Stabilizing Queuing Network with Model Data-Independent Control

Qian Xie
qx66@cornell.edu

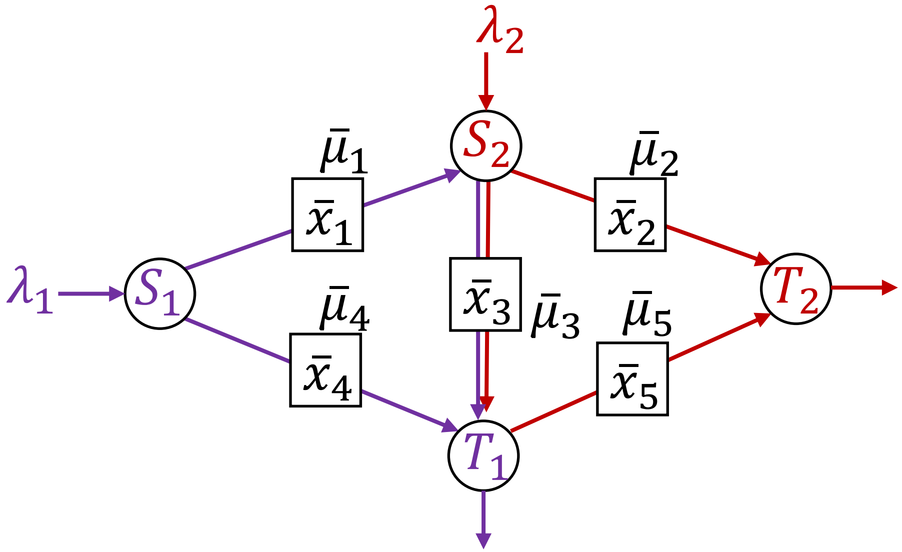
Joint work with Li Jin (SJTU, NYU)

Robust routing for queuing networks

- In practical settings, model data (arrival/service rates) may be
  - unavailable
  - hard to estimate

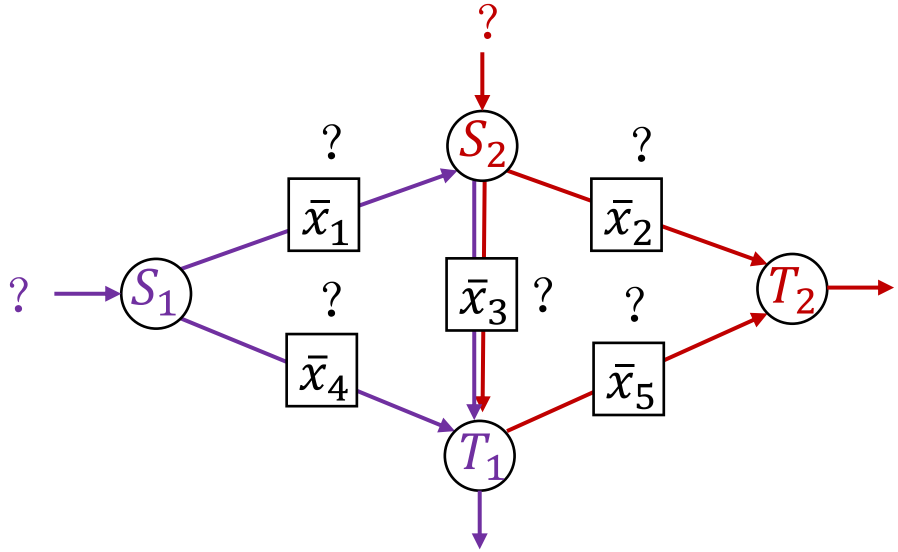
Robust routing for queuing networks

- In practical settings, model data (arrival/service rates) may be
  - unavailable
  - hard to estimate
- Suppose that we know the traffic state and network topology, but not the demand and supply



Robust routing for queuing networks

- In practical settings, model data (arrival/service rates) may be
  - unavailable
  - hard to estimate
- Suppose that we know the traffic state and network topology, but not the demand and supply



Learning-based vs. robust control

- How to make queuing control decisions in an unknown environment?

Learning-based vs. robust control

- How to make queuing control decisions in an unknown environment?
- Solution 1: learn the environment from observation
  - learning-based adaptive control
  - efficient & smart
  - require sufficient data
  - vulnerable to unhealthy data

Learning-based vs. robust control

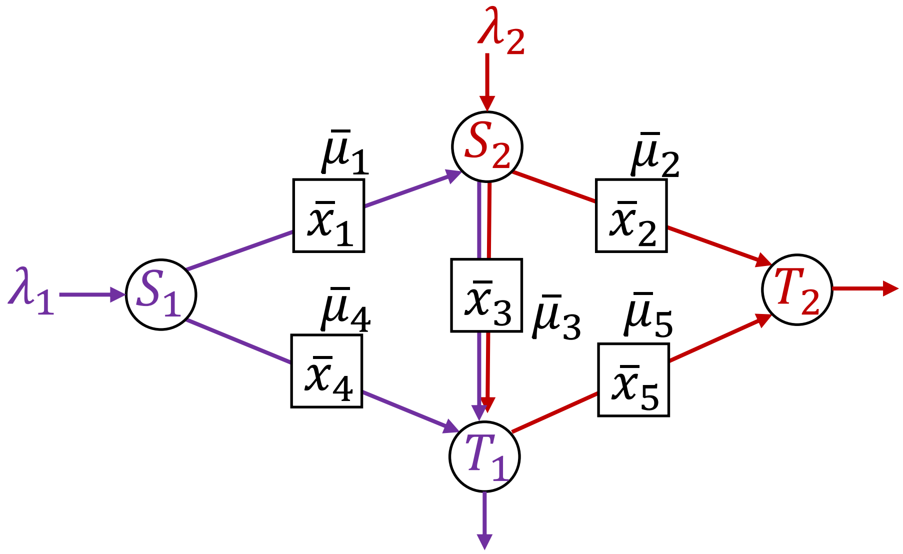
- How to make queuing control decisions in an unknown environment?
- Solution 1: learn the environment from observation
  - learning-based adaptive control
  - efficient & smart
  - require sufficient data
  - vulnerable to unhealthy data
- Solution 2: independent of environment parameters
  - robust control
  - easy & robust
  - guarantee stability but not efficiency
  - resist modeling error and/or non-stationary environment

Learning-based vs. robust control

- How to make queuing control decisions in an unknown environment?
- Solution 1: learn the environment from observation
  - learning-based adaptive control
  - efficient & smart
  - require sufficient data
  - vulnerable to unhealthy data
- Solution 2: independent of environment parameters
  - robust control
  - easy & robust
  - guarantee stability but not efficiency
  - resist modeling error and/or non-stationary environment
- Solution 2 motivates model-based independent control

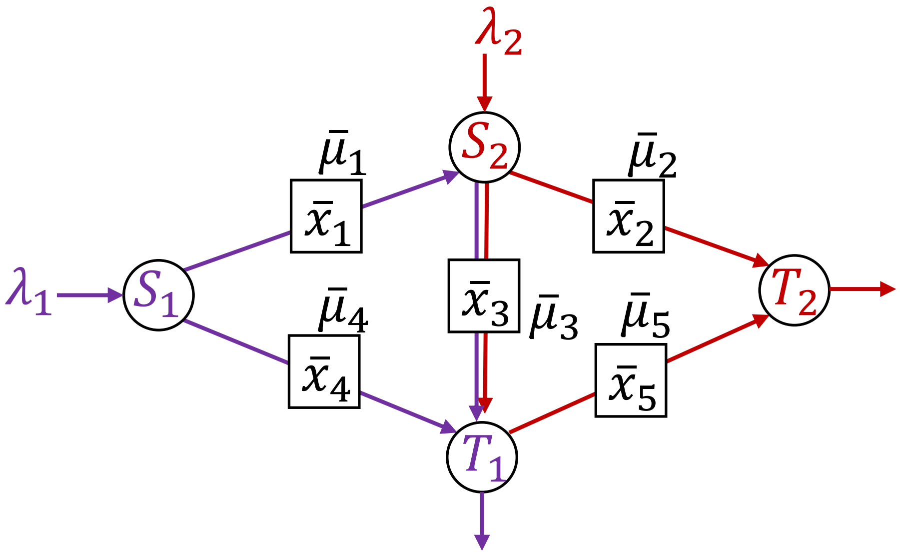
Setting

- Multi-class Jackson queueing network
  - Open, acyclic, multiple origin-destination (OD)
  - Poisson arrivals & exponential service times
  - Real-time OD-specific queue sizes can be observed



Setting

- Multi-class Jackson queueing network
  - Open, acyclic, multiple origin-destination (OD)
  - Poisson arrivals & exponential service times
  - Real-time OD-specific queue sizes can be observed
- MDI control actions
  - Routing
  - Sequencing
    - e.g., FCFS, preemptive-priority
  - Holding



Main results

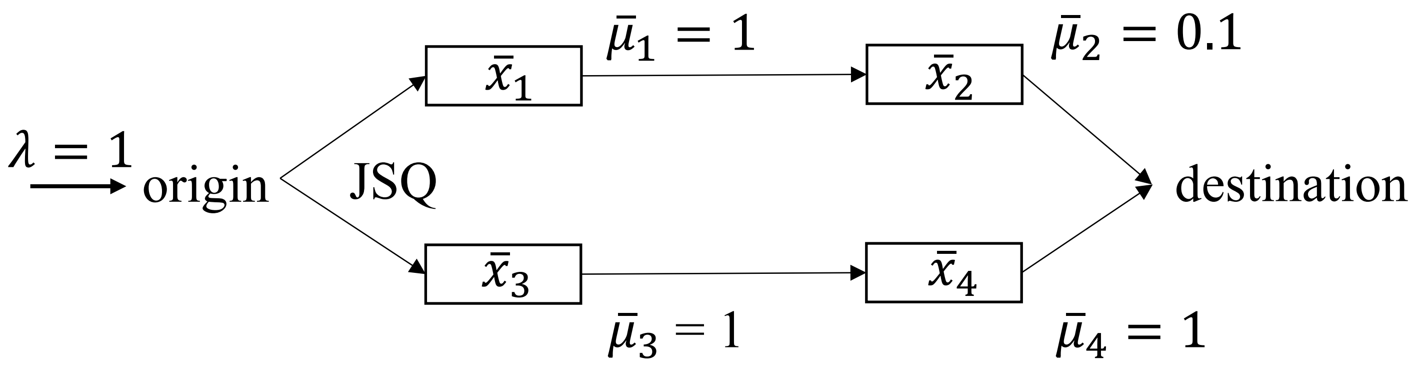
1. Easy-to-use criterion to check the stability of a multi-class network under a given MDI control policy
2. Stabilizing centralized MDI routing + sequeuncing policy for multi-class network (JSR)
3. Stabilizing decentralized MDI routing + holding policy for single-class network (JSQ-AS)

Naïve policy: JSQ

- Simple case: parallel queues
- Intuitive routing policy: join the shortest queue (JSQ)
  - Route the arrival to the shortest queue
  - Ties are broken uniformly at random
- Standard results:
  - System is stable if and only if arrival rate < total service rate
  - Optimal for symmetric servers
- JSQ is MDI, decentralized and throughput-maximizing

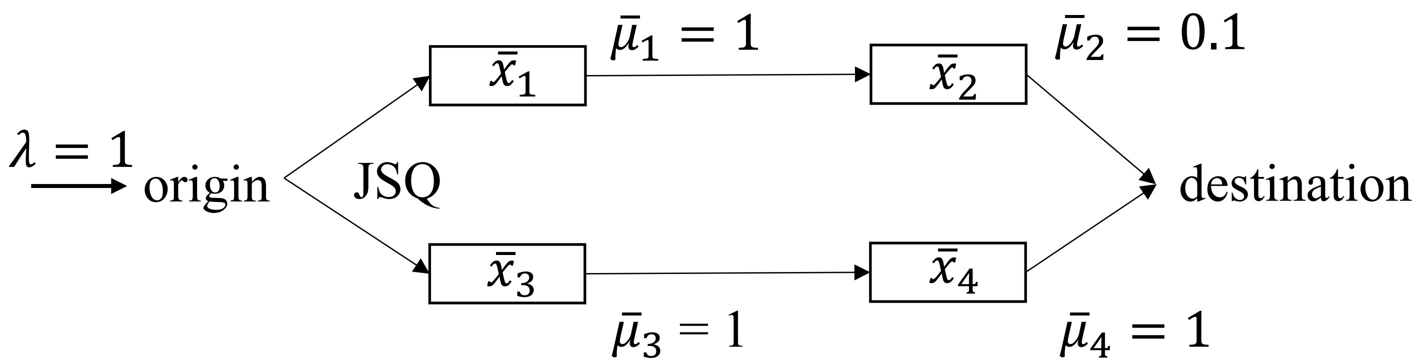
JSQ fails for networks

- What if we directly extend JSQ to networks?



JSQ fails for networks

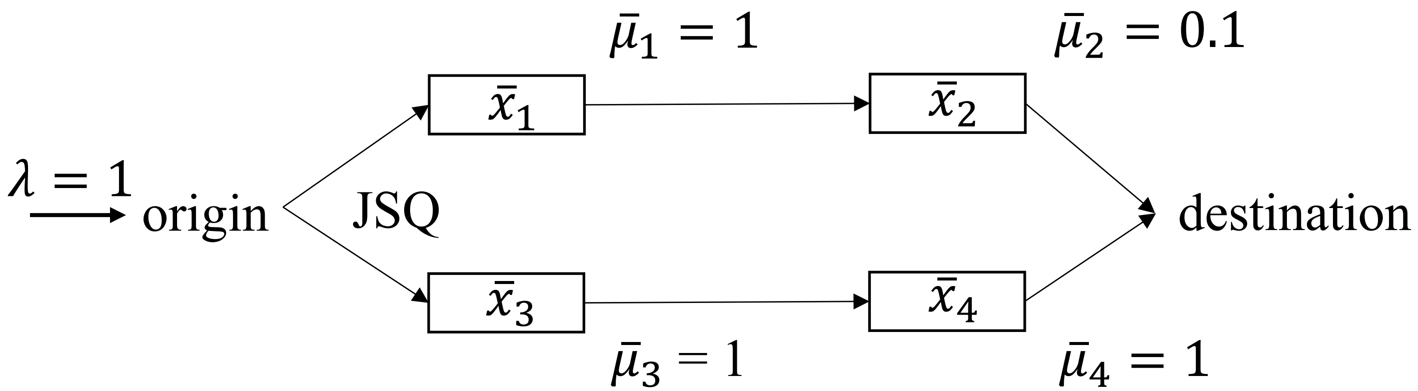
- What if we directly extend JSQ to networks?



- By symmetry & Burke's theorem, departure process from servers 1 & 3 are both Poisson(0.5)
- However, 0.5 > 0.1 (service rate of server 2)
- Thus, the network is unstable!

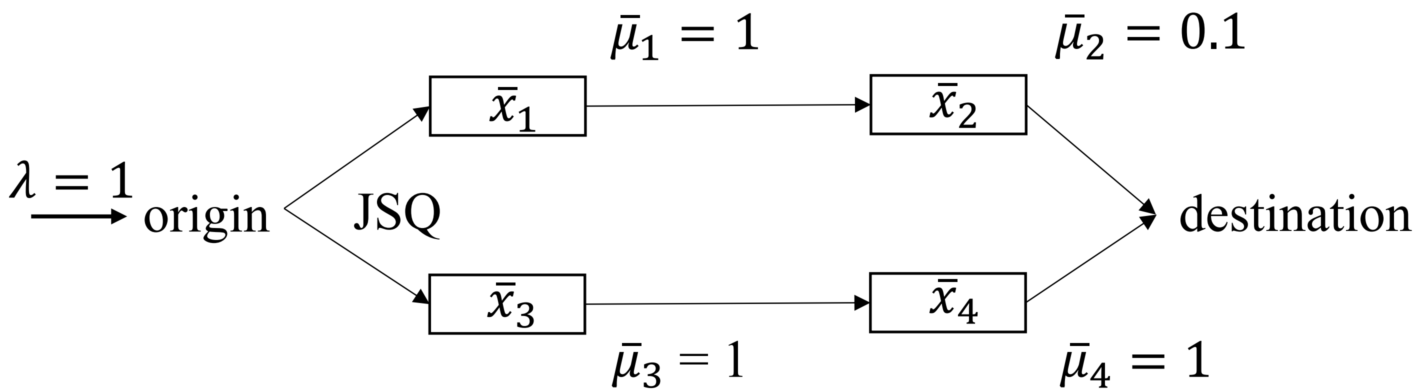
JSQ fails for networks

- Why JSQ fails?
  - Server 2 will be congested, but such information is not used at the origin



Solution: JSR

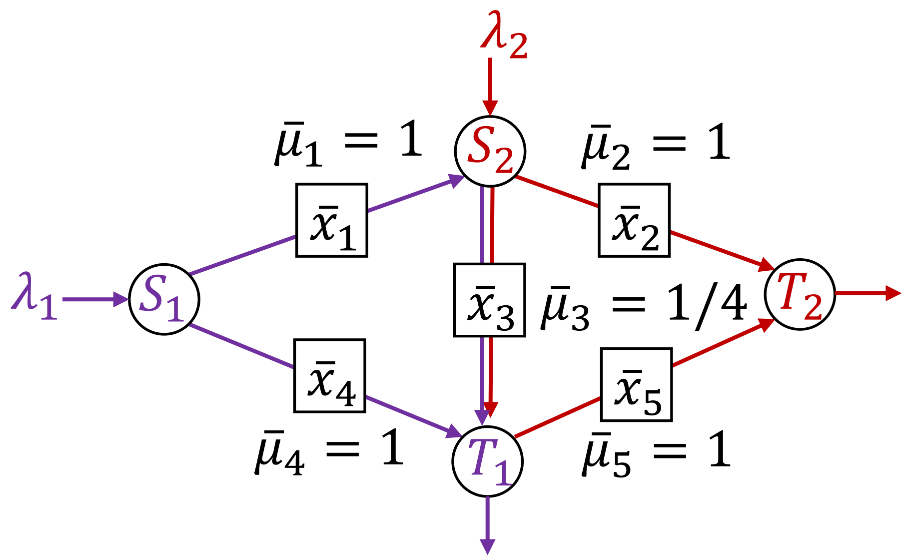
- Why JSQ fails?
  - Server 2 will be congested, but such info is not used at the origin



- To fix this, consider the total queue sizes on each route:
  - Join queue 1 if $\bar{x}_1 + \bar{x}_2 < \bar{x}_3 + \bar{x}_4$
  - Join queue 3 if $\bar{x}_1 + \bar{x}_2 > \bar{x}_3 + \bar{x}_4$
  - Ties broken uniformly at random
- Join the shortest queue (JSR)!

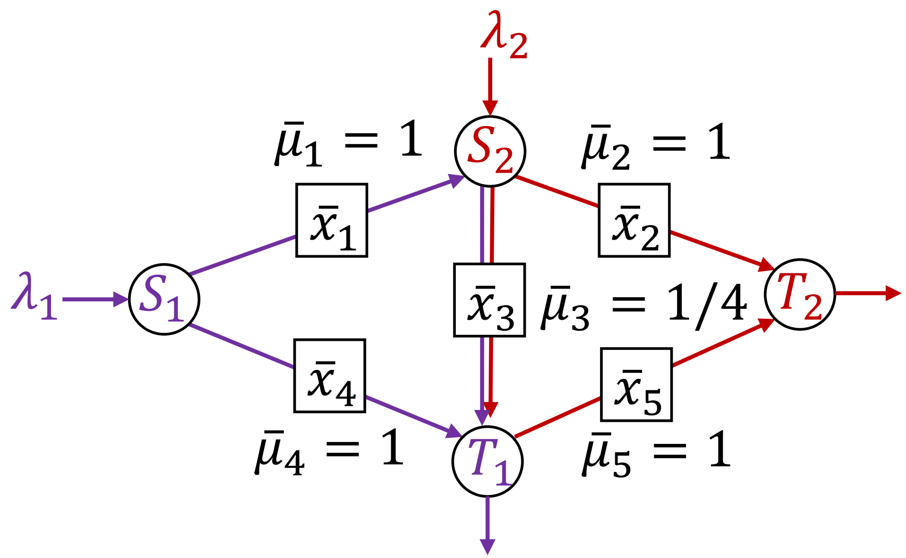
How about more complex networks?

- What if the network is multi-class and not series-parallel?



How about more complex networks?

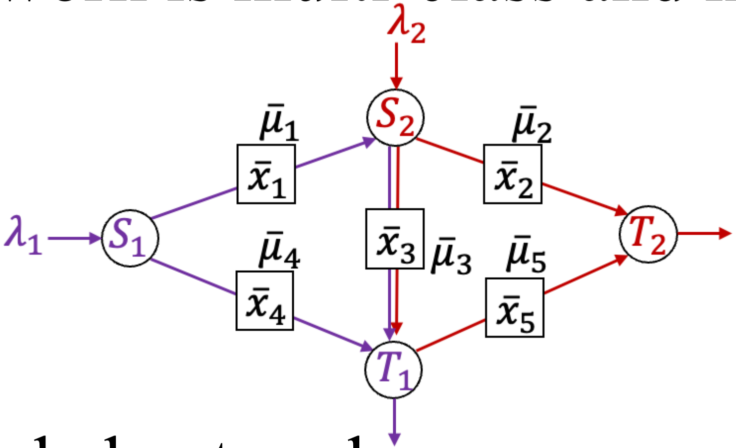
- What if the network is multi-class and not series-parallel?



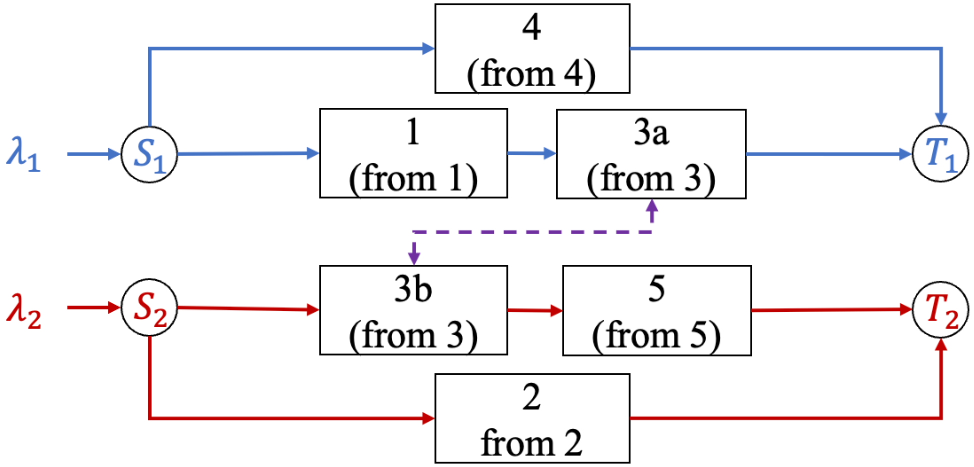
- JSQ is destabilizing
  - Queue at server 3 is unstable
  - Ignorance of downstream congestion
  - As $\bar{x}_3$ gets large, should allocate fewer class-1 jobs to server 1

How about more complex networks?

- What if the network is multi-class and not series-parallel?

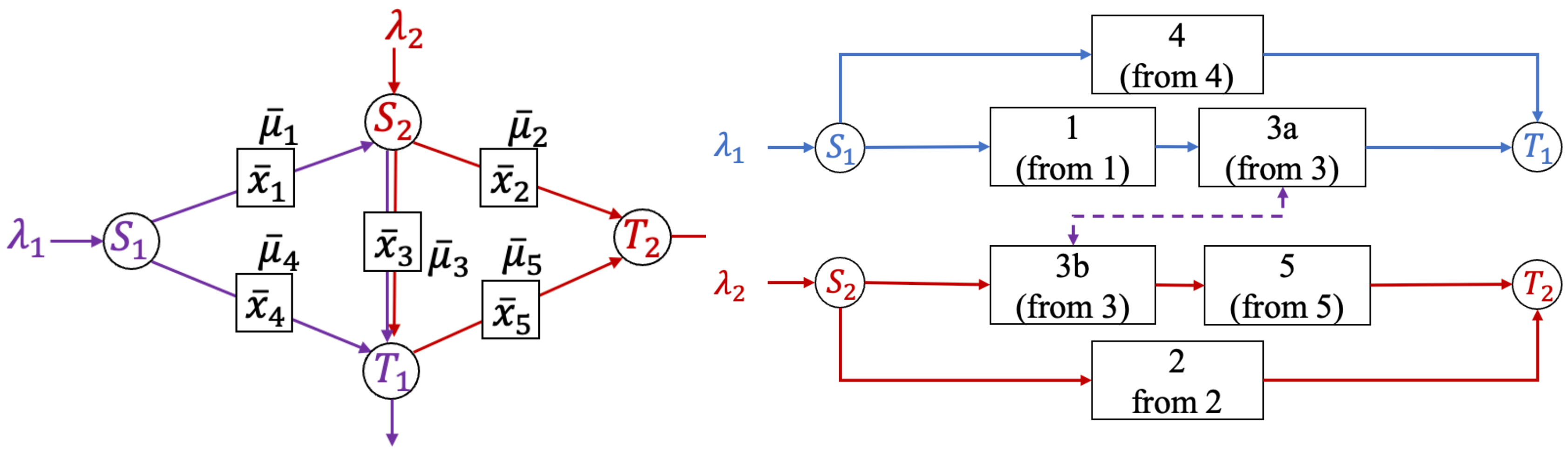


- Consider expanded network



How about more complex networks?

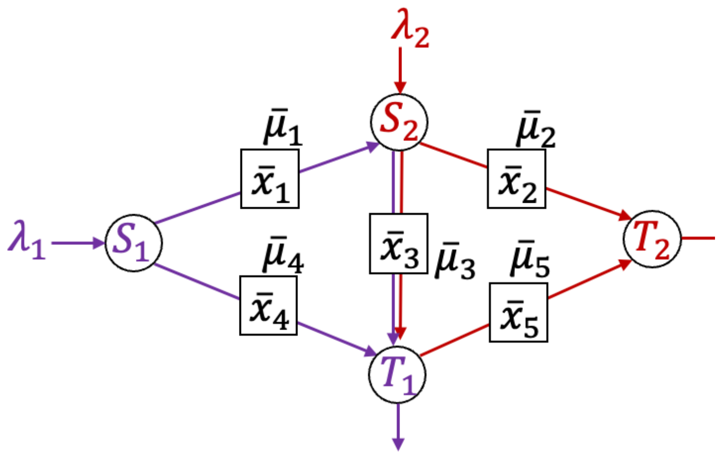
- Consider a multi-class network and its route expansion

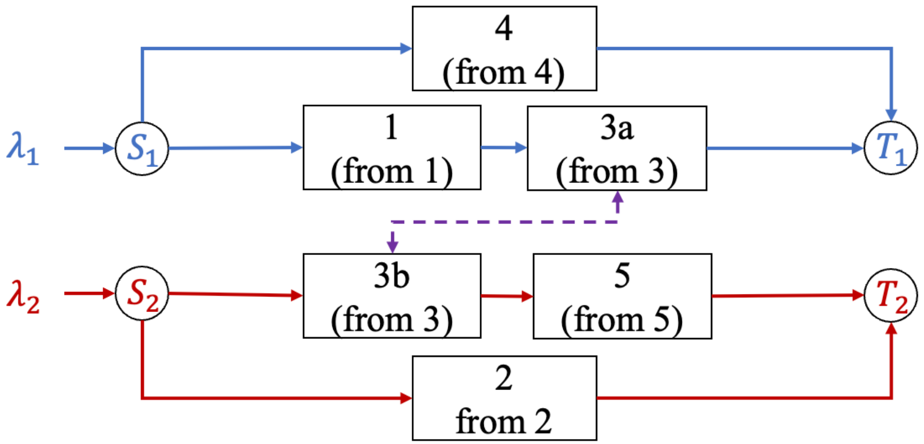


- How to extend the previous route-sum?

JSR for multi-class

- JSR: multi-class centralized control

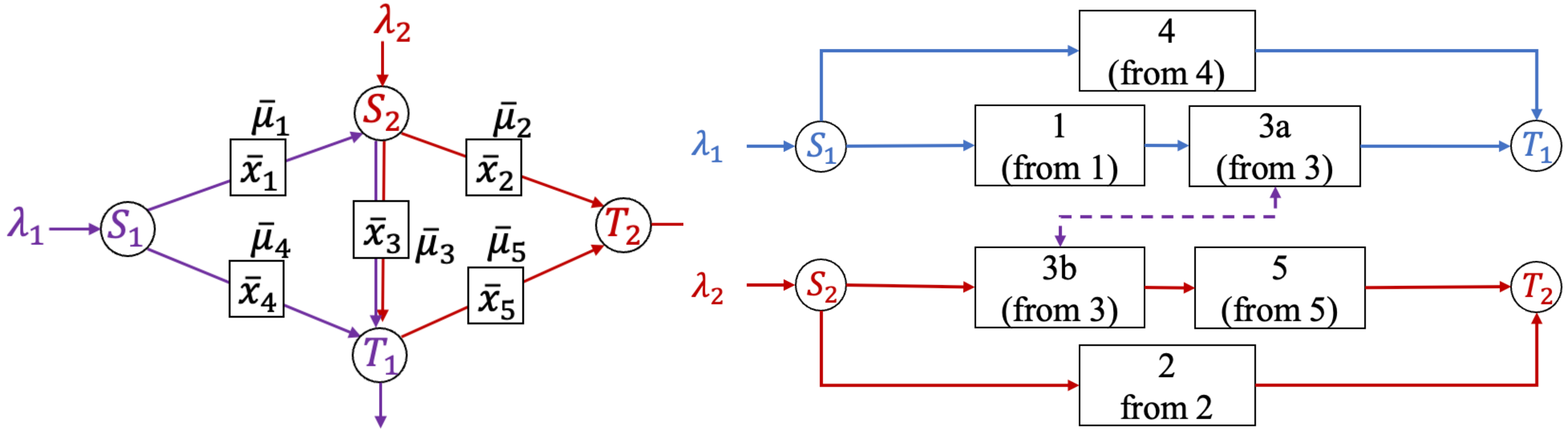




- Simplified JSR
  - A class-1 job arriving at S_1 is routed to server 1 if $x_1 + x_{3a} < x_4$, server 4 if $x_1 + x_{3a} > x_4$, and randomly otherwise
  - A class-2 job arriving at S_2 is routed to server 2 if $x_{3b} + x_5 > x_2$, server 3 if $x_{3b} + x_5 < x_2$, and randomly otherwise
  - The dominant class has a higher priority

JSR for multi-class

- JSR: multi-class centralized control



  - $f_{(1,3)}(x) = \max\{x_1, \alpha(x_1 + x_{3a})\}, f_{(4)}(x) = x_4$
  - A class-1 job joins the "shorter" route between (1,3) and (4)
  - $f_{(3,5)}(x) = \max\{x_{3b}, \alpha(x_{3b} + x_5)\}, f_{(2)}(x) = x_2$
  - A class-2 job joins the "shorter" route between (3,5) and (2)
  - Prioritize dominant class (imaginary service rate control)

Stability of the expanded network

- How can we tell if an expanded network is stable under a specific control policy?

Stability of the expanded network

- How can we tell if an expanded network is stable under a specific control policy?
- Proof based on piecewise-linear test function
  - As long as a piecewise-linear test function can be determined, one can always develop a smooth Lyapunov function to verify the Foster-Lyapunov stability criterion [Down & Meyn, 1997]
  - LP-based construction [Down & Meyn, 1997]
  - Rely on knowledge of model data and solving optimization

Stability of the expanded network

- How can we tell if an expanded network is stable under a specific control policy?
- Proof based on piecewise-linear test function
  - LP-based construction [Down & Meyn, 1997]
  - Rely on knowledge of model data and solving optimization
  - This work: explicit MDI construction

$$V(x) = \max \sum_{c} b_c \left(\max \sum_{k} a_k x_k \right)$$

  - Rely only on network topology (# routes & # classes)
  - Remark: test functions not necessarily to be MDI

Stability of the expanded network

- Mean velocity
  - $v_k(x)$: mean velocity of subserver k at state x
    $v_k(x) = \sum_c \lambda_c \pi^c_{S_c,k}(x) + \mu_{k'}(x) h_{k'}(x) - \mu_k(x) h_k(x)$
  - $\pi^c_{S_c,k}$: class-c routing probability from origin S_c to subserver k
  - μ_k: controlled service rate of subserver k
  - h_k: holding status of subserver k
  - k': the immediate upstream subserver of k
- Mean drift

$$D(x) = \sum_c b_c \sum_k a_k v_k(x)$$

  - Result of infinitesimal generator applied to Lyapunov function
  - The network is stable if the mean drift is negative: $D(x) \leq -\epsilon$

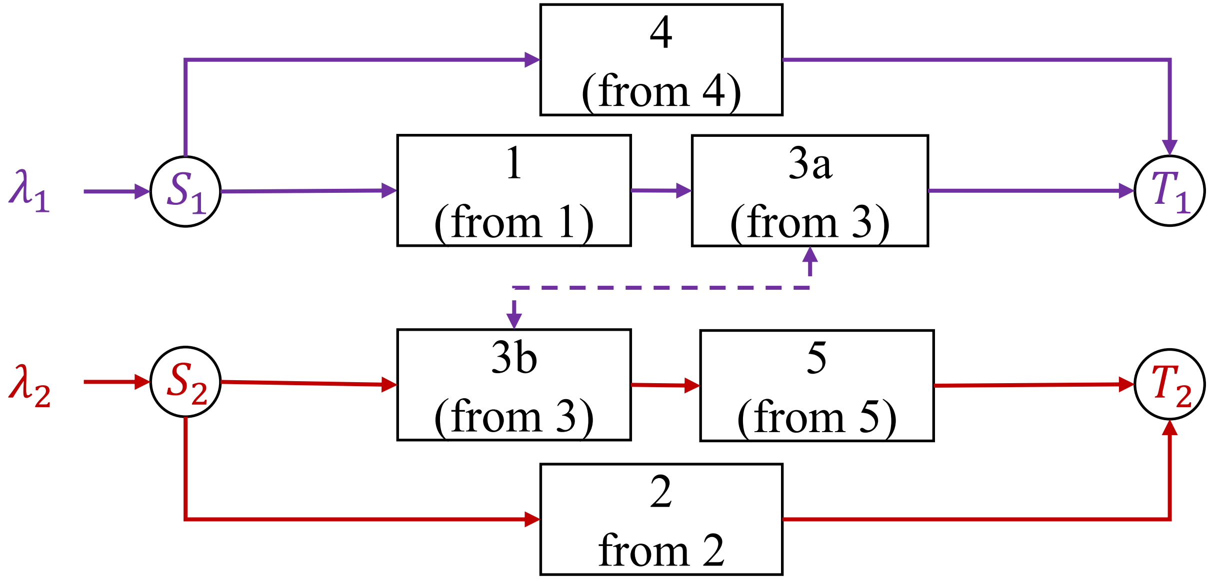
Stability of JSR policy

- Dominance
  - A class/route/subserver is dominant if changes in its traffic state immediately affect the test function V
  - A bottleneck is a dominant subserver while its immediate downstream subserver is not

- High-level idea
  - Identify bottlenecks and upstream subservers
  - Prioritize allocation to non-dominant route and discharge (service) of dominant class customers
  - Negative contribution to the mean drift!

Stability of JSR policy

- Consider the expanded network



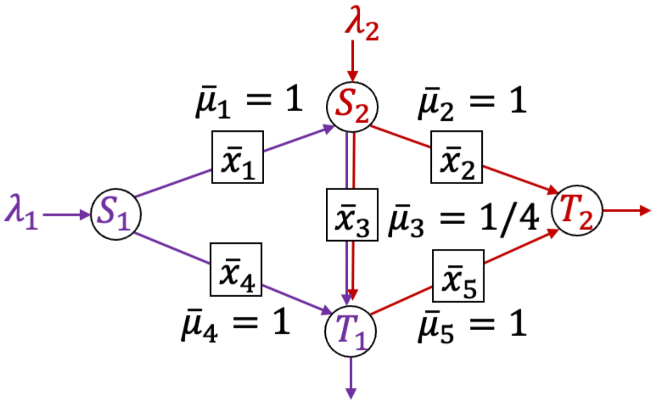
- Test function construction
  - $f_{(1,3)}(x) = \max\{x_1, \alpha(x_1 + x_{3a})\}, f_{(4)}(x) = x_4$
  - $g_1(x) = \max\{f_{(1,3)}(x), f_{(4)}(x), \beta(f_{(1,3)}(x) + f_{(4)}(x))\}$
  - $V(x) = \max\{g_1(x), g_2(x), \gamma(g_1(x) + g_2(x)\}$

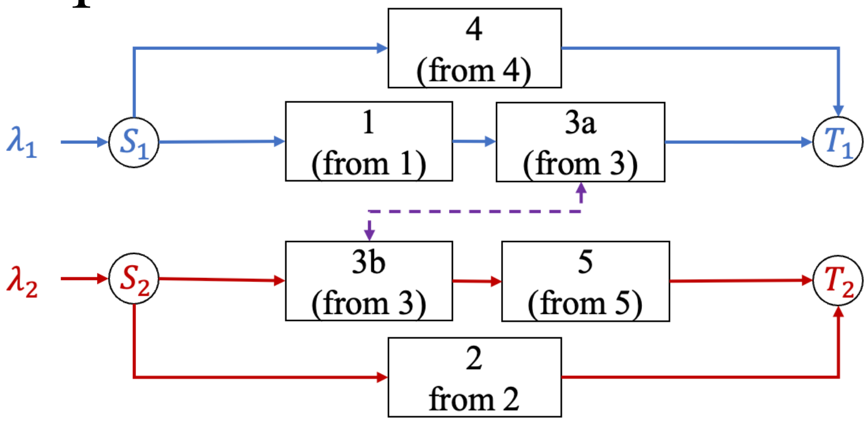
$\alpha = \beta \geq \frac{3}{4}$

$\gamma \geq \frac{1}{2}$

Stability of JSR policy

- Consider a numerical example



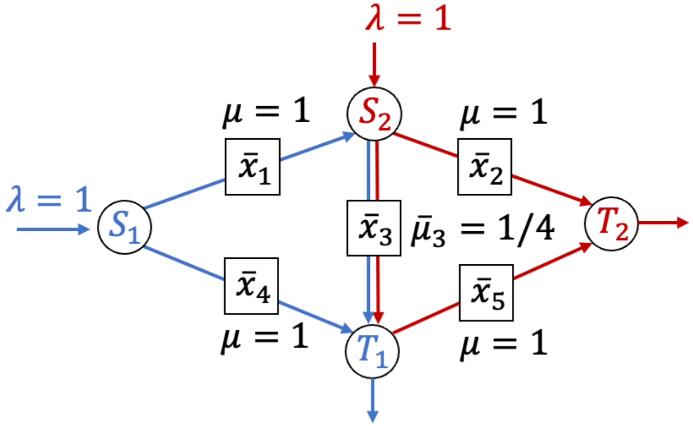


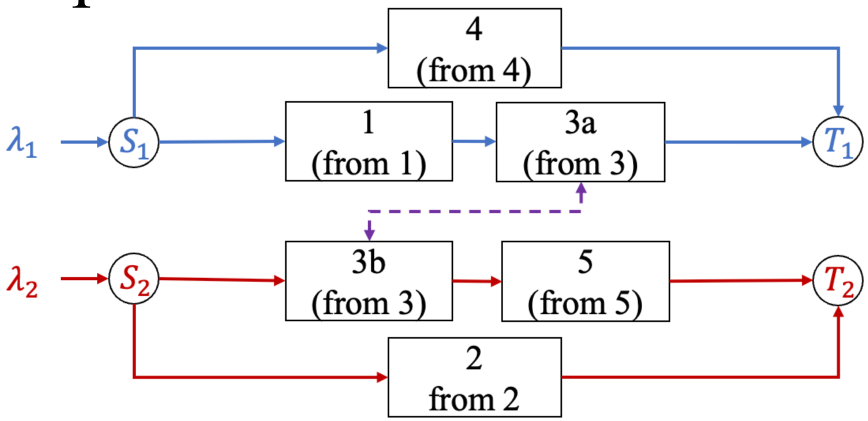
- The network can be stabilized by the JSR policy if and only if

$$\lambda_1 < 1, \qquad \lambda_2 < 1, \qquad \lambda_1 + \lambda_2 < 9/4$$

Stability of JSR policy

- Consider a numerical example





- Consider the following parameters for test functions:

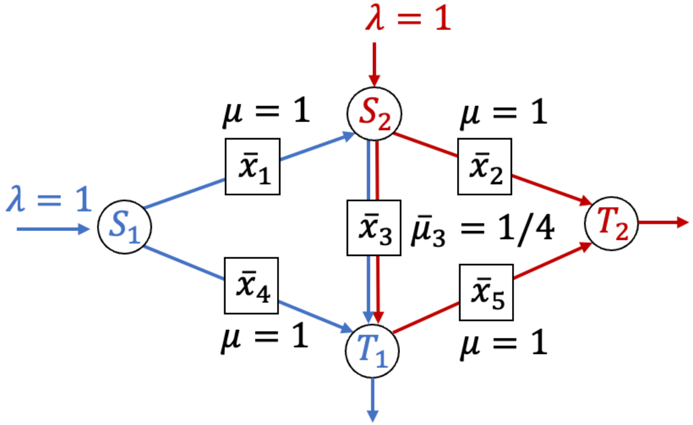
$$\alpha = \beta = \gamma = \frac{3}{4}, \qquad \epsilon = \left(\frac{3}{4}\right)^3$$

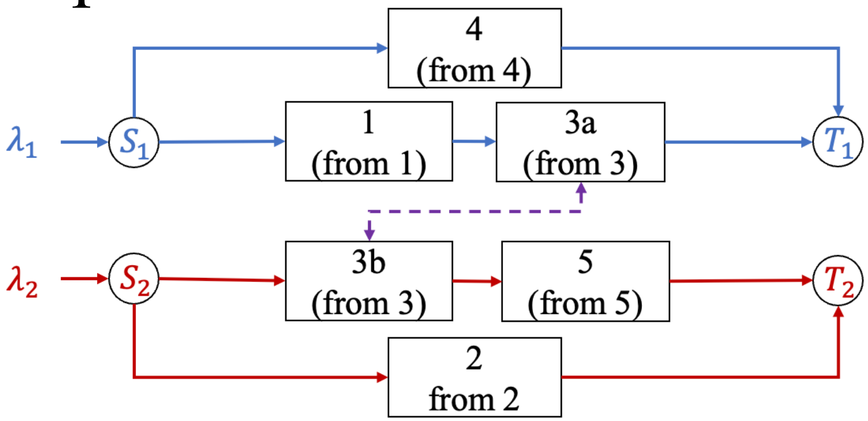
- Case 1: only one route is dominant, incoming job allocated to non-dominant route

$$D(x) \leq -\gamma\beta\alpha\mu = -\left(\frac{3}{4}\right)^3 \leq -\epsilon$$

Stability of JSR policy

- Consider a numerical example





- Consider the following parameters for test functions:

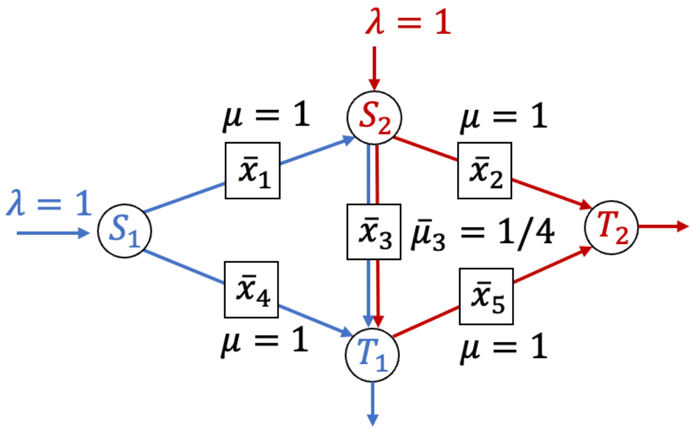
$$\alpha = \beta = \gamma = \frac{3}{4}, \qquad \epsilon = \left(\frac{3}{4}\right)^3$$

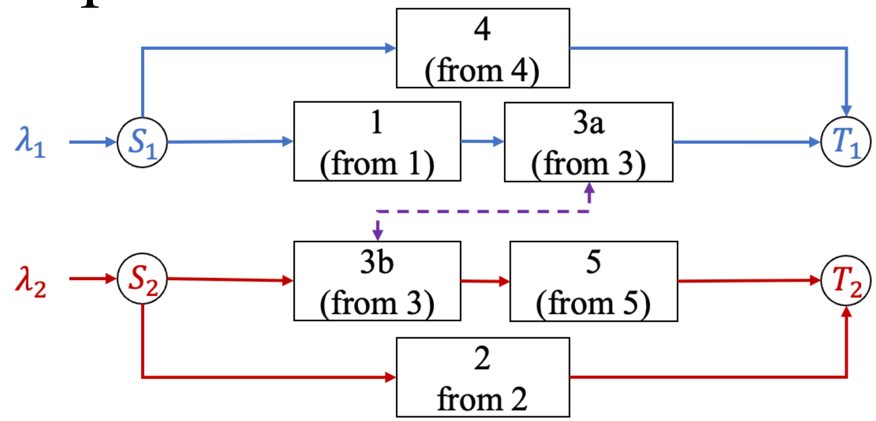
- Case 2: two routes with different OD are dominant, incoming job allocated to non-dominant route

$$D(x) \leq -\gamma\beta\alpha\mu = -\left(\frac{3}{4}\right)^3 \leq -\epsilon$$

Stability of JSR policy

- Consider a numerical example





- Consider the following parameters for test functions:

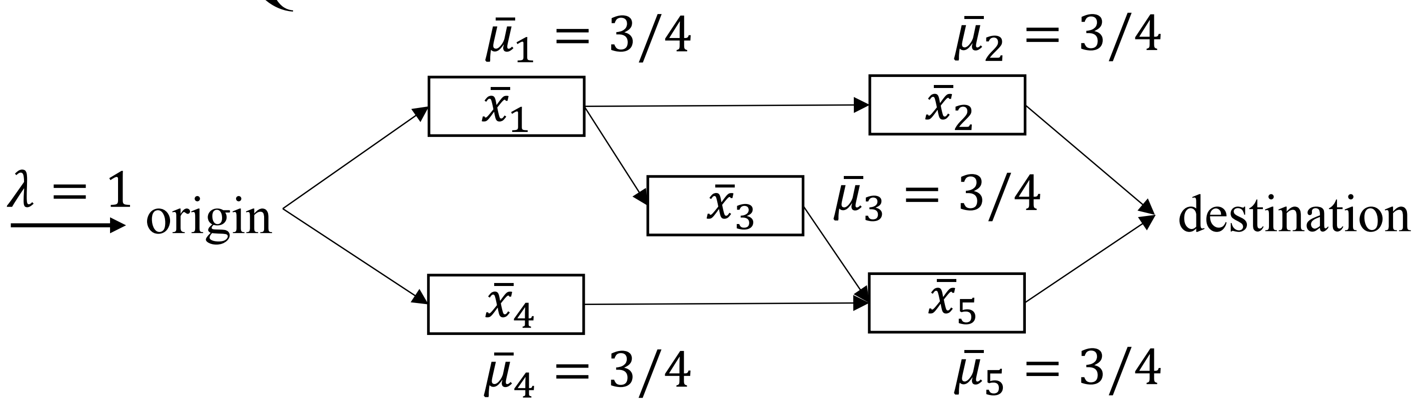
$$\alpha = \beta = \gamma = \frac{3}{4}, \qquad \epsilon = \left(\frac{3}{4}\right)^3$$

- Case 3: two routes with same OD or more than two routes are dominant, incoming job allocated to a random dominant route

$$D(x) \le \gamma\beta(\lambda - \mu - \alpha\mu) = -\left(\frac{3}{4}\right)^3 \le -\epsilon$$

Extend JSQ to decentralized setting

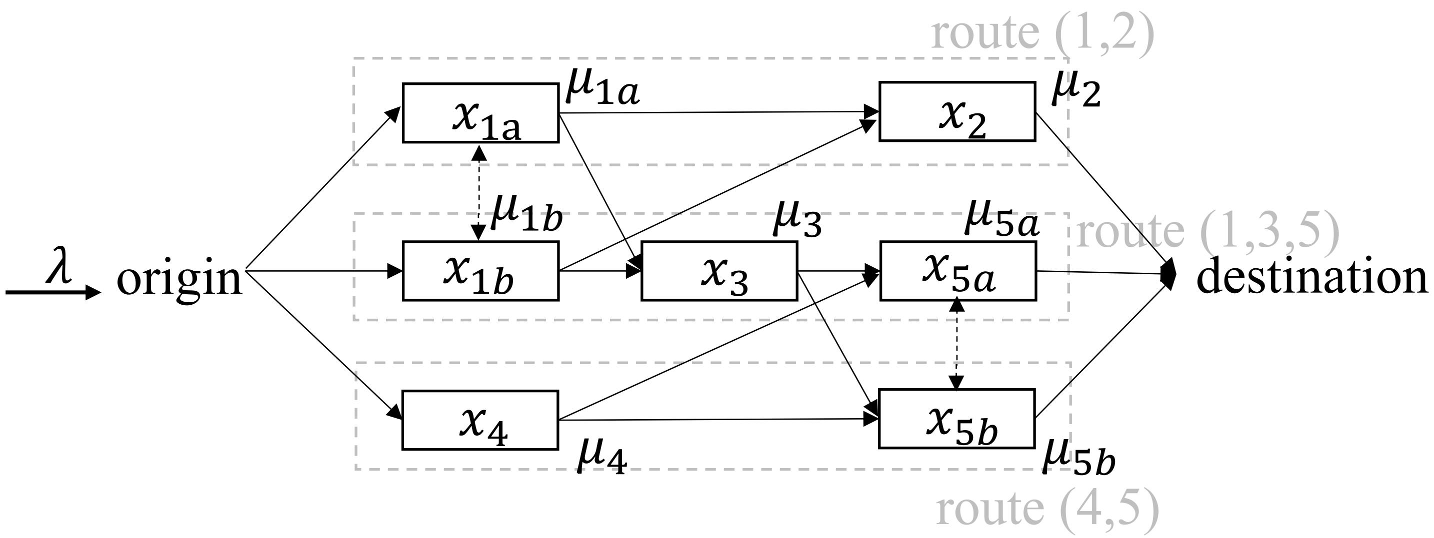
- How about decentralized setting?
  - The decision at each server is based on local traffic info
- Recall: JSQ fails for network



  - Queue at server 5 is unstable
  - Congestion info not propagated to upstream servers

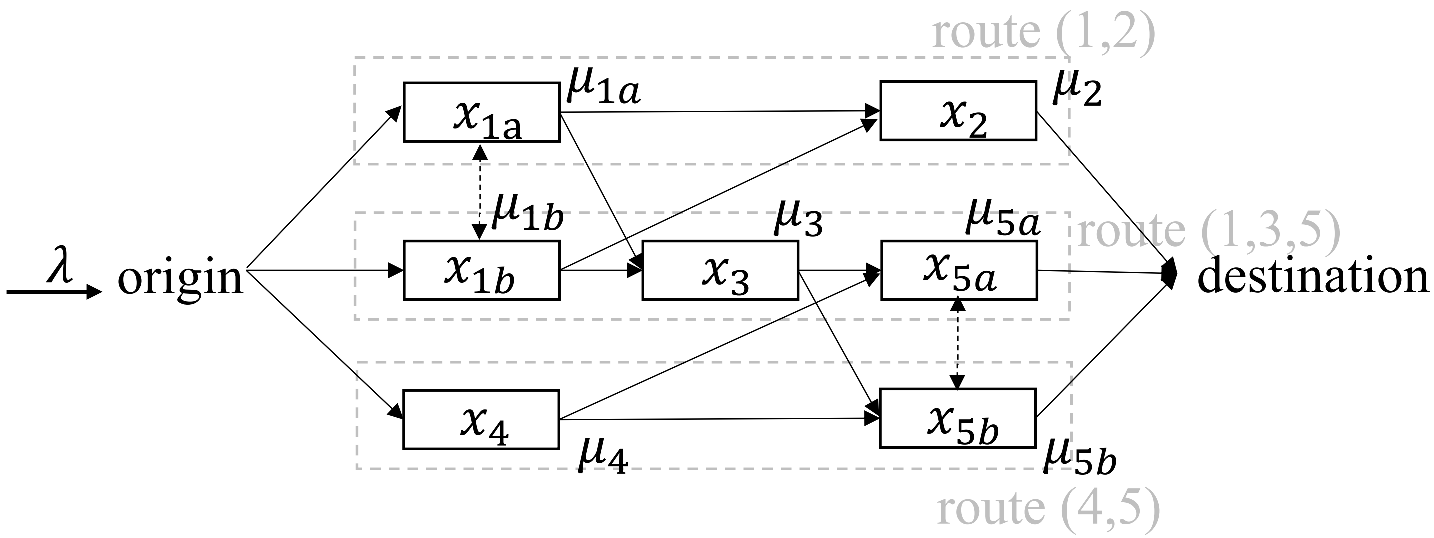
Extend JSQ to decentralized setting

- Recall why JSQ fails
  - Congestion info not propagated to upstream servers



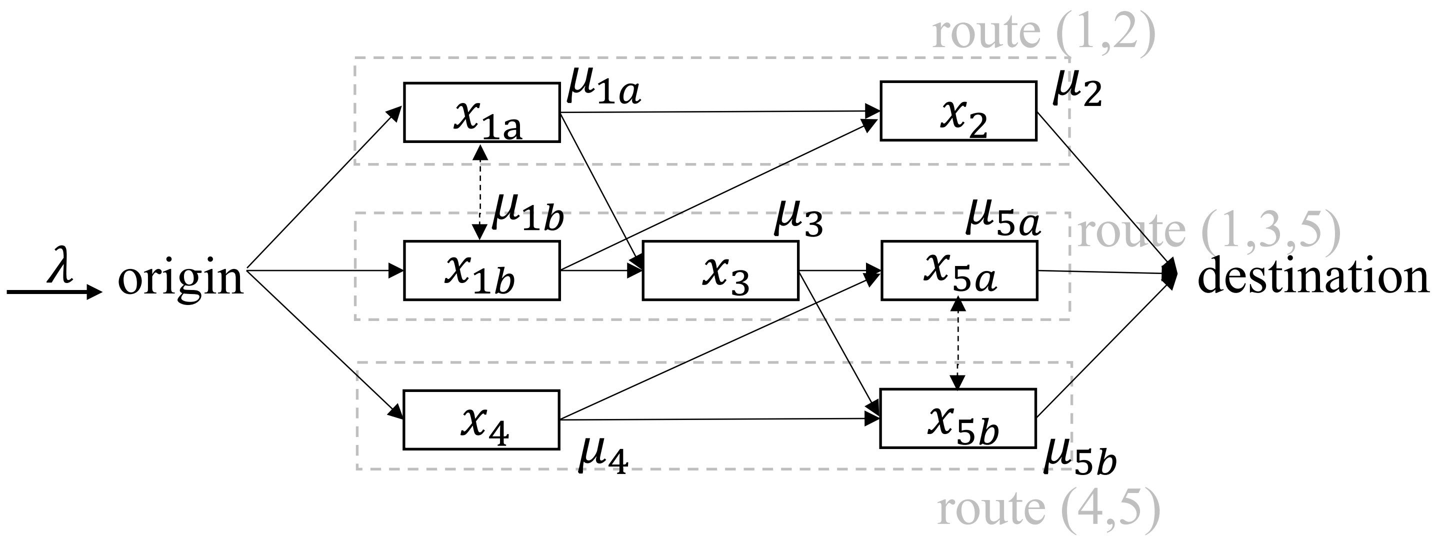
- Solution: artificial holding
  - keep upstream queue size > downstream queue size
  - e.g., subserver 3 is not allowed to discharge if $x_3 \leq x_{5b}$
- JSQ with artificial spillback (JSQ-AS)!

JSQ-AS for single class



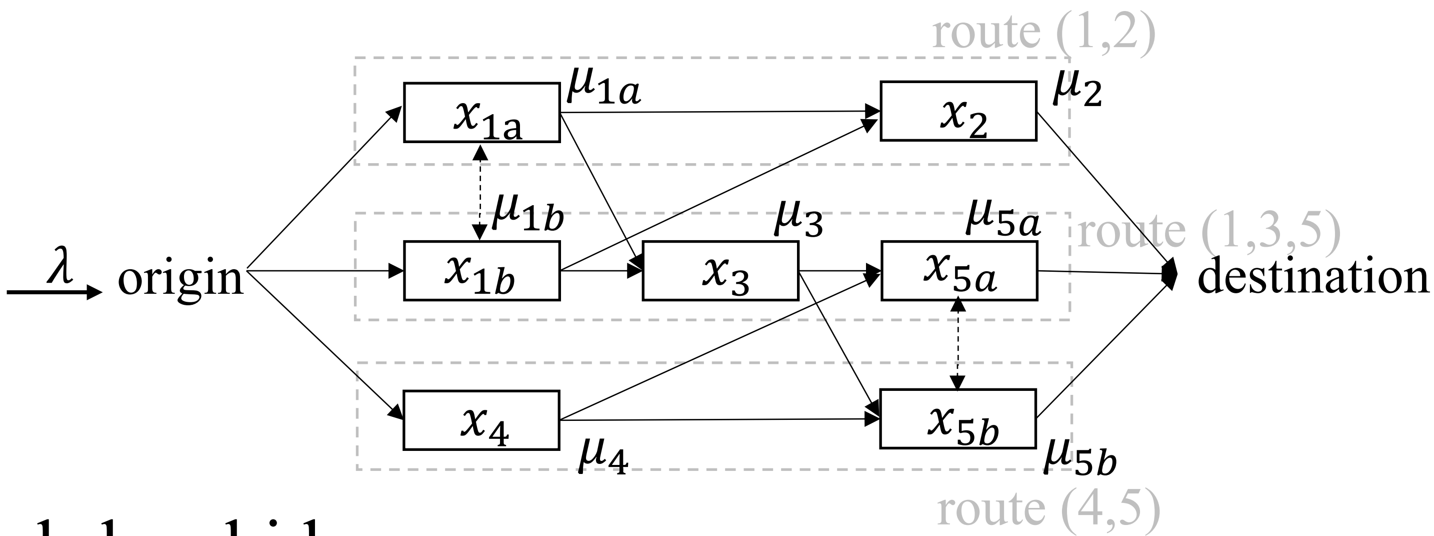
- JSQ-AS: decentralized control for single class
  - Routing: discharge job to shortest downstream queue
  - Holding: a completed job is held if current queue size $\leq$ the immediate downstream queue size
  - Imaginary switch (on expanded network): discharge job to the downstream of dominant subserver

Stability of JSQ-AS



- Consider an alternative test function:
  - $f_1(x) = \max\{x_{1a}, \frac{1+\delta}{2}(x_{1a} + x_2)\}$
  - $f_2(x) = \max\{x_{1b}, \frac{1+\delta}{2}(x_{1b} + x_3), \frac{1+2\delta}{3}(x_{1b} + x_3 + x_{5a})\}$
  - $f_3(x) = \max\{x_4, \frac{1+\delta}{2}(x_4 + x_{5b})\}$
  - $V(x) = \max\{f_1(x), f_2(x), f_3(x)\}$

Stability of JSQ-AS



- High-level idea:
  - Bottlenecks are nonempty and not in the holding status
  - Thus, bottlenecks can discharge customers and contribute negative terms to the drift
  - Either the route with the smallest first subserver queue length is non-dominant or every route is dominant
  - Thus, incoming job is routed to a non-dominant route if exists