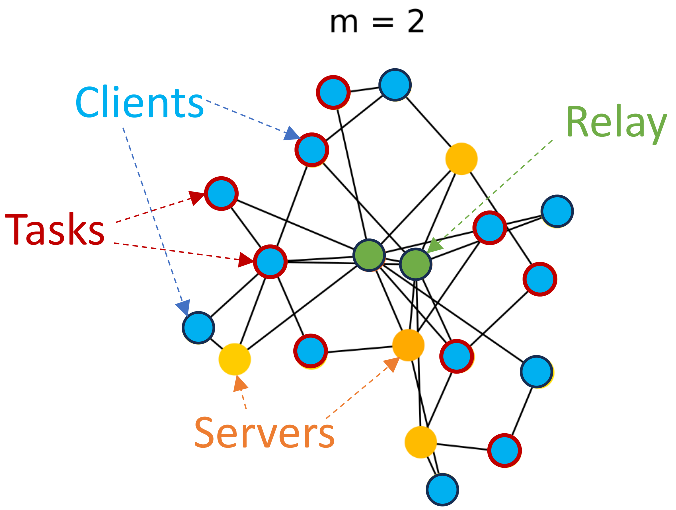
probing messages are **short-lived** flows



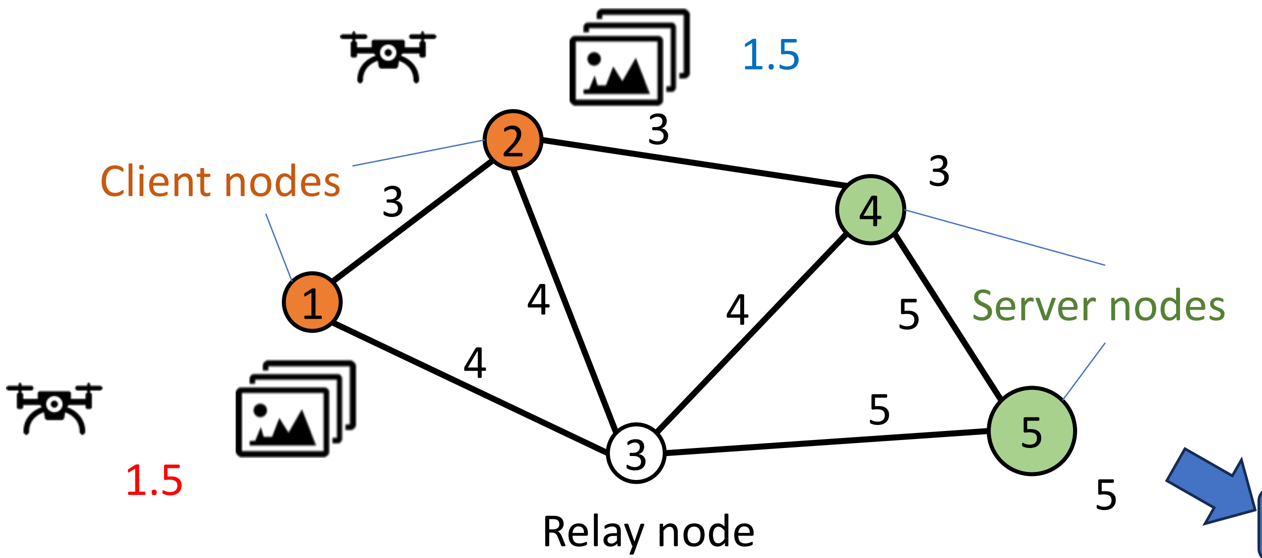
Links 13, 23, 35 **conflict** with each other, since node 3 has only one radio interface

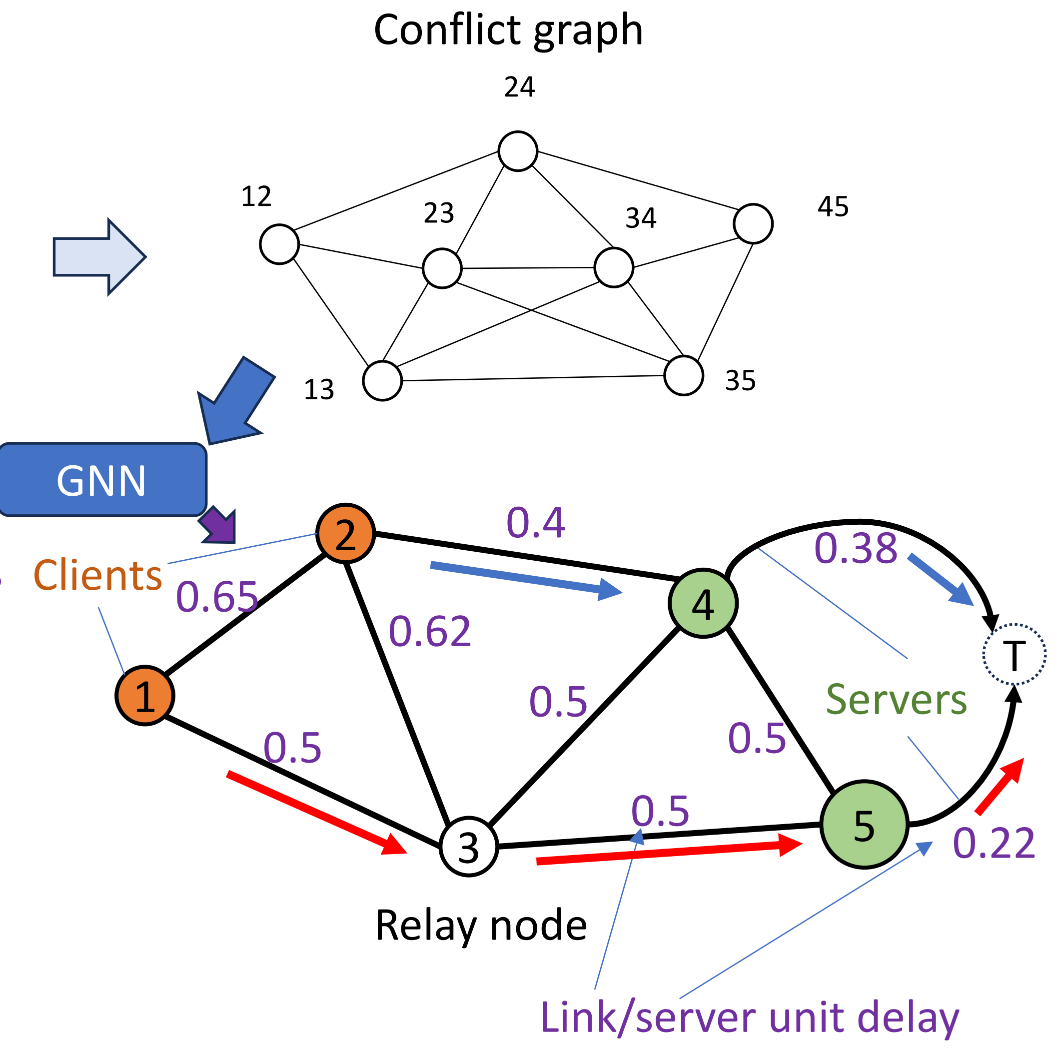
Alternative decision frameworks

- 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



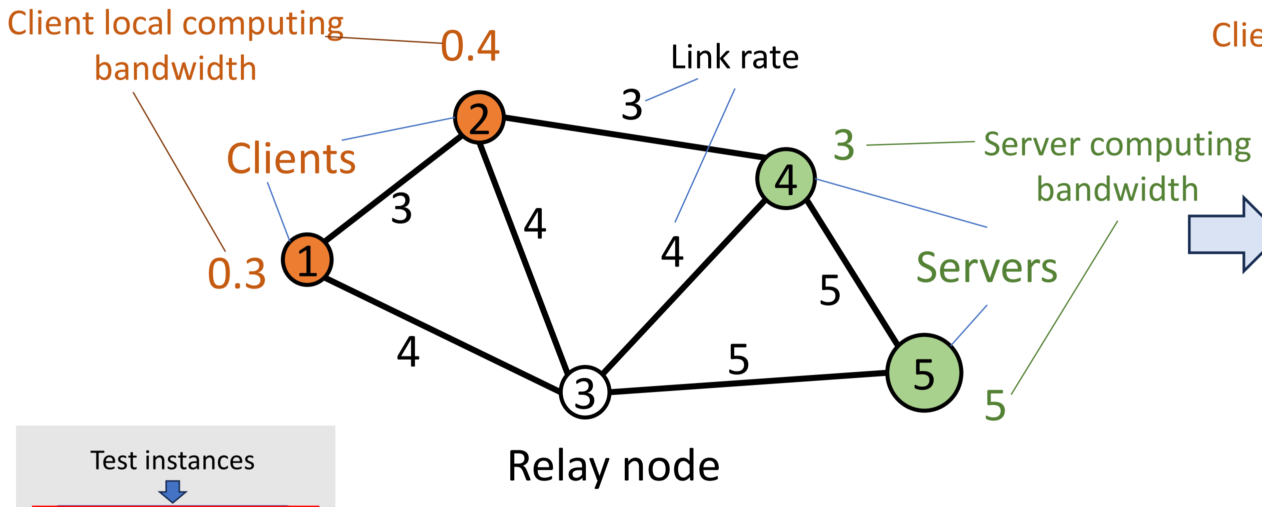
Our solution: keep shortest path decision, change *edge weights*

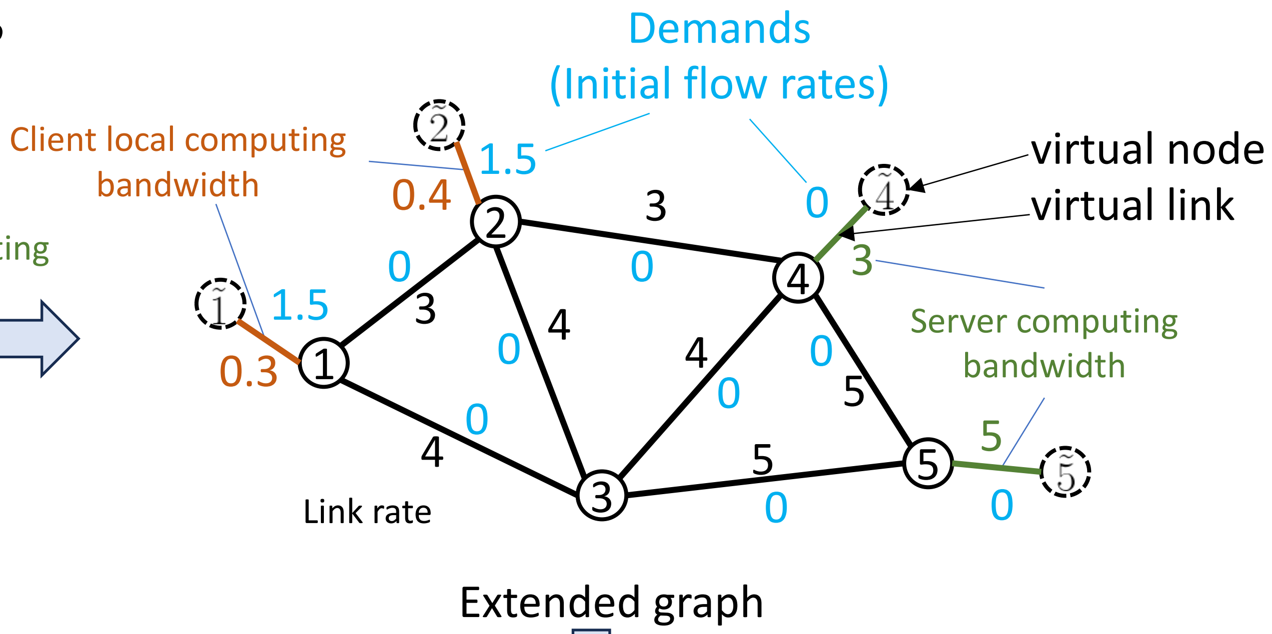


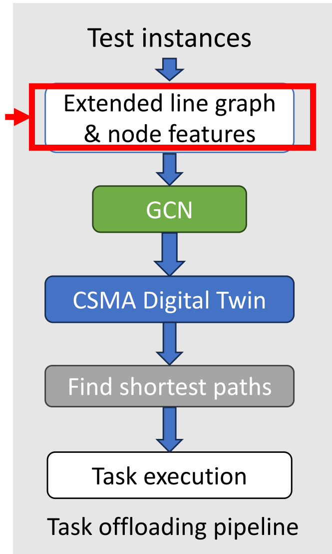


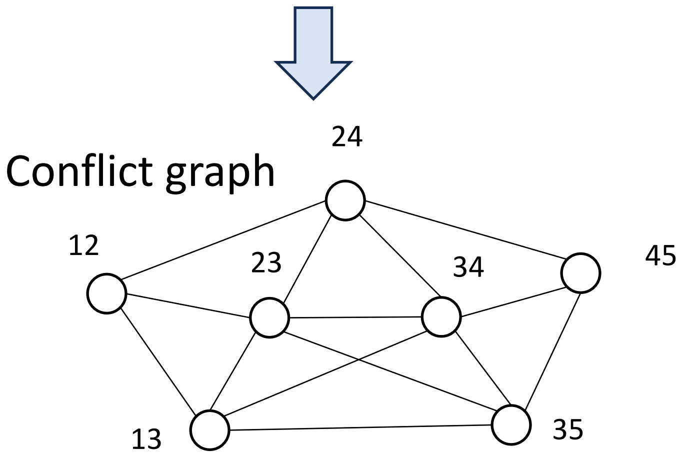
- Encode network context into congestion-aware edge weights
  - Using graph neural networks (GNNs)
  - To **improve the outcomes** of distributed greedy decisions
- What is network context?
  - Network topology
  - Average link rates
  - Interference relationships (conflict graph)
  - Demands: tasks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

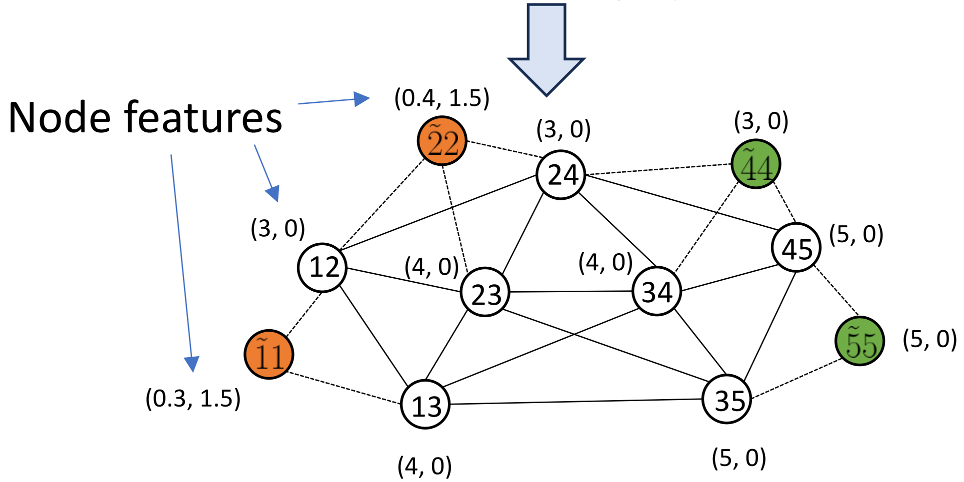
Our solutions: graph modeling







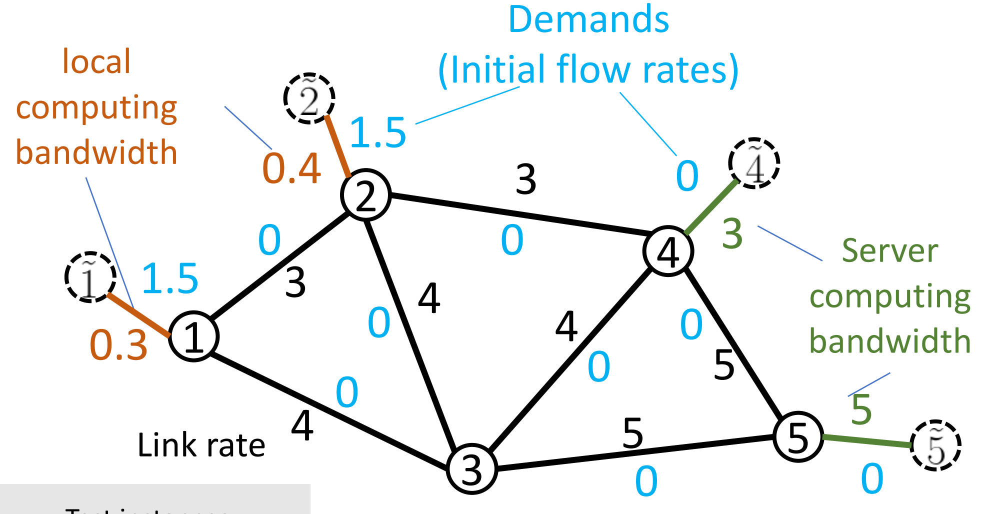




So that they can be processed by our GNN

Extended line graph

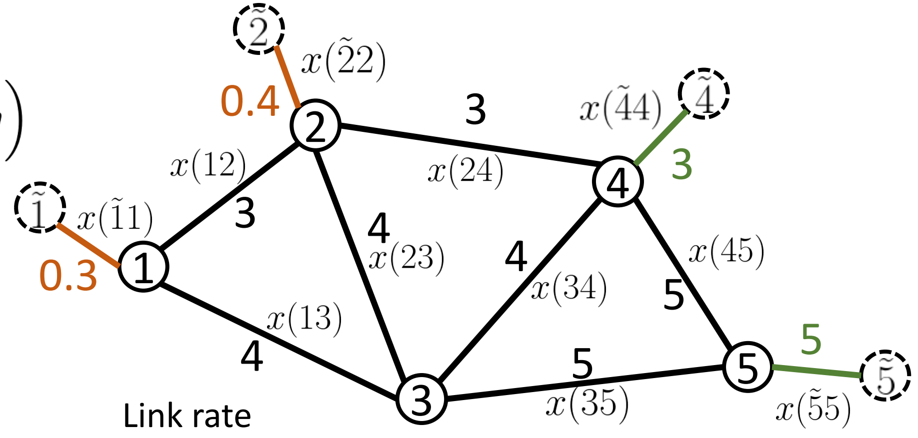
Step 1: edge weight prediction by GNN



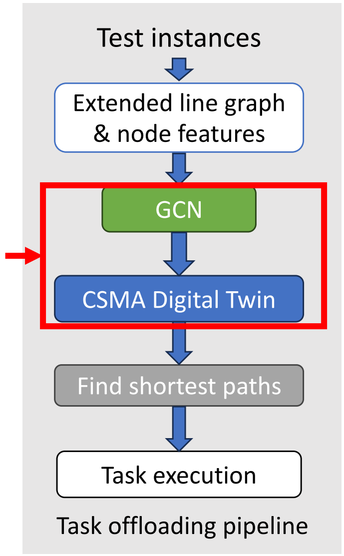
Extended graph

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

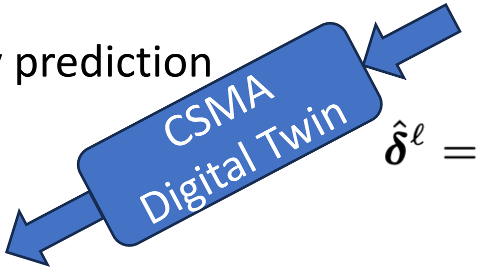
1.1 Traffic load prediction



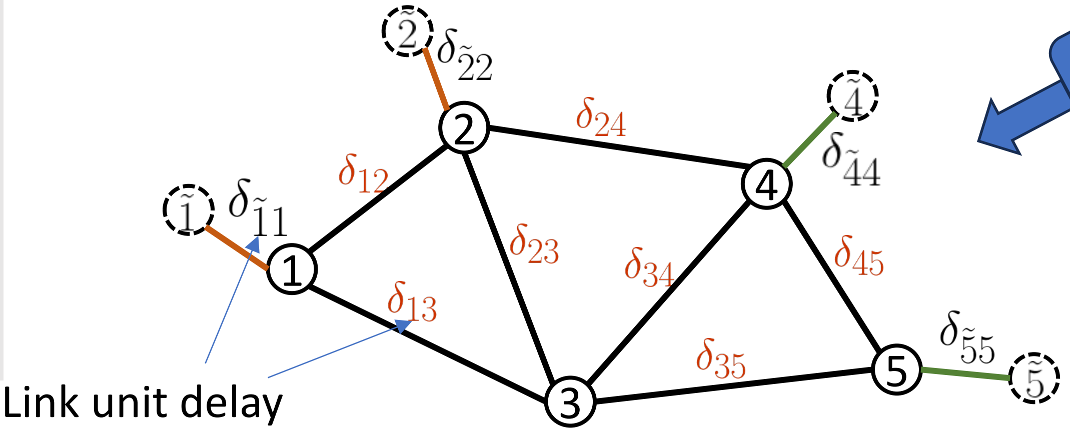
Extended graph



1.2 Link unit delay prediction

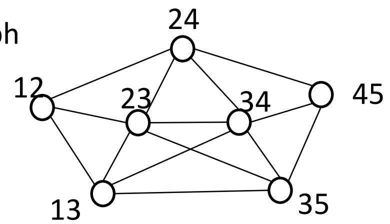


$$\hat{\boldsymbol{\delta}}^\ell = \varphi(\mathcal{G}^c, \mathcal{G}^\ell, \mathbf{r}^\ell, \mathbf{x}^\ell, T, K)$$



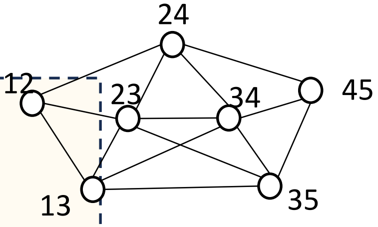
Extended graph

Conflict graph



CSMA Digital Twin

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



Inputs

- Conflict graph $\mathcal{G}^c \rightarrow \mathbf{A}^c$ (adjacency matrix)
- Per-link traffic load $\mathbf{x}^c$
- Average link rate (instantaneous link capacity) $\mathbf{r}^c$

Initially assume all links are busy $\mathbf{b}^c(0) = \mathbf{1}^c$

for $k \in \{0, \dots, K\}$ **do**

$$\boldsymbol{\mu}^c(k) = \frac{\mathbf{r}^c}{\mathbf{1} + \mathbf{A}^c \mathbf{b}^c(k)}$$

$$\mathbf{b}^c(k+1) = \min\left\{\frac{\mathbf{x}^c}{\boldsymbol{\mu}^c(k)}, \mathbf{1}^c\right\}$$

end for

long-term link service rate

Expected number of contending neighbors

Contention probability

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

$M/M/1$ queueing model

Non-empty queue $\frac{\lambda}{\mu}$

Expected time in the system $\frac{1}{\mu - \lambda}$

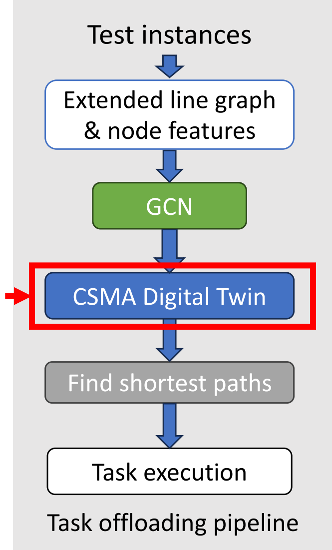
Output Unit delay on physical links

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

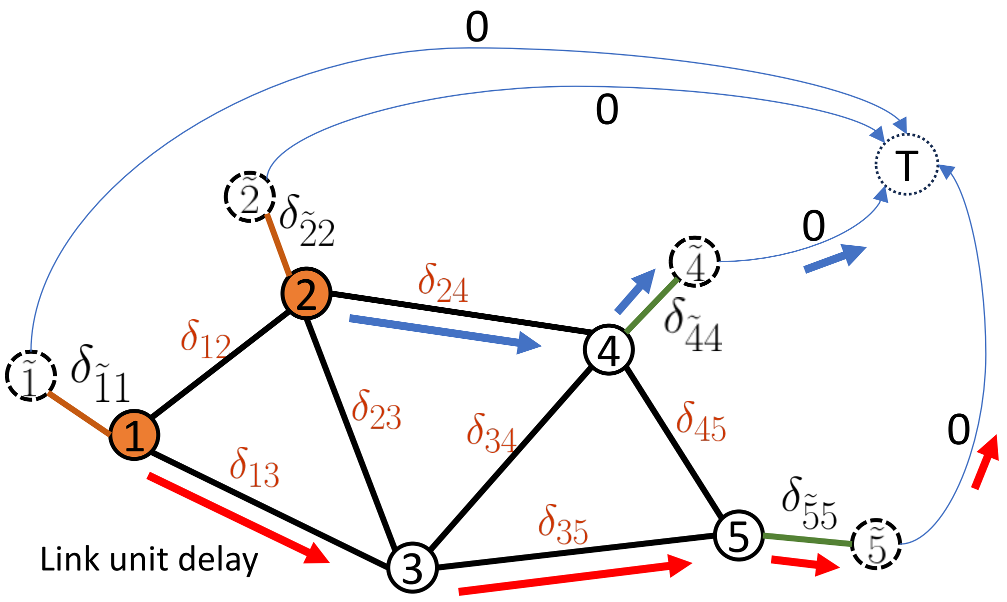
long-term link service rate

Per-link traffic load

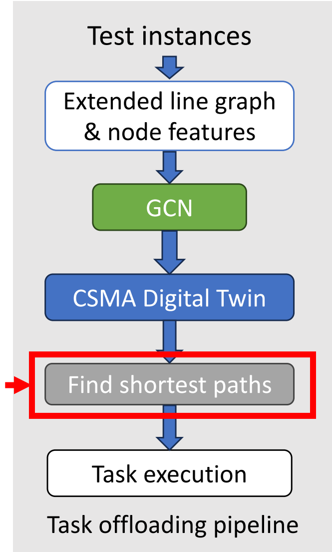
Virtual links are contention free



Step 2: distributed offloading & routing decisions

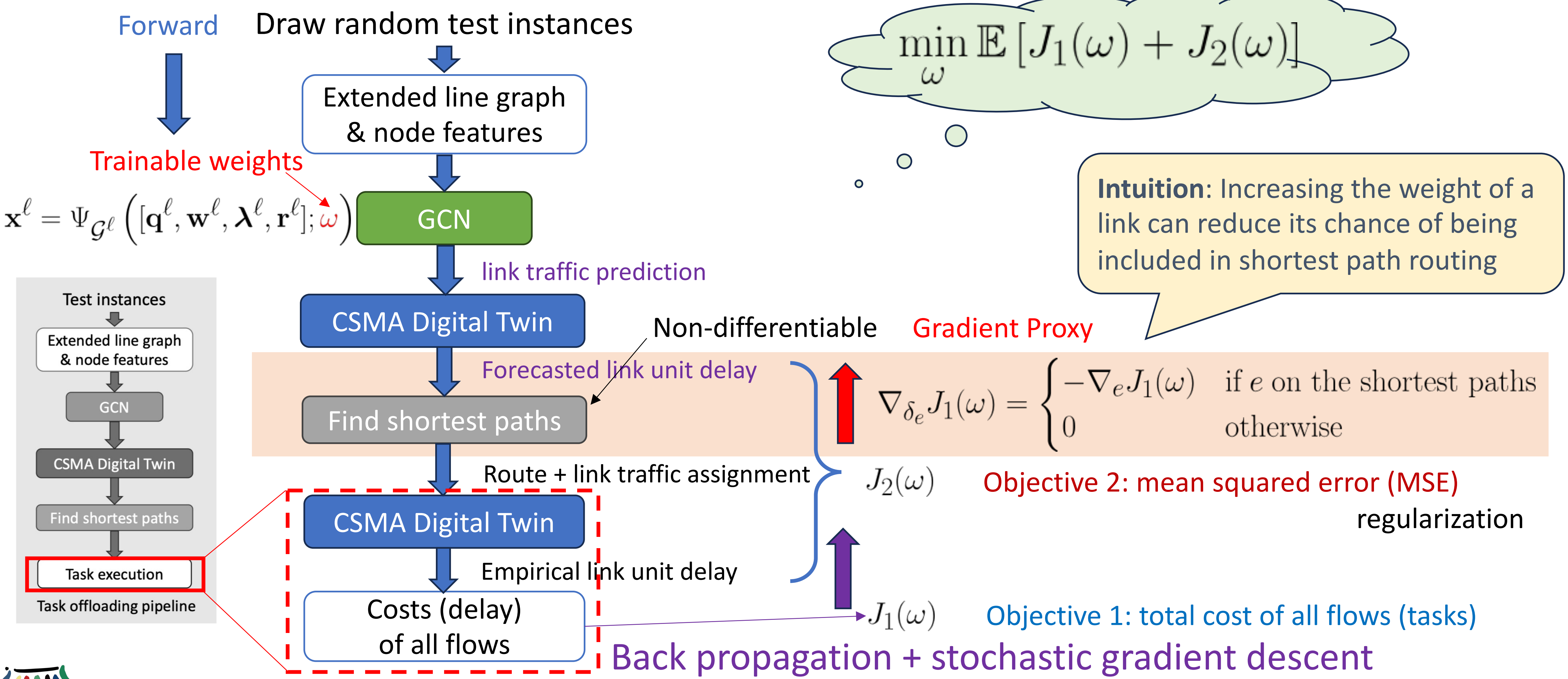


Extended graph



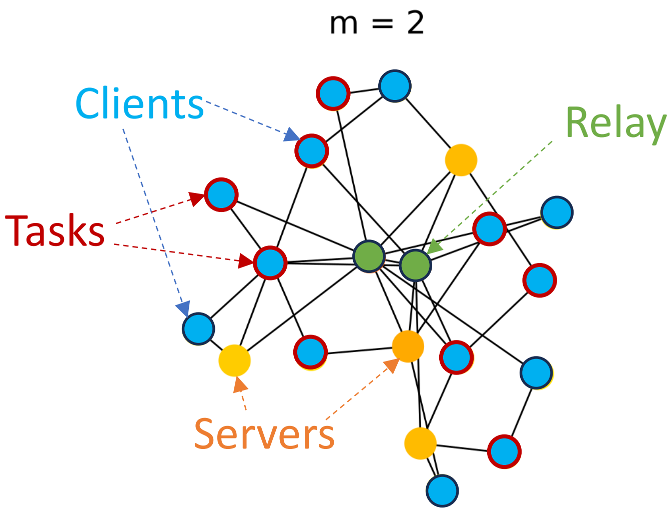
Clients (nodes 1, 2) find their own shortest paths to the virtual sink

GCN training



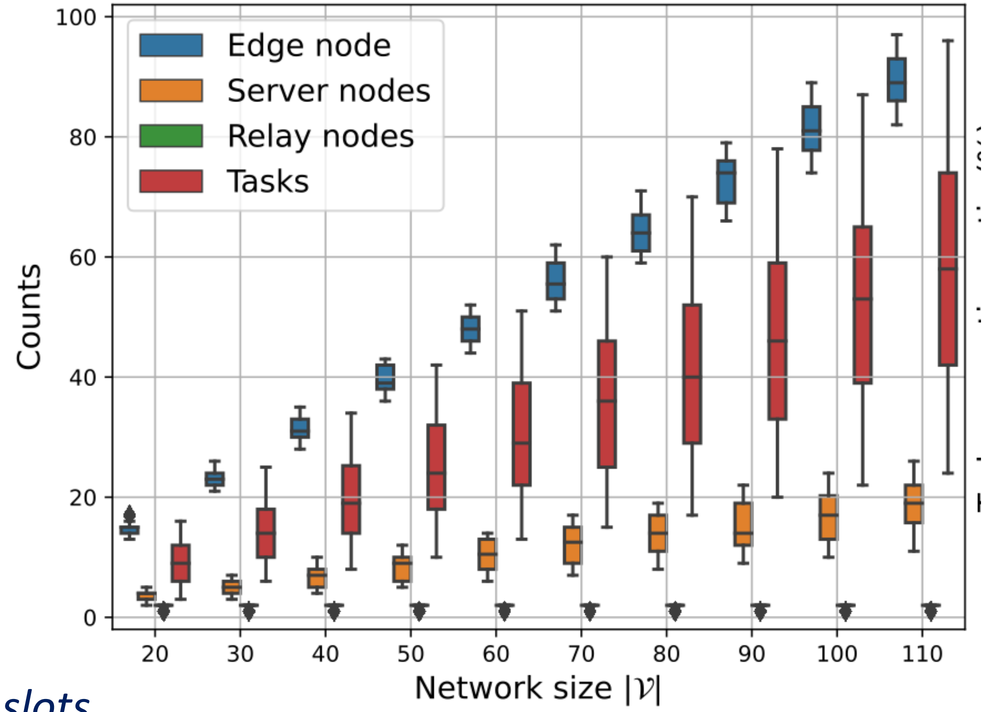
Numerical experiment

Random network topology and offloading instances



Barabási–Albert model

10 network sizes x 10 graphs x 10 offloading instances



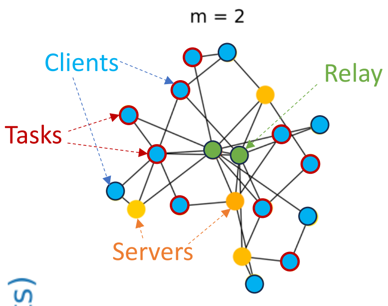
(a)

T = 1000 time slots

Three offloading policies under test

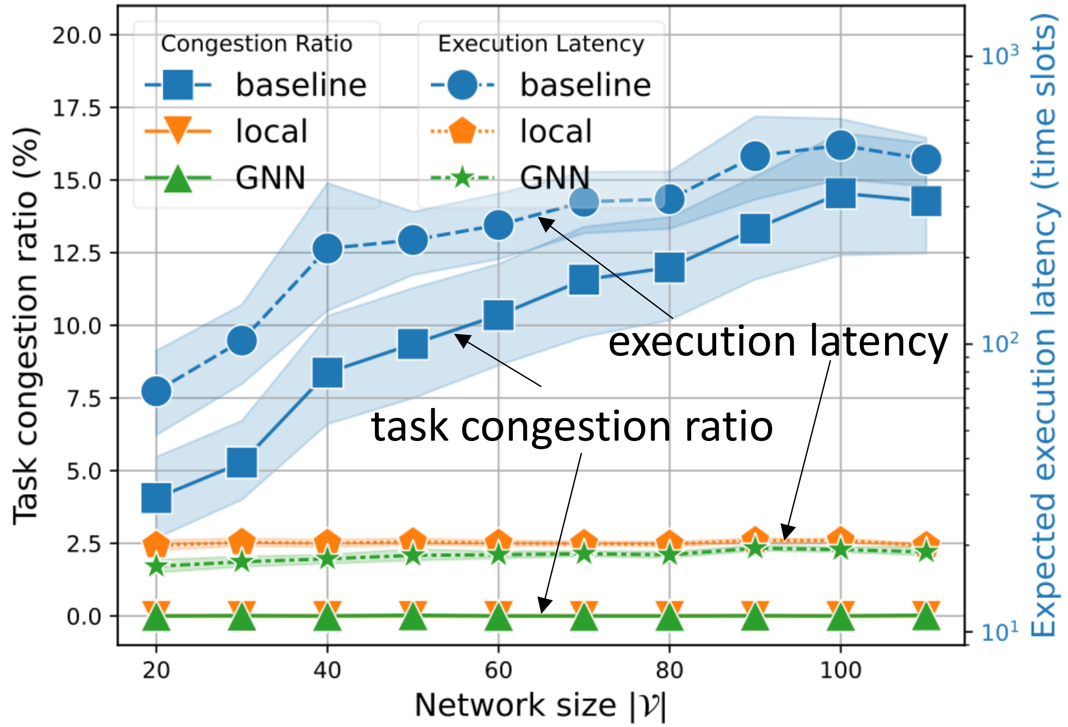
Local: every task is executed at its source client node
GNN: distributed greedy decisions based on link/server unit delay predicted by GNN
Baseline: distributed greedy decisions based on (1/link rate) – network context agnostic

Numerical results

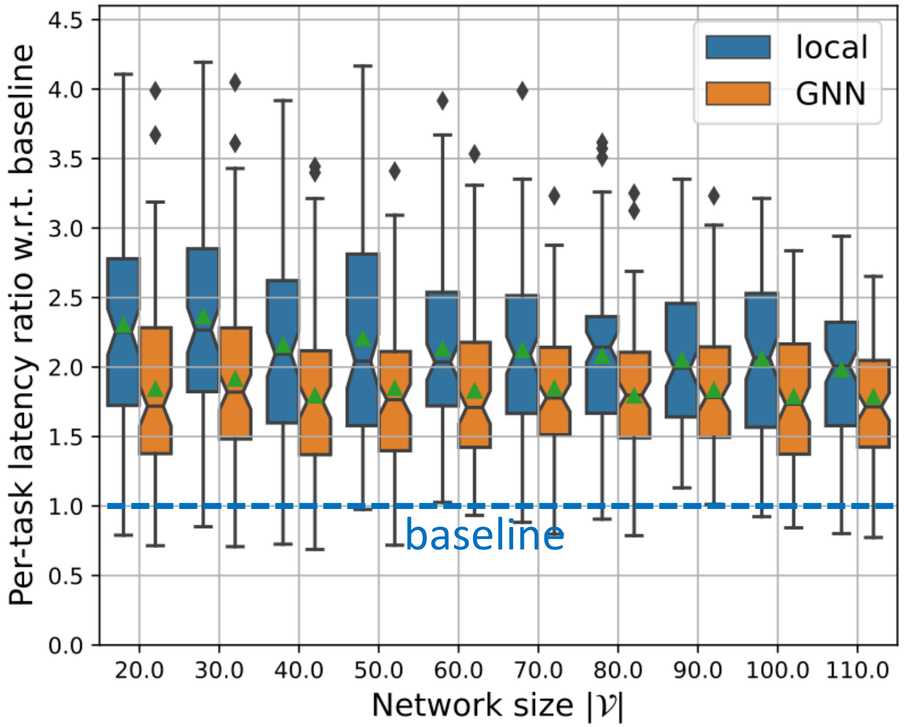


Barabási–Albert model

T = 1000 time slots



Average per-task execution latency ratio



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)

For a task NOT congested under the baseline, its execution time under local and GNN policies is longer, when everything else the same.
- GNN is still better than local policy
- Room for improvement

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

Paper

Code

Thank You!