The Drift Method
Switch Scheduling and Load Balancing

R. Srikant

c3.ai DTI / ECE / CSL

University of Illinois at Urbana-Champaign

Data Centers and Cloud Computing

Large-scale data processing

hadoop APACHE Spark

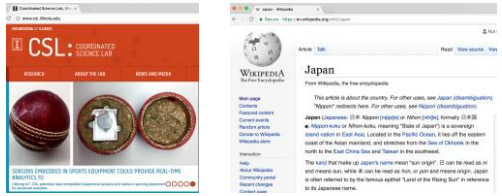
Search engines

Google Bing

Cloud computing

amazon web services™ EC2

Webpages

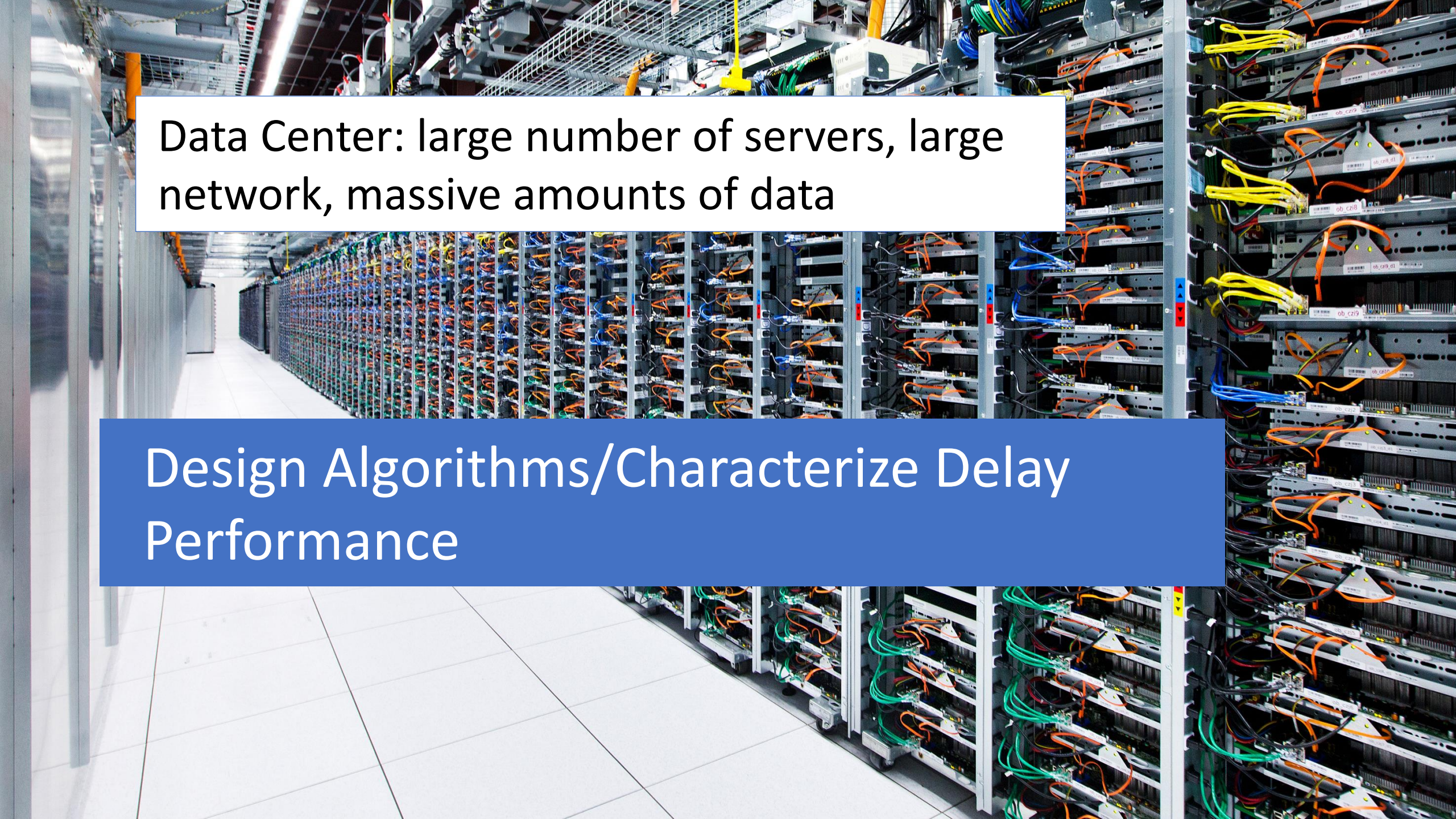


Data analytics

Google Analytics

Cloud storage

Dropbox



Data Center: large number of servers, large network, massive amounts of data

Design Algorithms/Characterize Delay Performance

Outline

- Drift Method: An Introduction
- Scheduling in Switches (heavy traffic $\rho \to 1$)
- Load Balancing in Cloud Computing Systems ($\rho < 1$)
- Conclusions

Part I: Introduction

Stochastic Dynamical Systems

$$W_{k+1} = f_\theta(W_k, noise)$$

- Communication Networks, Queueing Systems: W_k is a vector of queue lengths
  - The "noise" is the randomness in the arrival/departure process
- Reinforcement Learning: W_k is a vector of neural network parameters used to approximate the value function associated with a Markov process.
  - The "noise" is observation of the states which is assumed to evolve according the Markov process
- θ: parameters of a control policy (e.g., a scheduling or a routing policy)

Performance Analysis/Design

$$W_{k+1} = f_\theta(W_k, noise)$$

- Evaluate the performance of a control policy θ in terms of a performance metric
  - E.g.: $E(c(W(\infty))$ or $\frac{1}{T}\sum_{t=0}^{T} c(W(t))$
  - Example: $W = q$, vector of queue lengths, and $c(q) = q_1 + q_2 + \cdots + q_n$
  - In RL, W is the vector of neural network parameters used to approximate the value function V of a Markov process and $c(W)$ is some measure of the accuracy of the approximation
  - In addition to performance analysis, we may also be interested in getting an improved control policy (not in today's talk)

Steady-State Analysis

- Lyapunov drift for moment bounds: non-negative L s.t.

$$E\left[L\big(W(t+1)\big) - L\big(W(t)\big) \middle| F_{t-1}\right] \leq -c\big(W(t)\big) + K$$

Thus,

$$E(c\big(W(t)\big) \leq E(L\big(W(t)\big) - E(L\big(W(t+1)\big) + K$$

- Recall: we are interested in bounding $E(c(W(\infty))$

Steady-State Analysis

- In steady-state

$$E(L\big(W(t)\big) - E(L\big(W(t+1)\big) = 0$$

Then,

$$E(c\big(W(\infty)\big) \leq K$$

- Technicality: one has to show that $E(L\big(W(\infty)\big)$ exists
- How do we choose the Lyapunov function?

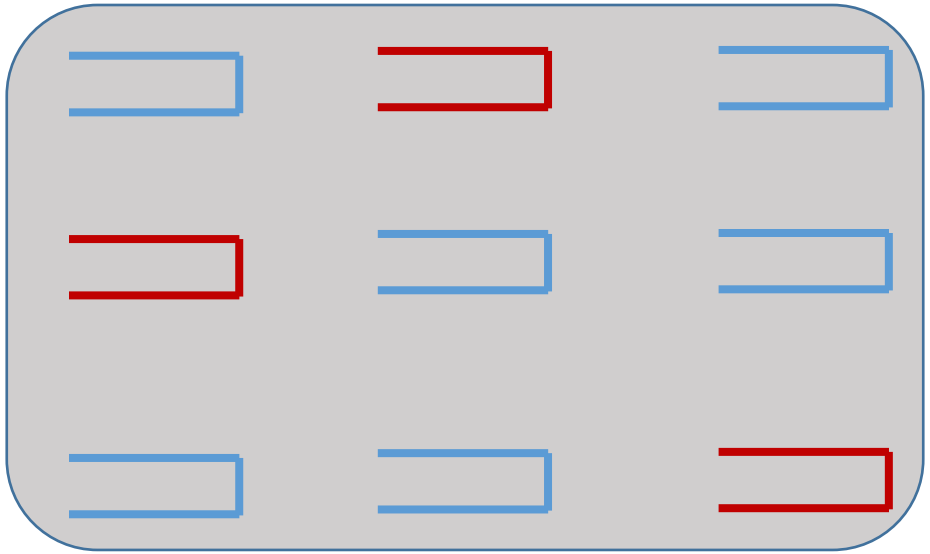
Part II: Switch Scheduling

Collaborator

Siva Theja Maguluri, GaTech

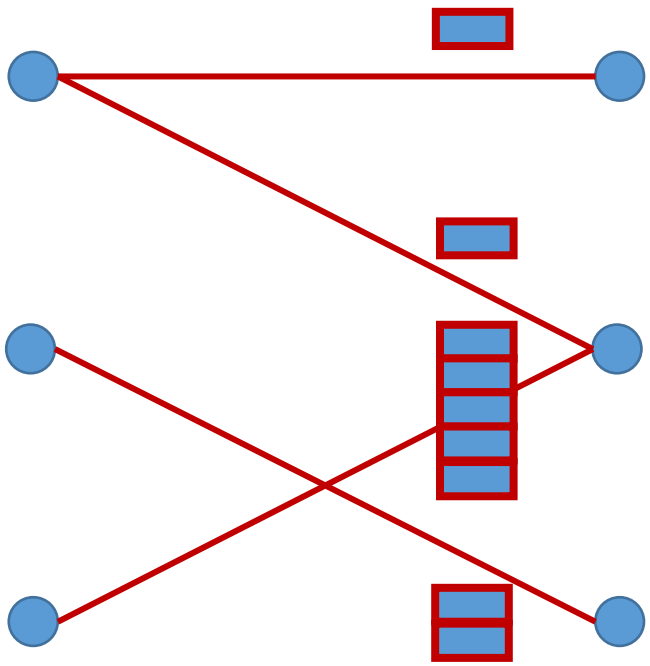
nxn Switch: Abstraction of a Data Center

- The matrix of queues operates in discrete-time
  - Queue (i, j) contains packets generated at server i destined for server j
- Key constraint: In each time slot, one can remove at most one packet from the matrix from each row and at most one packet from each column



Bipartite Graph Interpretation

- Packets arrive to the edges of a bipartite graph
  - Link (i, j) is an edge from $\text{server } i$ to $\text{server } j$

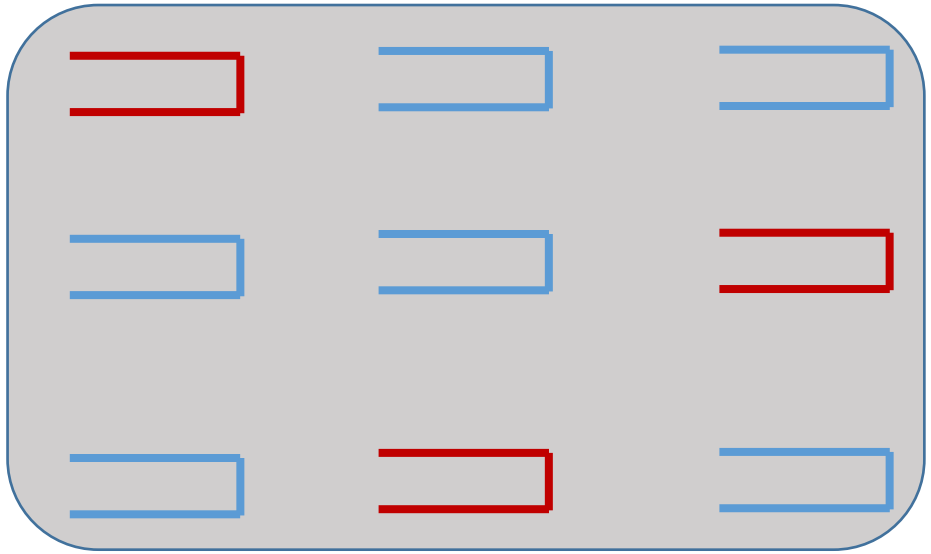


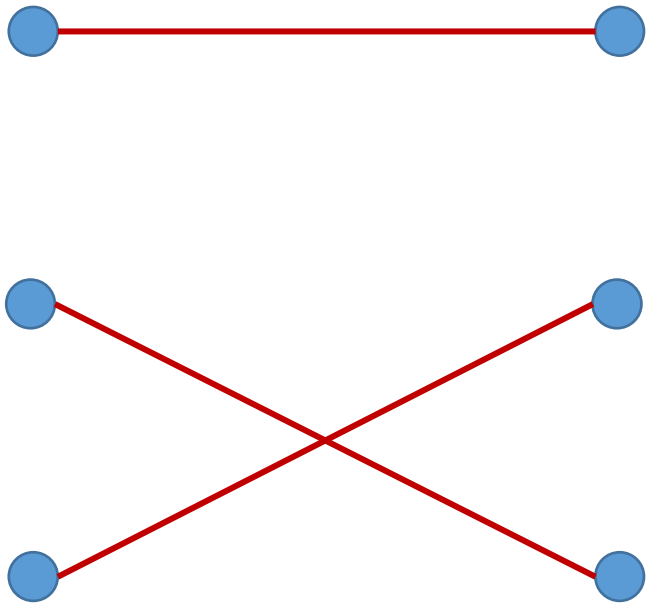
- Only edges with non-zero backlogs are shown here

- New arrivals add to the backlog/create a new edge

Choose a Matching

- Remove one packet from each edge in the matching





Stability Condition

- λ_{ij}: Arrival rate of packets to matrix element (i,j). Necessary condition for stability:

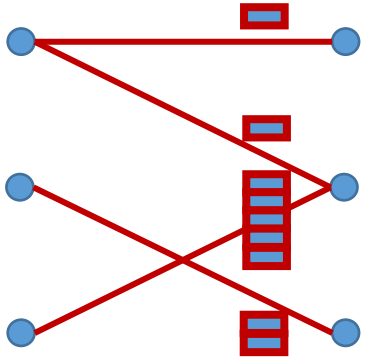
$$\sum_i \lambda_{ij} < 1, \qquad \sum_j \lambda_{ij} < 1.$$

- The necessary condition is also sufficient by the Birkhoff-von Neumann theorem (the matrix Λ is column and row sub-stochastic)

- Define $\rho = \max\{ \sum_i \lambda_{ij} , \sum_j \lambda_{ij} \}$ to be the load on the switch

Open Conjecture I

- Is there a scheduling algorithm (i.e., a matching in each time slot) such that the expected packet delay in the network is $O(1)$?



- Why is the question interesting?
  - Practical significance: network size-independent delay
  - There exists a known lower bound which is $O(1)$
  - Conjecture: Yes

(Previously Open) Conjecture II

- In the limit as $\rho \to 1$ (the load increases), there exists a scheduling algorithm such that

$$\frac{E(Delay)}{\text{Lower Bound}} \to O(1)$$

- Remarks:
  - Computing E(Delay) is equivalent to computing the expectation of the sum of the components of a Brownian motion constrained to live in a cone in R^{n^2}.
  - The steady-state distribution of which is unknown!

Theorem

- In the limit $\rho \to 1$, using MaxWeight Matching,

$$\frac{E(Delay)}{\text{Lower Bound}} \to \left(2 - \frac{1}{n}\right)$$

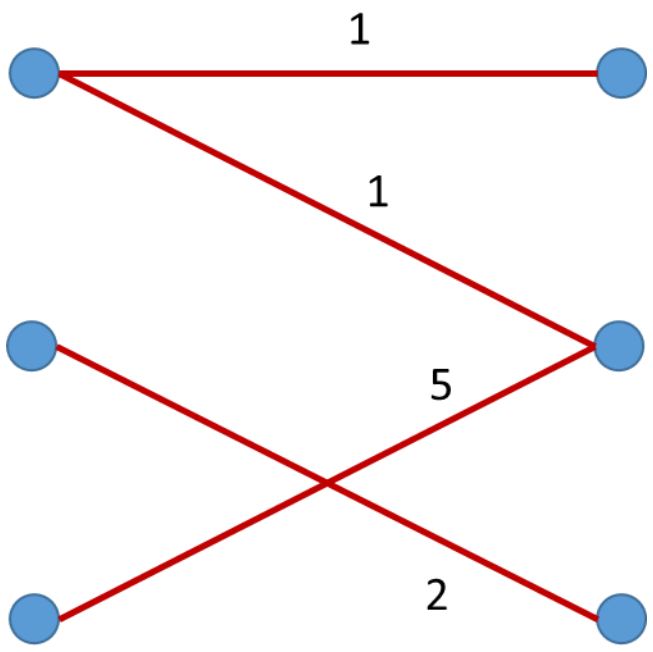
- Remark: in terms of the (reflected) Brownian motion,

$$E\left(\sum_{ij} X_{ij}\right) = O(n),$$

even though there are n^2 components in the state of the Brownian motion

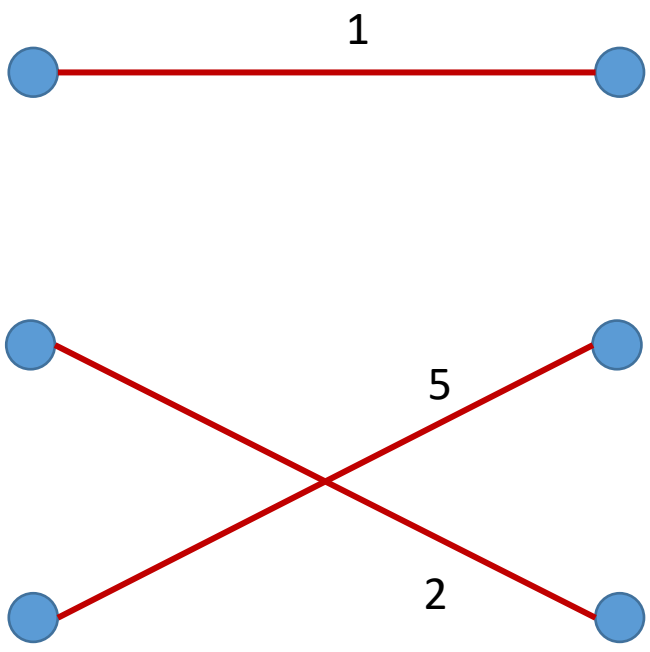
MaxWeight Matching

- Associate a weight with each edge: the backlog of packets at that link
- Find a matching with the largest weight
- Remove one packet from each edge in the matching



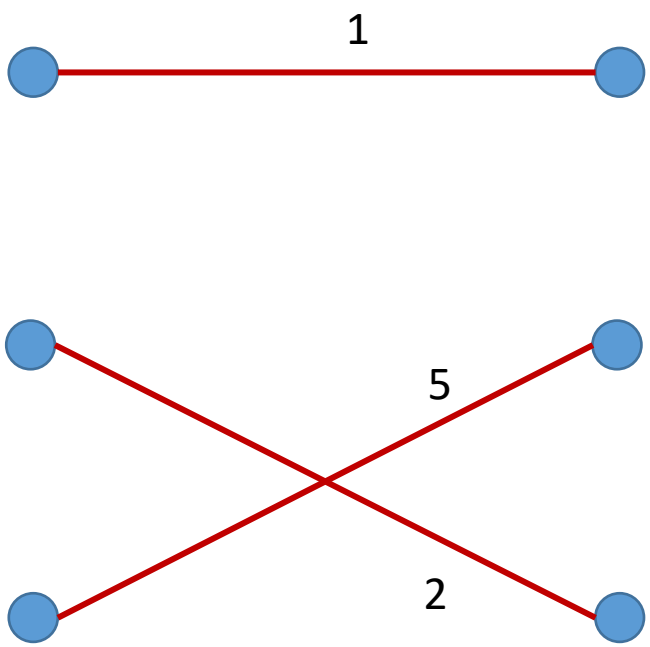
MaxWeight Matching

- Associate a weight with each edge: the backlog of packets at that link
- Find a matching with the largest weight
- Remove one packet from each edge in the matching



Back to the Switch Problem

- One packet from each link in the MaxWeight matching is removed
- In the meantime, arrivals to other links increase their weight
- At some point, there will be at least two matchings with the largest weight
- Important Insight: Eventually all (maximal) matchings will have roughly equal weight



State-Space Collapse

- Simplest 2x2 case: All matchings have equal weights implies state collapses from 4 dimensions to 3 dimensions

- Important Related Work: Andrews-Jung-Stolyar (2007), Shah-Wischik (2012)

State Space Collapse for Switch

$$q = \begin{array}{c} \\ w_1 \\ \vdots \\ w_i \\ \vdots \\ w_n \end{array} \begin{array}{c} \begin{array}{ccccc} \widetilde{w}_1 & \cdots & \widetilde{w}_j & \cdots & \widetilde{w}_n \end{array} \\ \begin{bmatrix} w_1 + \widetilde{w}_1 & & w_1 + \widetilde{w}_j & & w_1 + \widetilde{w}_n \\ & & \vdots & & \\ w_i + \widetilde{w}_1 & \cdots & w_i + \widetilde{w}_j & \cdots & w_i + \widetilde{w}_n \\ & & \vdots & & \\ w_n + \widetilde{w}_1 & & w_n + \widetilde{w}_j & & w_n + \widetilde{w}_n \end{bmatrix} \end{array}$$

- The $2n$ variables w_i and $\widetilde{w_j}$ characterize the state. The algorithm pushes the state to lie in a $(2n-1)$-dimensional space, instead of R^{n^2}
- Good policies seem to force a dramatic state-space collapse to achieve good performance in high-dimensional problems

Drift of a Lyapunov Function

- Let $q_{\|}$ denote the projection of the state vector in the lower-dimensional space. Define $q_{\perp} = q - q_{\|}$

Key Lemma:

$$E\left(\|q_{\perp}(t+1)\| - \|q_{\perp}(t)\| \;\middle|\; q(t)\right) < 0 \text{ for “large” } \|q_{\perp}(t)\|$$

- Using Hajek (1982) or Bertsimas, Gamarnik, Tsitsiklis (2001), one can then show that $E(\|q_{\perp}\|) \ll E(\|q\|)$ in steady-state, i.e., state-space collapse

Completing the Proof

- In steady-state: $E\left(\left\|q_{\|}(t+1)\right\|^2\right) = E\left(\left\|q_{\|}(t)\right\|^2\right)$
  - $\left\|q_{\|}\right\|^2 = \sum_i \left(\sum_j q_{ij}\right)^2 + \sum_j \left(\sum_i q_{ij}\right)^2 - \frac{1}{n}\left(\sum_{ij} q_{ij}\right)^2$

- A long sequence of manipulations yields the result:

$$\lim_{\rho \to 1} (1-\rho) E\left(\sum_{ij} q_{ij}\right) = O(n)$$

- From here, it is straightforward to show the main result on the ratio of the delay to the lower bound is $O(1)$

Comparing with the Traditional Approach

- In steady-state: $E(\|q(t+1)\|^2) = E(\|q(t)\|^2)$

- Yields:

$$\lim_{\rho \to 1} (1-\rho) E\left(\sum_{ij} q_{ij}\right) = O(n^2)$$

- Throwing away the "unnecessary" component of q gives us a much better result

Part III: Load Balancing

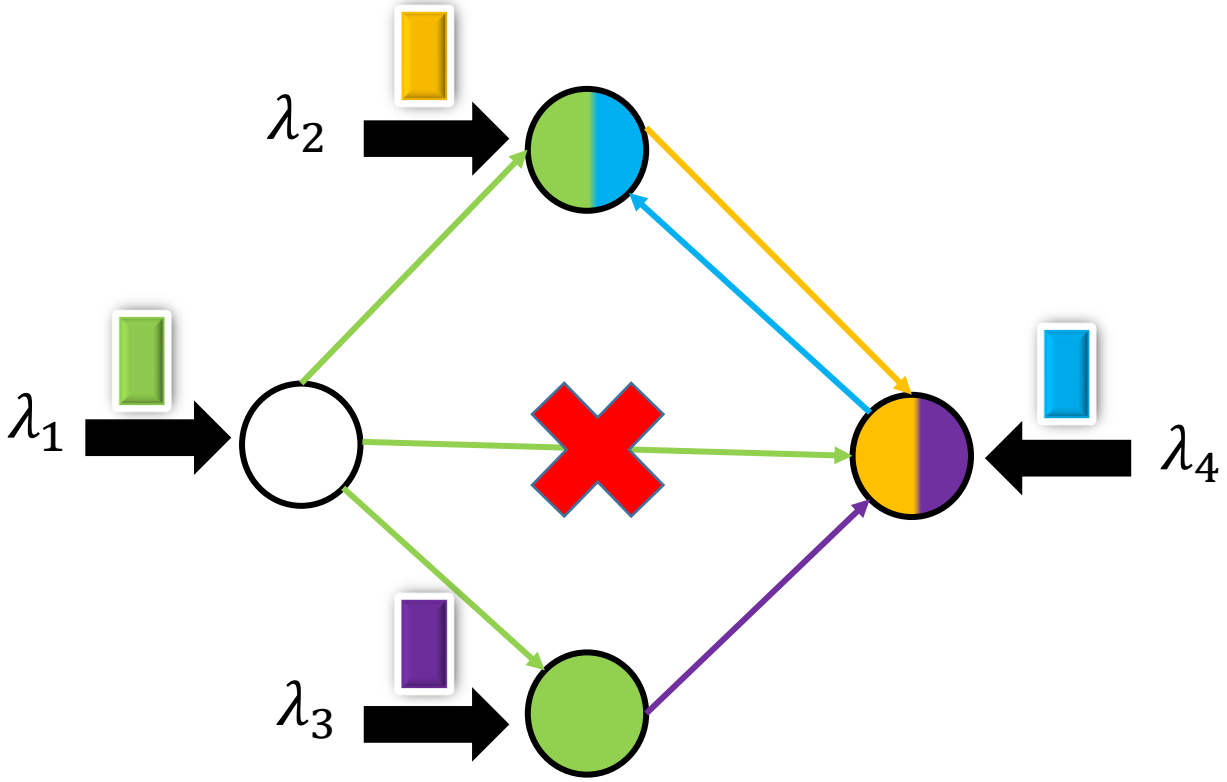
Coauthors

Wentao Weng

Xingyu Zhou

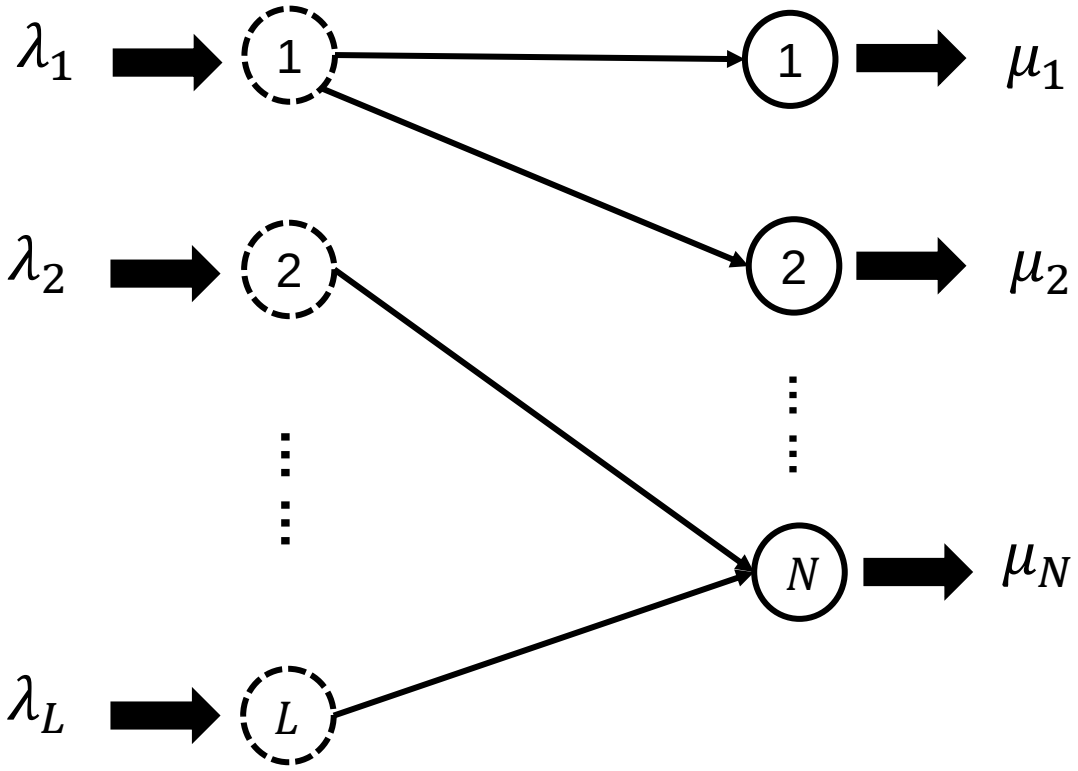
Load Balancing with Locality

- A job to a server can only be served by servers with the data it needs. **<u>Goal: optimal response time</u>**



- Green job can only be served at the servers with green data
- Load balancing does not involve all servers for all jobs
- Other models: Wang, Zhu, Ying, Tan and Zhang (2014), Xie and Lu (2015), Xie, Yekkehkhany, Lu (2016),…
- Important related work: Mukherjee, Borst, Leeuwarden (2018), Cruise, Jonckheere, Shneer (2020)

More than Stability…



- Poisson arrivals, exponential service times
- Is JSQ (join-the-shortest-queue) optimal in this model?
  - In the sense of minimizing mean response time
- What's the mean response time when the system size does not scale to infinity?
  - Bounds on the deviation from the mean-field limit for finite-sized systems

Optimality

- For a **well-connected** system,

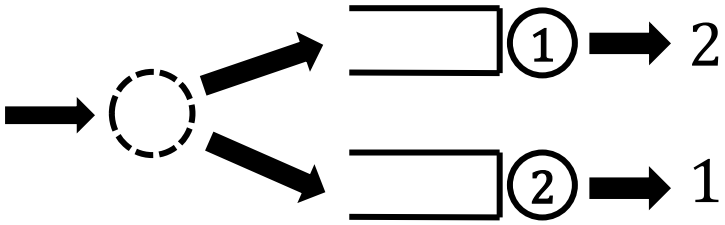
Never wait in a queue

JSQ has asymptotically zero delays for homogeneous servers

Join-the-Fastest-of-the-Shortest-Queues

Stronger than zero delays

JFSQ is asymptotically optimal for heterogeneous servers



JFSQ: In case of a tie in the shortest queue lengths, join the top queue

Optimality

- For a **well-connected** system,

Contemporaneous work: Rutten and Mukherjee (2020)

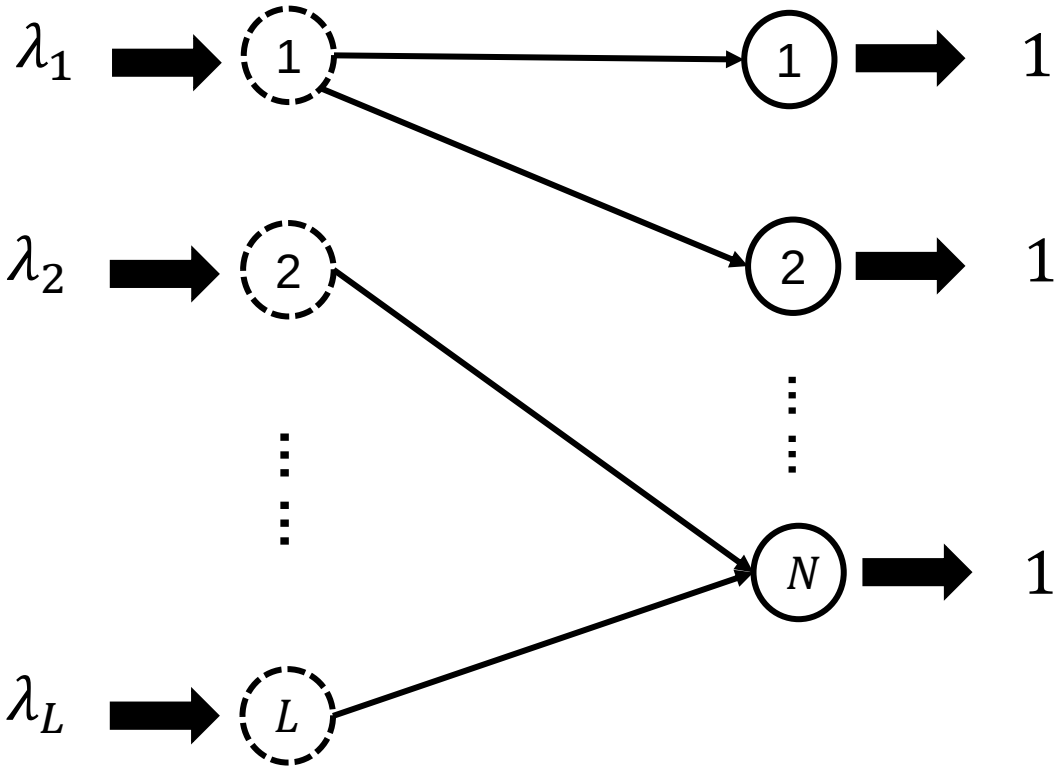
JSQ has asymptotically zero delays for homogeneous servers

JFSQ is asymptotically optimal for heterogeneous servers

A delay bound for finite systems

This talk

Model assumptions (simpler case)



- Finite buffer size: b
- Traffic:

$$\sum \lambda_i = N\lambda, \lambda \in [0,1)$$

  - N can scale, λ is fixed.
- We are interested in the large-server limit and in performance guarantees for large, but finite-sized systems

Notation

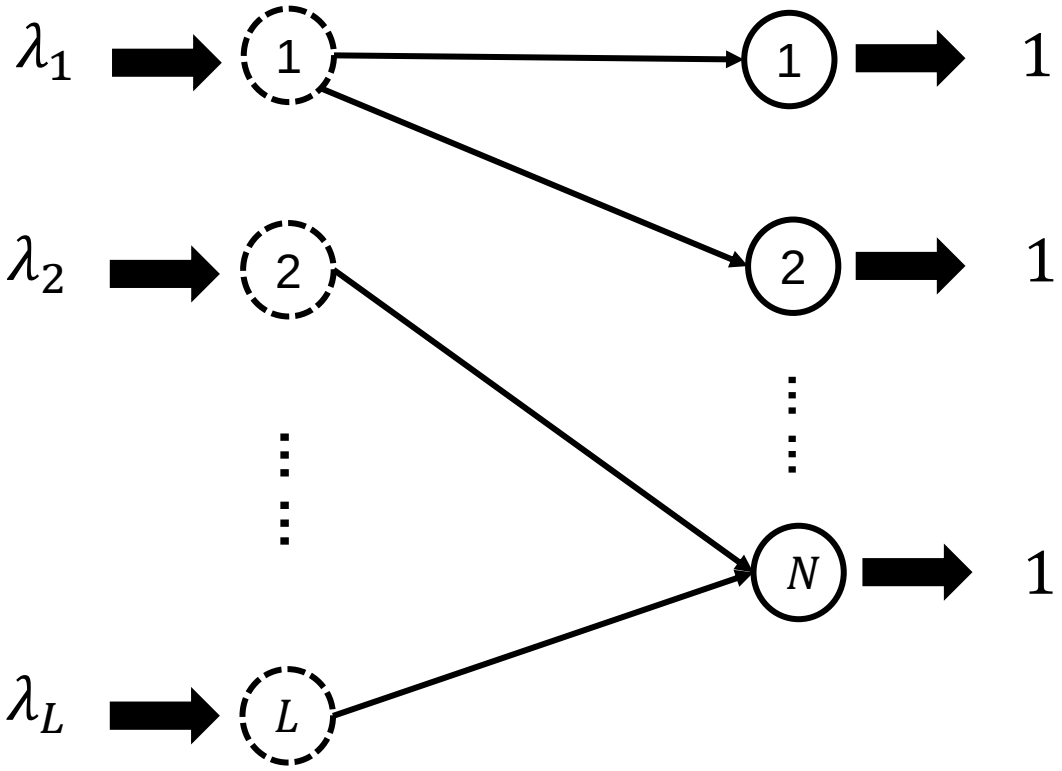
- N: number of servers, Finite buffer size: b

- S_i is the fraction of servers with at least i backlogged jobs.
  - Thus, $NS_i = N\lambda$

- Note that $N \sum_{i \geq 1} S_i$ is the total number of jobs in the system

- In an "ideal" system, we would like $\sum_{i \geq 1} S_i \approx \lambda$

Main result

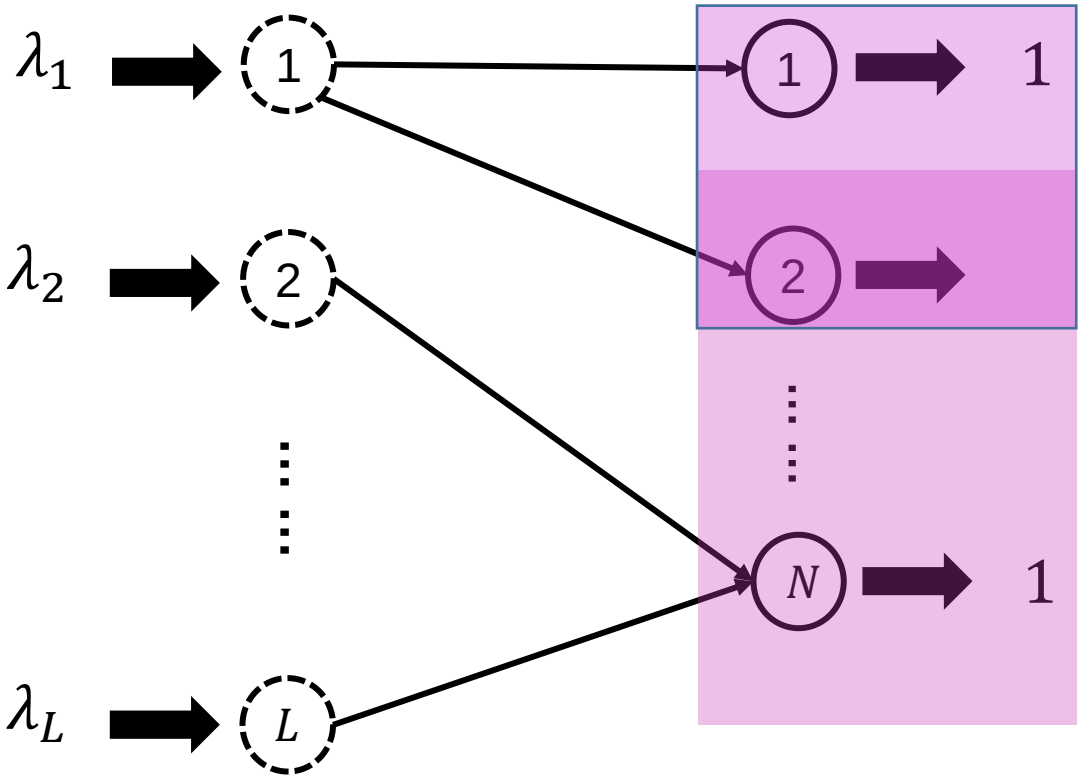


"**Well-Connected Graph**" assumption:
For any subset A of servers such that
$$|A| \geq \frac{1-\lambda}{2} N,$$
the sum of arrival rates of ports not connected with A is bounded by $\tilde{d}N$.

For any $0 < \epsilon \leq \frac{1-\lambda}{4}$, with $\tilde{d} \leq \frac{\epsilon}{2b}$,
under JSQ,
$$E\left[\max\left(\sum_{i\geq 1} S_i - (\lambda+\epsilon), 0\right)\right] = O\left(\frac{b^2}{\epsilon N}\right),$$
and the blocking probability is at most
$$\frac{\tilde{d}}{\lambda} + O\left(\frac{b^2}{\epsilon N}\right).$$

$S_i(t)$ is the fraction of servers with queue length at least i

Interpretation



- Assumption: Any large subset of servers must be connected to a large number of sources
- Is JSQ nearly optimal for our model?
  - Yes! Asymptotically goes to 1
- What is the blocking probability?
  - Asymptotically goes to zero
- For finite-sized systems, close to the above asymptotic values
  - Depending on the buffer size, the connectivity of the graph, and the size of the system

Proof: The Goal is to Match the Mean-Field

- $S_i = \frac{1}{N} * |$#of servers with queue length$\geq i|$

- For JSQ in classical load balancing, when $N \to \infty$,

$$S_1 \to \lambda, S_i \to 0, \forall i \geq 1$$

- Want to show similar result for bipartite load balancing

Drift method for the Large-Server Limit

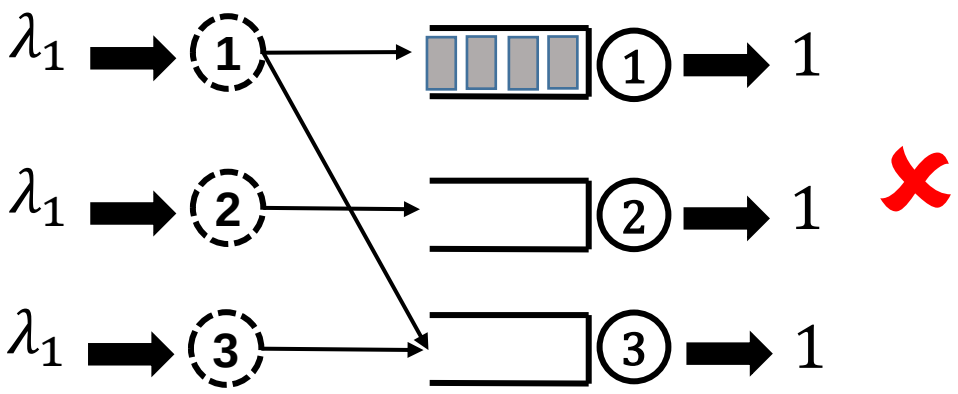
- Metric of Interest: $E\left[\left(\sum_{i=1}^{b} S_i - (\lambda + \epsilon)\right)^+\right]$

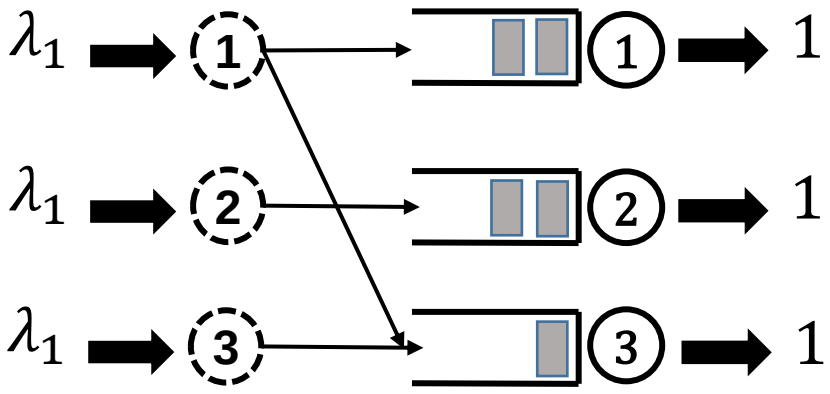
- Use the Lyapunov function $L(S) = \frac{1}{2}\max\left(\sum_{i=1}^{b} S_i - (\lambda + \epsilon), 0\right)^2$

Key Difficulty in the Drift Analysis

- We need to show that the system does not idle when there are enough jobs in the system

Resource Pooling



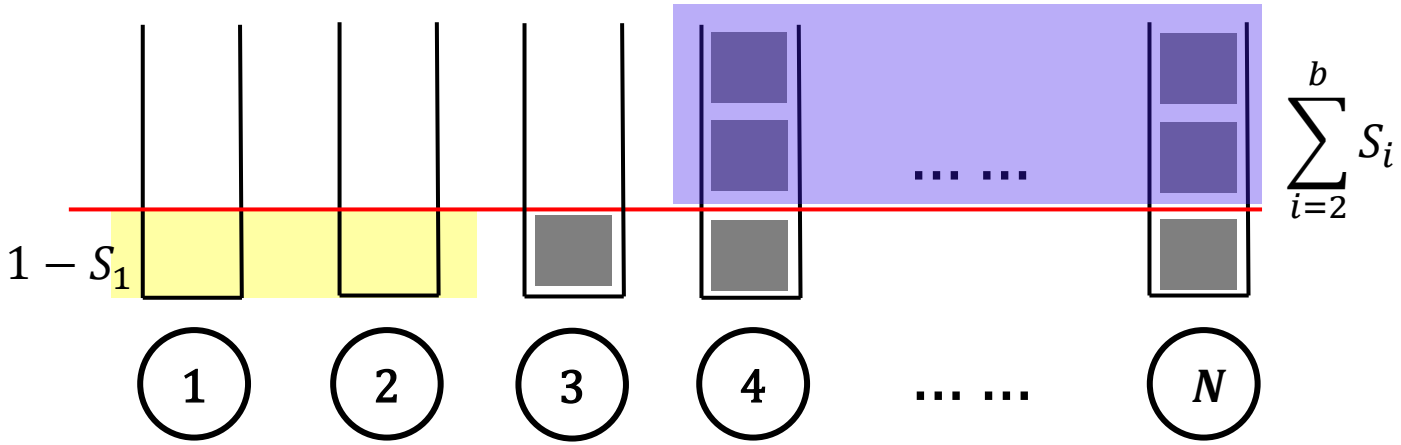


Use a Second Lyapunov function

- Consider the Lyapunov function (Liu and Ying, 2020)

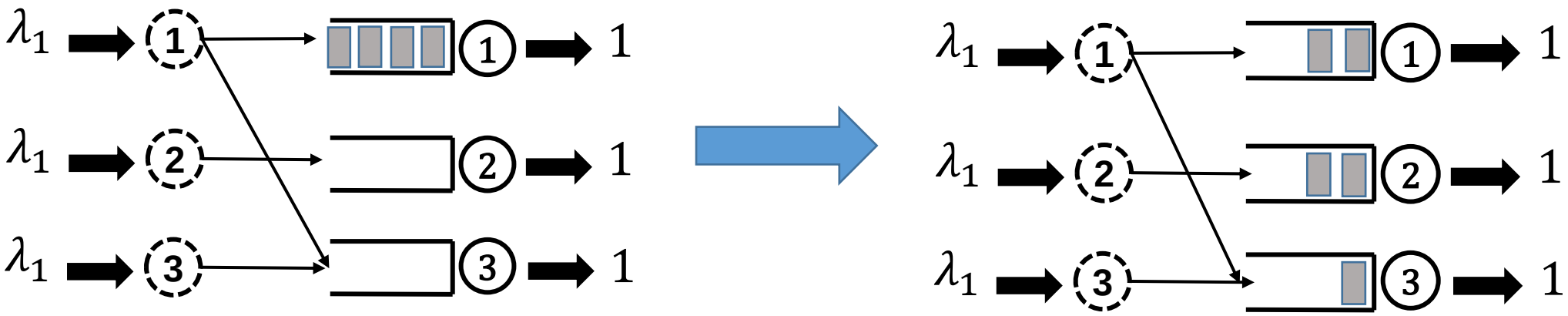
$$V(s) = \min\left(\sum_{i=2}^{b} s_i, \left(\lambda + \frac{1}{2}\epsilon + 4\delta - s_1\right)^+\right)$$

- If $V(s)$ is small: either purple region or yellow region is small.



Large V Implies Many Idle Servers

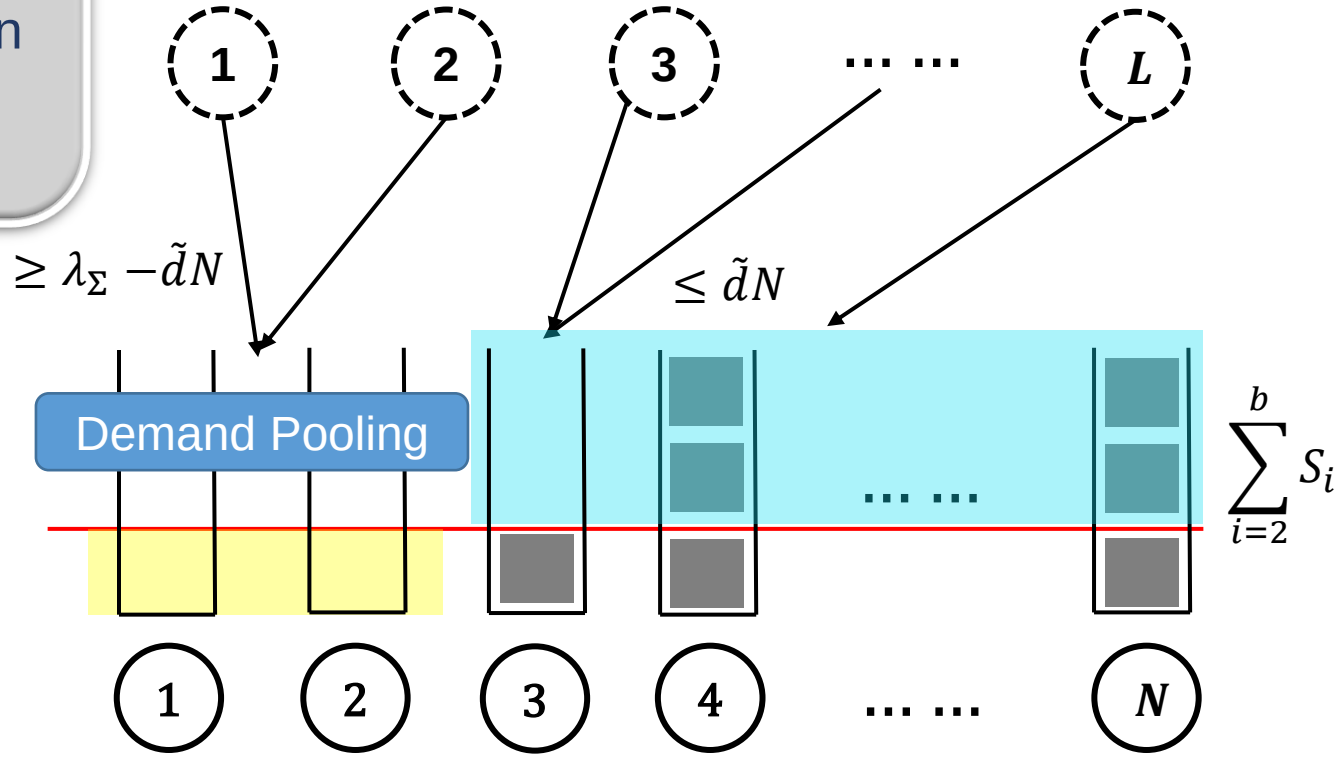
- $V(s) = \min\left(\sum_{i=2}^{b} s_i, \left(\boldsymbol{\lambda + \frac{1}{2}\epsilon + 4\delta - s_1}\right)^+\right)$
- **If $\boldsymbol{V(s) \geq \frac{1}{2}\epsilon + \delta}$**, then $s_1 \leq \lambda + 3\delta$
  - **#of idle servers must be large:** $N(1 - s_1) \geq \frac{N(1-\lambda)}{2}$
  - Show V has negative drift



“Well Connectedness” implies negative drift

Consequence of the “**Well Connectedness**” assumption:
If the number of idle servers is large, then the arrival rate must be large

- # idle servers $= N(1 - s_1) \geq \frac{N(1-\lambda)}{2}$
- Implies # idle servers decreases



Back to Main Result

- Putting it all together yields

$$E\left[\max\left(\sum_{i=1}^{b} S_i - (\lambda+\epsilon), 0\right)^{+}\right] = O\left(\frac{b^2}{\epsilon N}\right)$$

- The analysis of JFSQ is a bit more complicated, but the key ideas presented here are used there as well

Conclusions

- Lyapunov drift is a powerful tool to study the performance of control policies in complex (deterministic and) stochastic systems
  - Two examples: cloud computing and RL
- Not covered here:
  - Stability analysis (books of S.+Ying, Asmussen, Meyn+Tweedie, Meyn)
  - Utility-based resource allocation (books of Neely, S.+Ying)
    - Useful for online learning too
  - Stein's method (Stein, Dai+Braverman, Gurvich, Ying, Stolyar)
  - Reinforcement learning (work with Cayci, Satpathi, He)