Improved Queue-Size Scaling for Input-Queued Switches via Graph Factorization

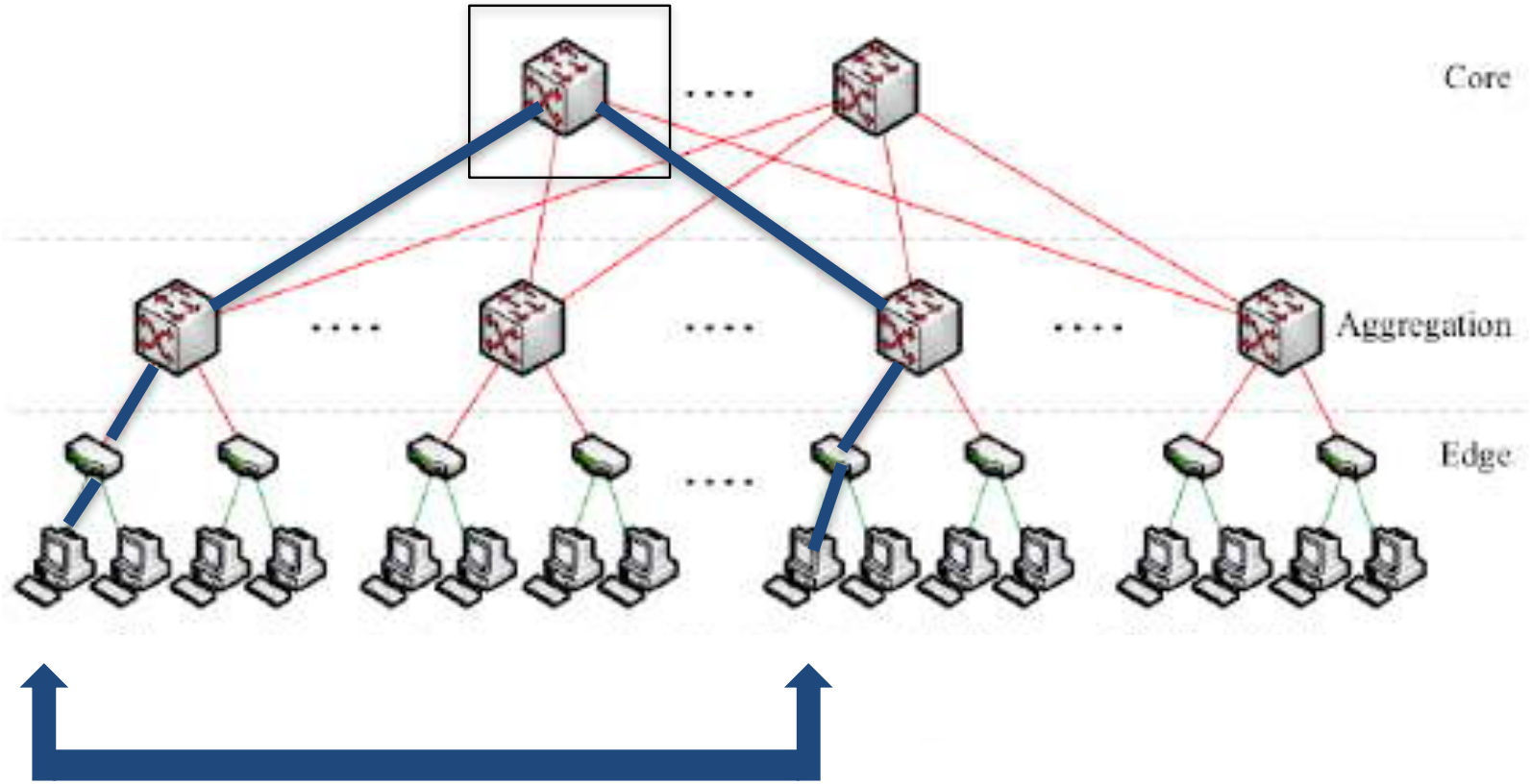
Jiaming Xu

The Fuqua School of Business
Duke University

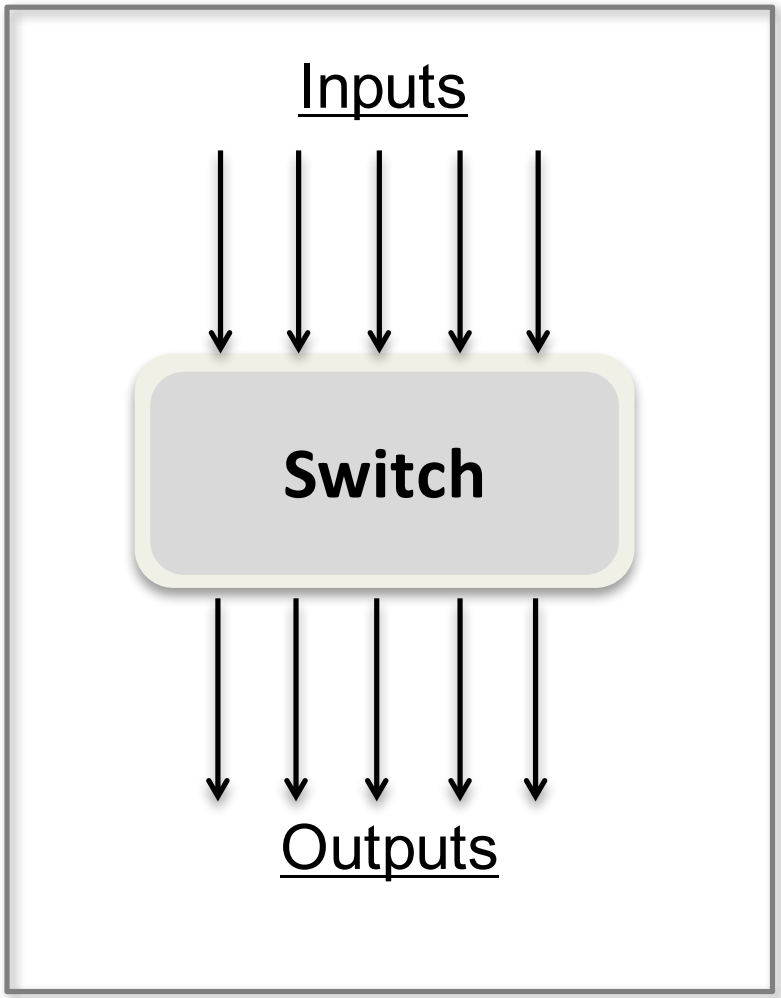
Joint work with
Yuan Zhong (Chicago Booth)

Mostly OM Workshop, June 2, 2019

Data Center Switches

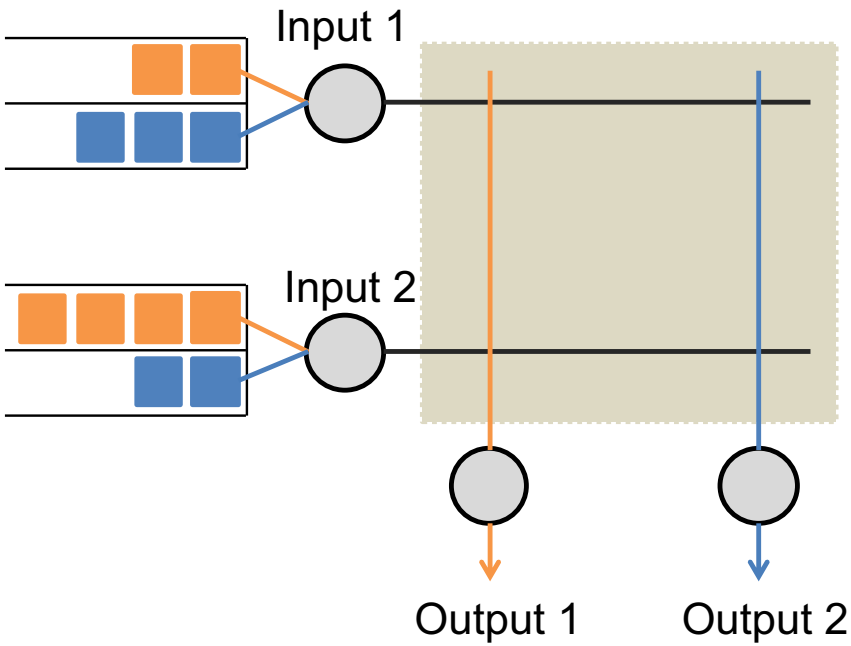


Data Center Switches



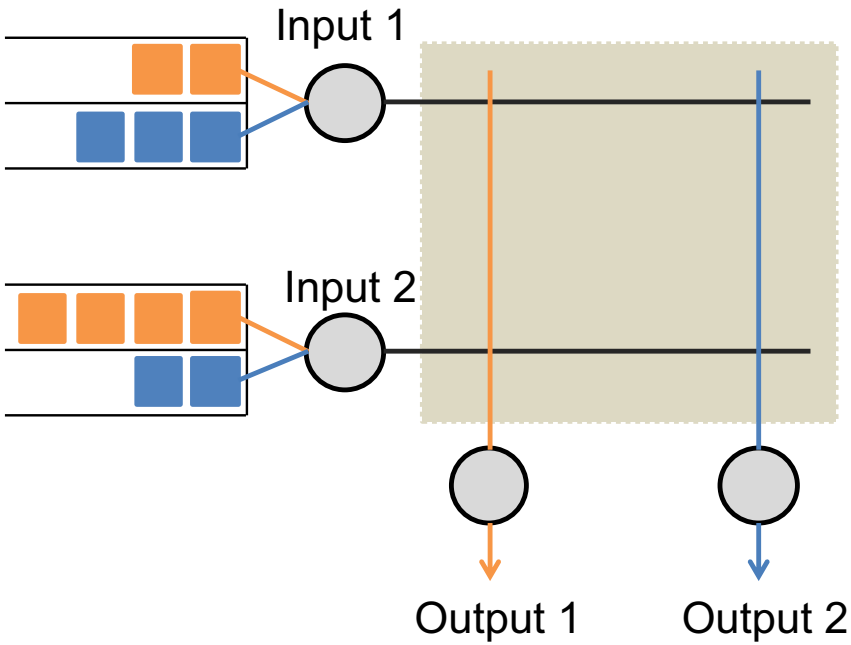
HP Data Center Switch

Input-Queued Switch



- $n \times n$ input-queued switch: n inputs and n outputs
- unit-sized packets
- n^2 queues: (input, output) $\leftrightarrow$ queue

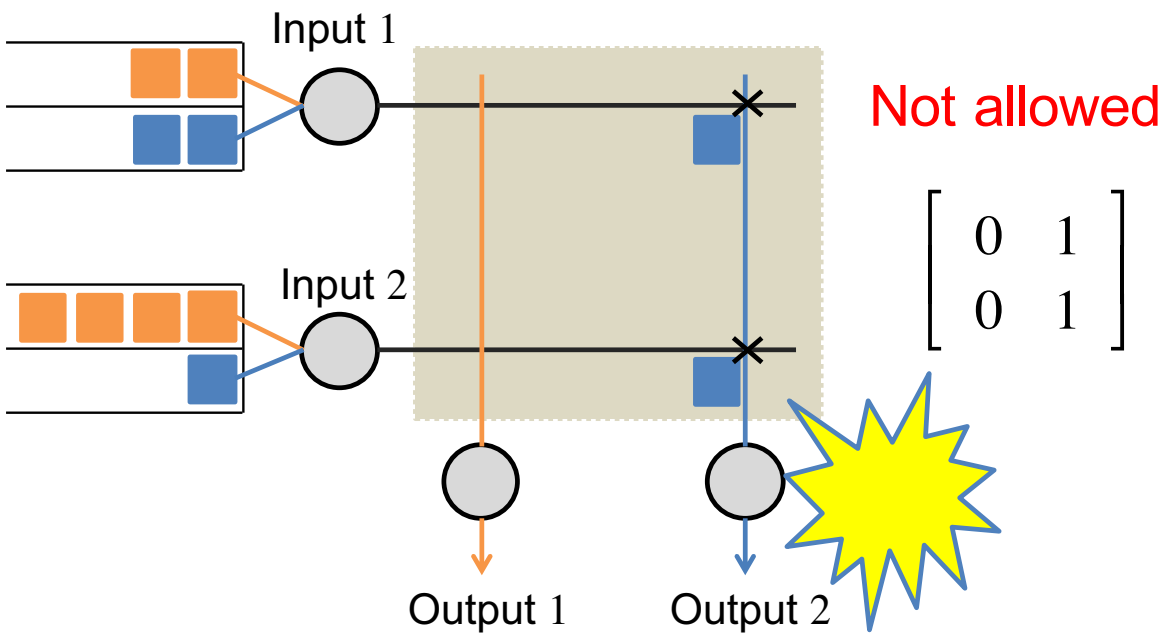
Input-Queued Switch



Matching constraints (2n resource constraints):

- each input can connect to at most one output
- each output can connect to at most one input

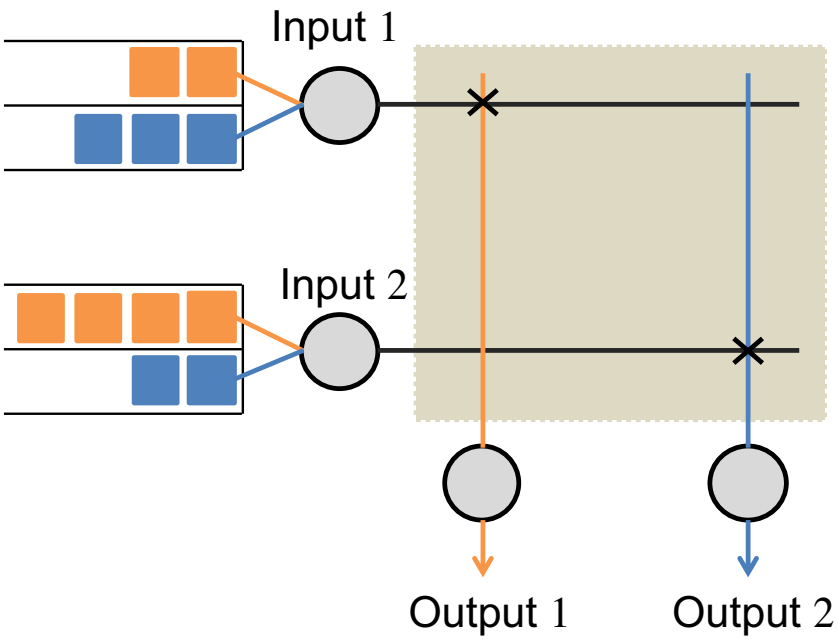
Input-Queued Switch



Matching constraints (2n resource constraints):

- each input can connect to at most one output
- each output can connect to at most one input

Input-Queued Switch



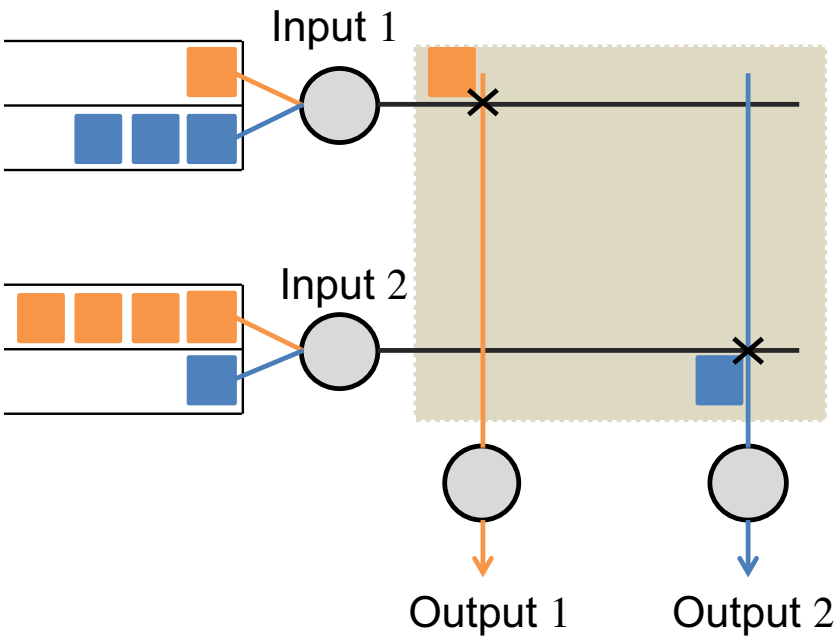
Allowed

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Matching constraints (2n resource constraints):

- each input can connect to at most one output
- each output can connect to at most one input

Input-Queued Switch



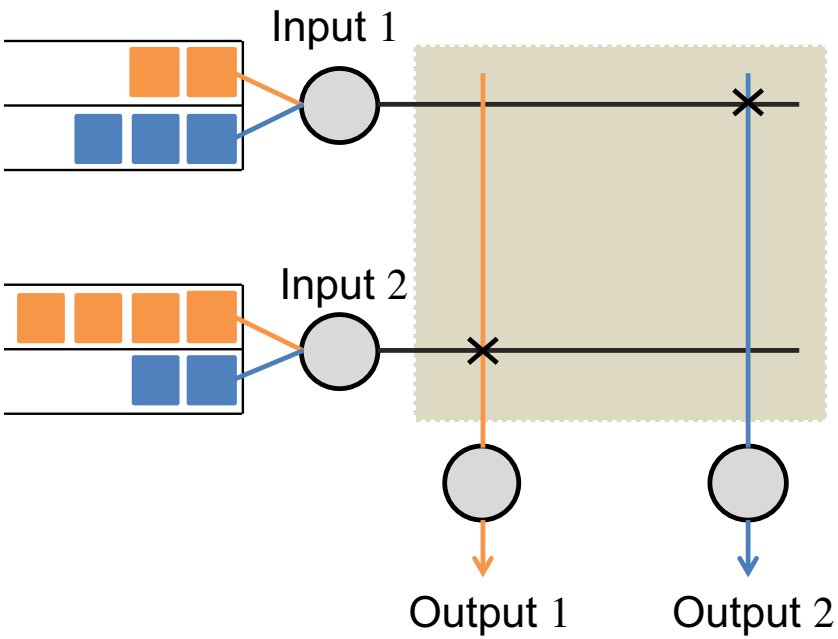
Allowed

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Matching constraints (2n resource constraints):

- each input can connect to at most one output
- each output can connect to at most one input

Input-Queued Switch



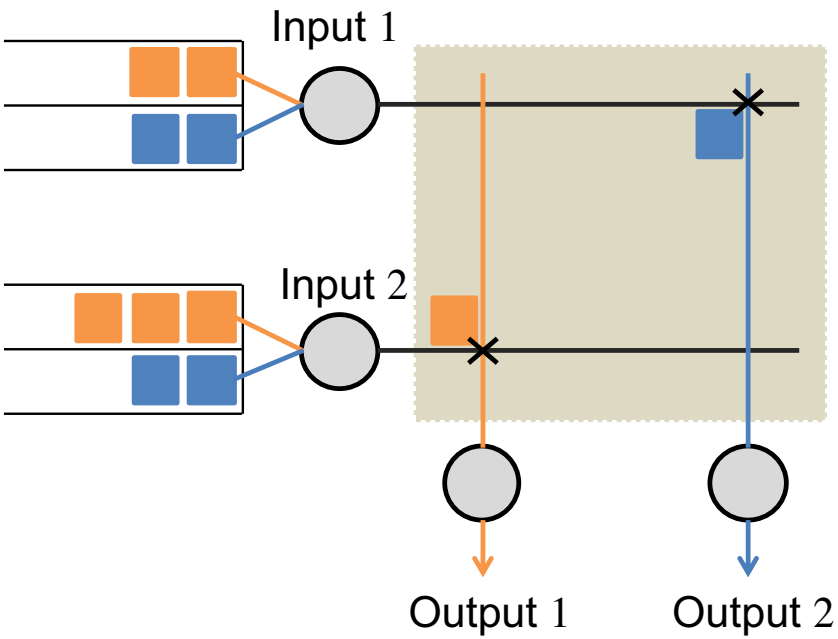
Allowed

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

Matching constraints (2n resource constraints):

- each input can connect to at most one output
- each output can connect to at most one input

Input-Queued Switch



Allowed

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

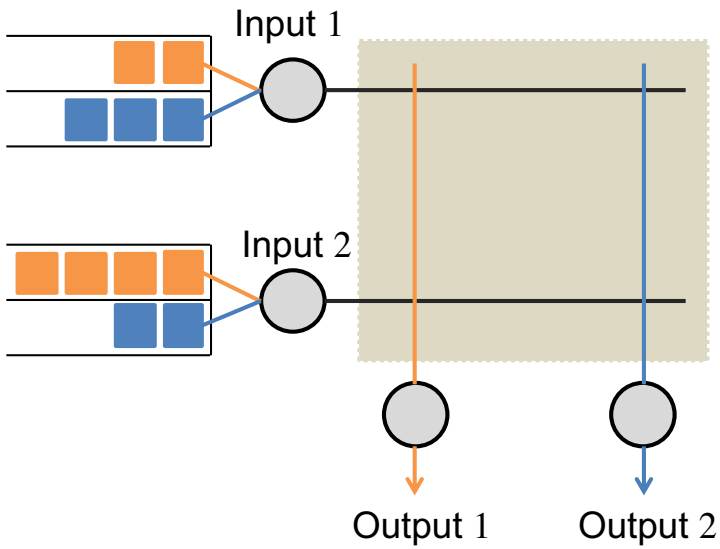
Matching constraints (2n resource constraints):

- each input can connect to at most one output
- each output can connect to at most one input

Queueing Dynamics

$$Q = \begin{bmatrix} 2 & 3 \\ 4 & 2 \end{bmatrix}$$

Q_{12}



- Independent Bernoulli arrivals with rate λ_{ij}
- $\mathbf{\Lambda} = [\lambda_{ij}]$ is admissible if

$$\sum_i \lambda_{ij} < 1 \text{ and } \sum_j \lambda_{ij} < 1$$

- Focus on uniform arrival rates: $\lambda_{ij} = \rho/n$ and

$$\rho = \sum_i \lambda_{ij} = \sum_j \lambda_{ij}$$

- Input-queued switches extensively studied
- Throughput and stability well understood
- Refiner metrics (moments of queue sizes/delay) less understood, but become increasingly important in the big data era

- Input-queued switches extensively studied
- Throughput and stability well understood
- Refiner metrics (moments of queue sizes/delay) less understood, but become increasingly important in the big data era

Focus of this talk:

How $\sum_{ij} \mathbb{E}[Q_{ij}]$ scales with n (large system) and $1-\rho$ (heavy traffic)?

- Input-queued switches extensively studied
- Throughput and stability well understood
- Refiner metrics (moments of queue sizes/delay) less understood, but become increasingly important in the big data era

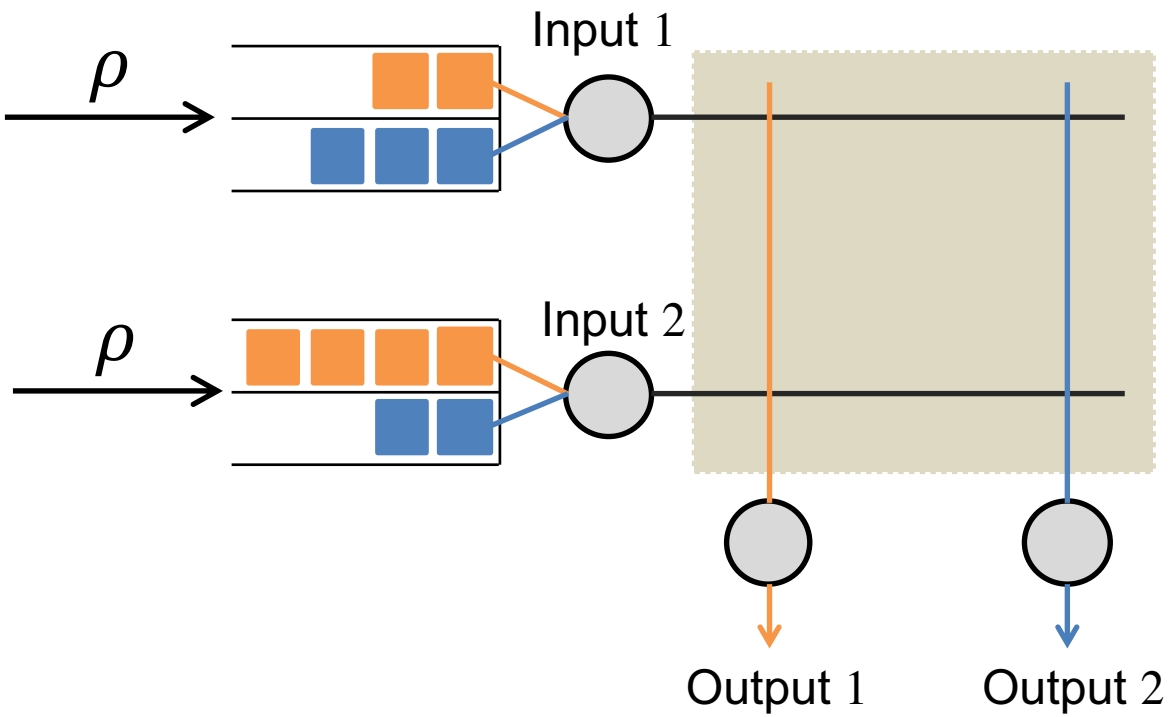
Focus of this talk:

How $\sum_{ij} \mathbb{E}[Q_{ij}]$ scales with n (large system) and $1-\rho$ (heavy traffic)?

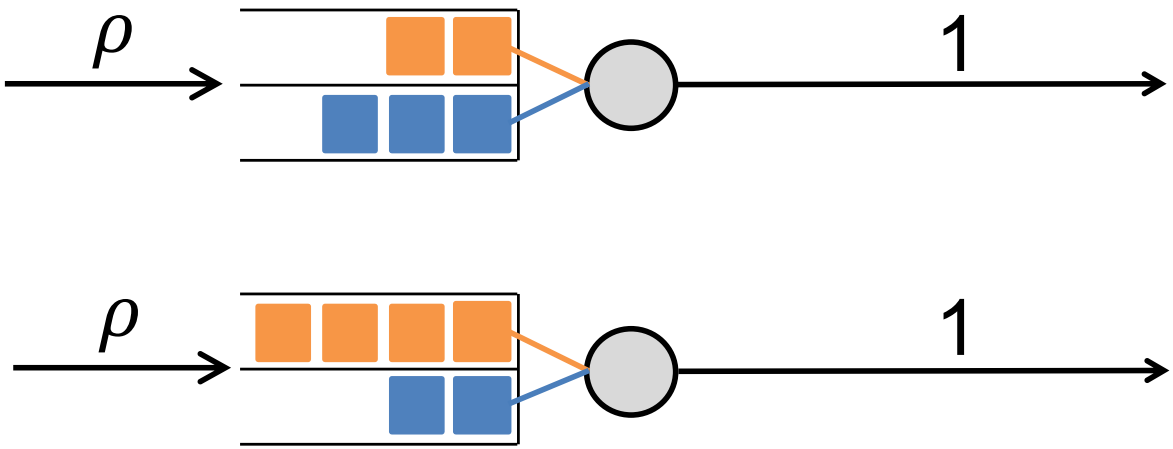
Outline of the remainder

1. A universal lower bound
2. Previously best-known and our improved upper bounds
3. Our policy via batching + graph factorization
4. Summary and concluding remarks

A Universal Lower Bound



A Universal Lower Bound

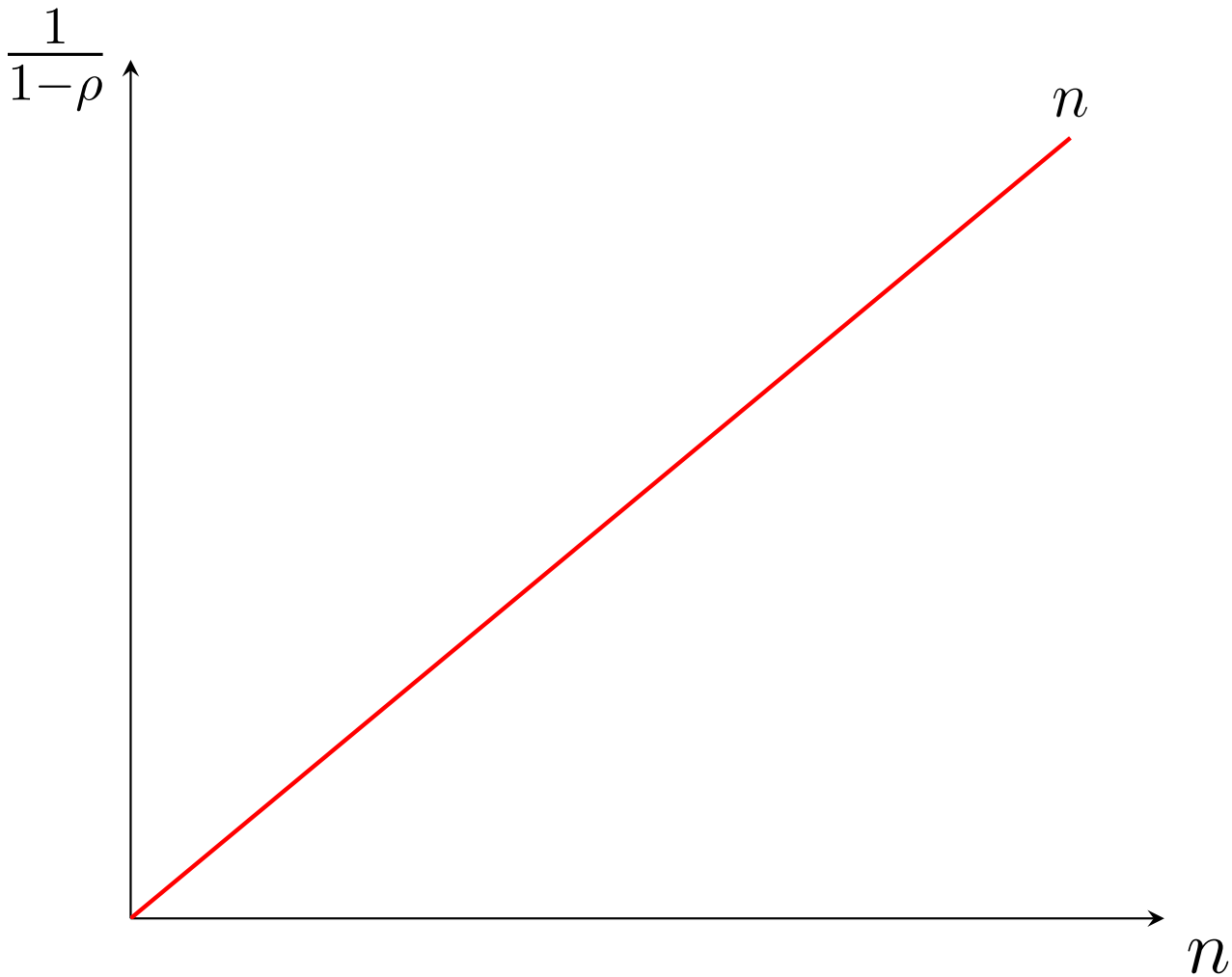


- Decouples into n independent components
- Expected total queue size scales as

$$\frac{n}{1-\rho}$$

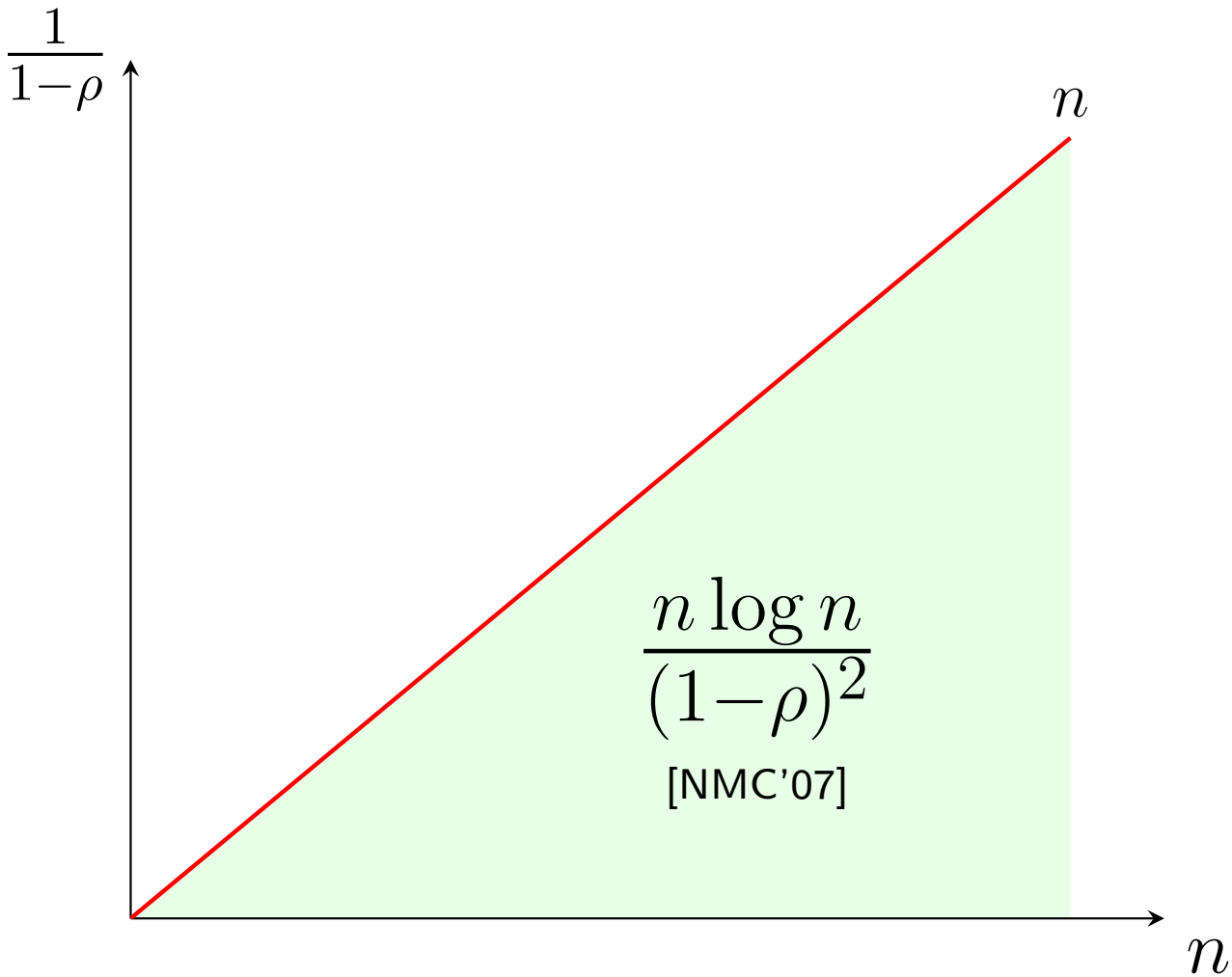
- A universal lower bound for any policy

Previously Best-known Upper Bounds



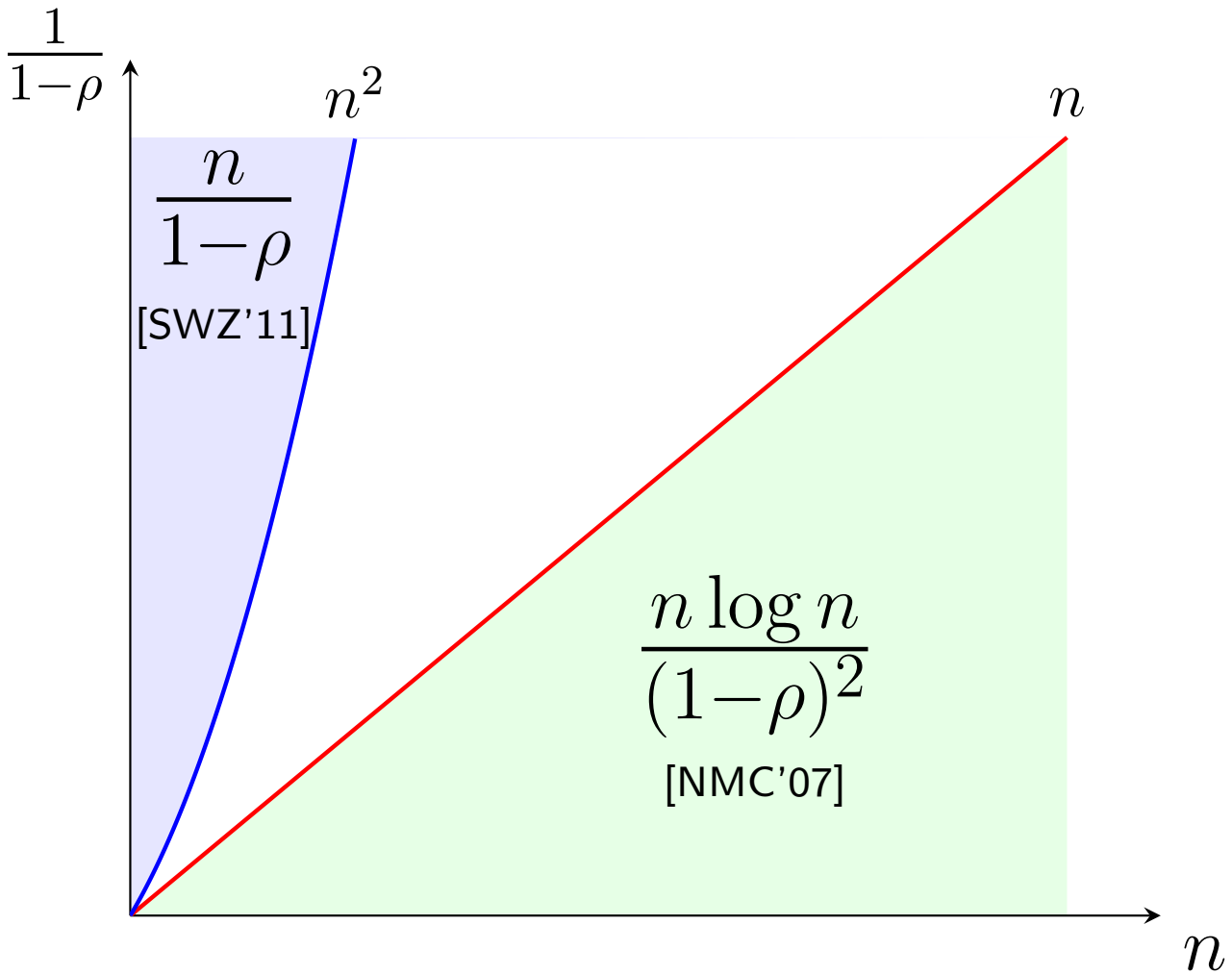
Universal Lower bound: $\frac{n}{1-\rho}$

Previously Best-known Upper Bounds



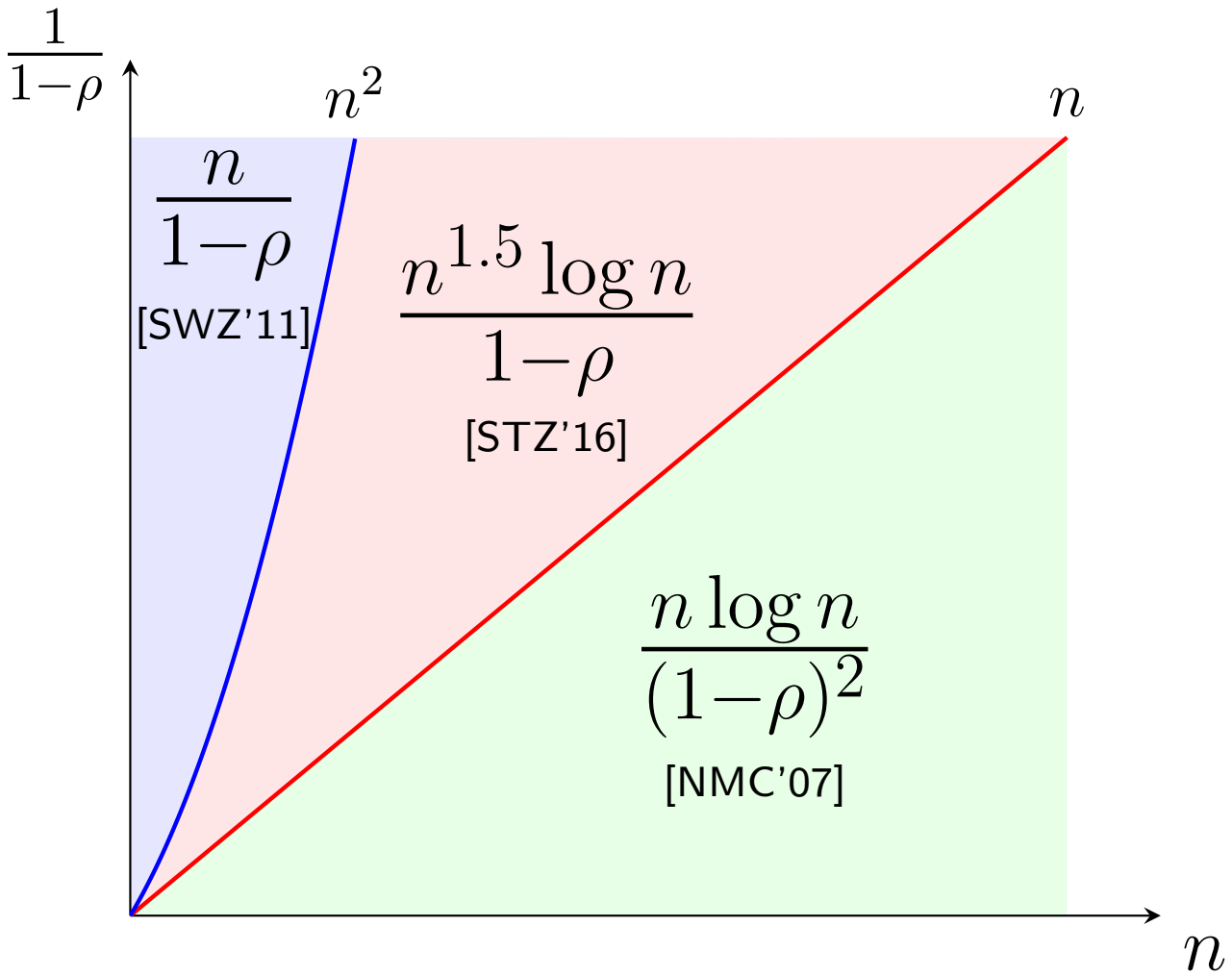
Universal Lower bound: $\frac{n}{1-\rho}$

Previously Best-known Upper Bounds



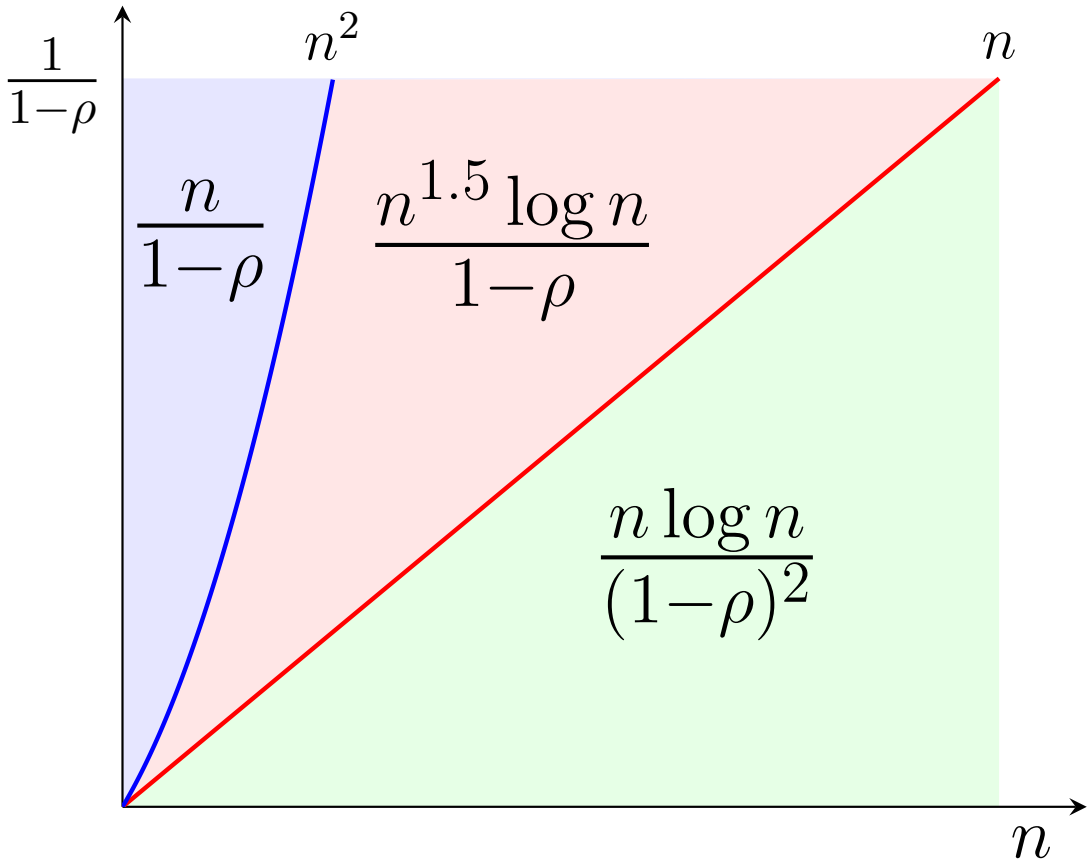
Universal Lower bound: $\frac{n}{1-\rho}$

Previously Best-known Upper Bounds

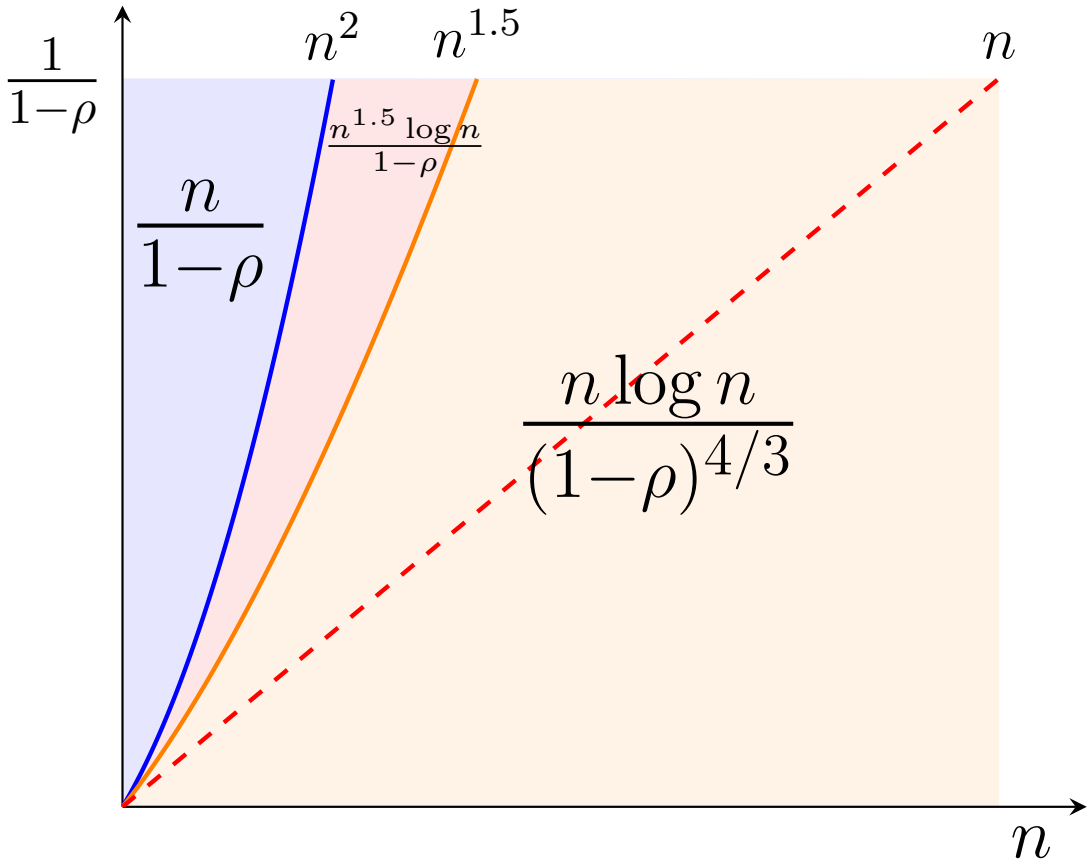


Universal Lower bound: $\frac{n}{1-\rho}$

Our Improved Upper Bound



Our Improved Upper Bound



Improvements:

- $\frac{1}{1-\rho} < n$: $\quad \frac{n\log n}{(1-\rho)^2} \longrightarrow \frac{n\log n}{(1-\rho)^{4/3}}$
- $\frac{1}{1-\rho} = n$: $\quad n^{2.5}\log n \longrightarrow n^{7/3}\log n$
- $n < \frac{1}{1-\rho} \le n^{1.5}$: $\quad \frac{n^{1.5}\log n}{1-\rho} \longrightarrow \frac{n\log n}{(1-\rho)^{4/3}}$

Main Theorem

Theorem (X. and Zhong '19)

Consider an $n \times n$ input-queued switch for which the n^2 arrival streams form independent Bernoulli processes with a common arrival rate ρ/n, where $\rho \in (0,1)$. There exists a scheduling policy under which

$$\mathbb{E}\left[\sum_{i,j=1}^{n} Q_{ij}(\tau)\right] \leq c\frac{n}{(1-\rho)^{4/3}} \log \frac{n}{1-\rho}, \quad \forall \tau \in \mathbb{N}$$

Main Theorem

Theorem (X. and Zhong '19)

Consider an $n \times n$ input-queued switch for which the n^2 arrival streams form independent Bernoulli processes with a common arrival rate ρ/n, where $\rho \in (0,1)$. There exists a scheduling policy under which

$$\mathbb{E}\left[\sum_{i,j=1}^{n} Q_{ij}(\tau)\right] \leq c\frac{n}{(1-\rho)^{4/3}} \log \frac{n}{1-\rho}, \quad \forall \tau \in \mathbb{N}$$

Remarks

- A multiplicative factor $\frac{1}{(1-\rho)^{1/3}} \log \frac{n}{1-\rho}$ away from the lower bound
- Computational complexity per slot is at most polynomial in n

Key innovation: efficient scheduling via graph factorization

Question

Given a queue matrix $\mathbf{q} = (q_{ij})_{i,j=1}^{n}$, how to deplete the packets as much as possible without wasting service opportunities?

Key innovation: efficient scheduling via graph factorization

Question

Given a queue matrix $\mathbf{q} = (q_{ij})_{i,j=1}^{n}$, how to deplete the packets as much as possible without wasting service opportunities?

$$
\begin{aligned}
k^* = \max_{k,\mathbf{g}} \quad & k \\
\text{s.t.} \quad & \sum_i g_{ij} = \sum_j g_{ij} = k \\
& g_{ij} \le q_{ij} \quad \text{no service waste} \\
& g_{ij} \in \mathbb{N}
\end{aligned}
$$

Key innovation: efficient scheduling via graph factorization

Question

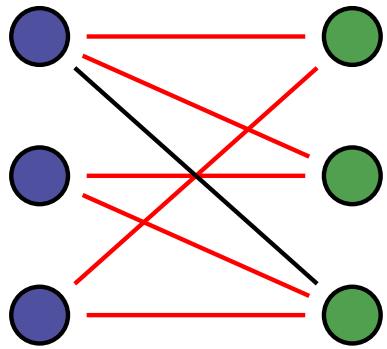
Given a queue matrix $\mathbf{q} = (q_{ij})_{i,j=1}^{n}$, how to deplete the packets as much as possible without wasting service opportunities?

$$k^* = \max_{k,\mathbf{g}} \quad k$$

$$\text{s.t.} \quad \sum_i g_{ij} = \sum_j g_{ij} = k$$

$$g_{ij} \le q_{ij} \quad \text{no service waste}$$

$$g_{ij} \in \mathbb{N}$$



- $\mathbf{g}$ can be viewed as a k-factor (spanning k-regular graph) of a bipartite multigraph $\mathbf{q}$

Key innovation: efficient scheduling via graph factorization

Question

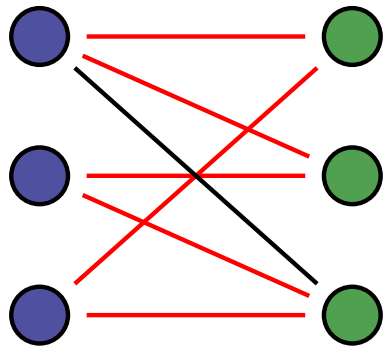
Given a queue matrix $\mathbf{q} = (q_{ij})_{i,j=1}^{n}$, how to deplete the packets as much as possible without wasting service opportunities?

$$k^* = \max_{k,\mathbf{g}} \quad k$$

$$\text{s.t.} \quad \sum_i g_{ij} = \sum_j g_{ij} = k$$

$$g_{ij} \le q_{ij} \quad \text{no service waste}$$

$$g_{ij} \in \mathbb{N}$$



- $\mathbf{g}$ can be viewed as a k-factor (spanning k-regular graph) of a bipartite multigraph $\mathbf{q}$
- A simple upper bound: $k^* \le \min\left\{\min_i \sum_j q_{ij},\ \min_j \sum_i q_{ij}\right\}$
- Is the upper bound tight?

Key innovation: efficient scheduling via graph factorization

Question

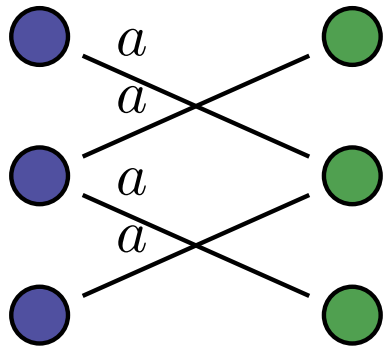
Given a queue matrix $\mathbf{q} = (q_{ij})_{i,j=1}^{n}$, how to deplete the packets as much as possible without wasting service opportunities?

$$k^* = \max_{k,\mathbf{g}} \quad k$$

$$\text{s.t.} \quad \sum_i g_{ij} = \sum_j g_{ij} = k$$

$$g_{ij} \le q_{ij} \quad \text{no service waste}$$

$$g_{ij} \in \mathbb{N}$$



- $\mathbf{g}$ can be viewed as a k-factor (spanning k-regular graph) of a bipartite multigraph $\mathbf{q}$
- A simple upper bound: $k^* \le \min\left\{\min_i \sum_j q_{ij},\ \min_j \sum_i q_{ij}\right\}$
- Is the upper bound tight?

Largest k-factor of random queue matrices (multigraphs)

Theorem (X. and Zhong '19)

Let $\mathbf{q} = (q_{ij})_{i,j=1}^{n}$ be an $n \times n$ queue matrix with $q_{ij} \overset{\text{i.i.d.}}{\sim} \text{Binom}(m, p)$. With probability $1 - n^{-16}$, $\mathbf{q}$ has a k-factor with

$$k \geq pmn - \sqrt{304pmn \log n}.$$

- Matches the upper bound up to a constant factor:

$$k^* \leq \min \left\{ \min_i \sum_j q_{ij}, \ \min_j \sum_i q_{ij} \right\} \leq pmn - \sqrt{pmn \log n}$$

Largest k-factor of random queue matrices (multigraphs)

Theorem (X. and Zhong '19)

Let $\mathbf{q} = (q_{ij})_{i,j=1}^{n}$ *be an* $n \times n$ *queue matrix with* $q_{ij} \overset{\text{i.i.d.}}{\sim} \text{Binom}(m, p)$. *With probability* $1 - n^{-16}$, $\mathbf{q}$ *has a* k*-factor with*

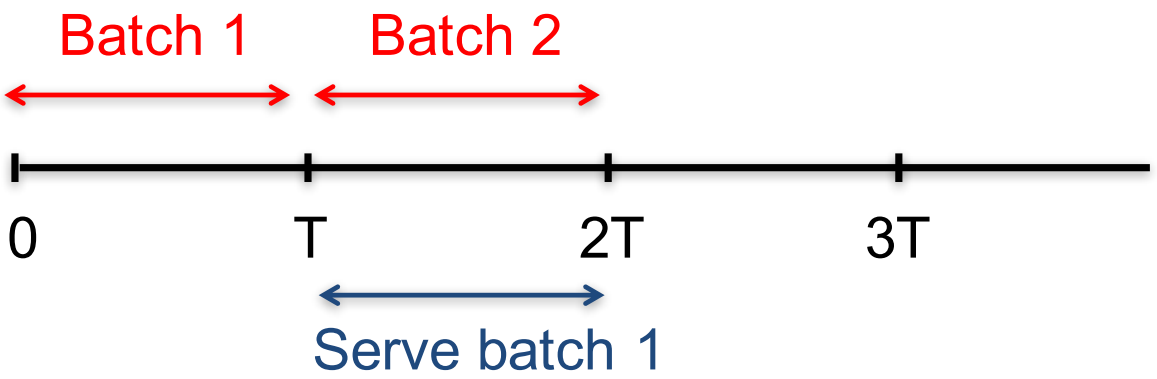
$$k \geq pmn - \sqrt{304pmn \log n}.$$

- Matches the upper bound up to a constant factor:

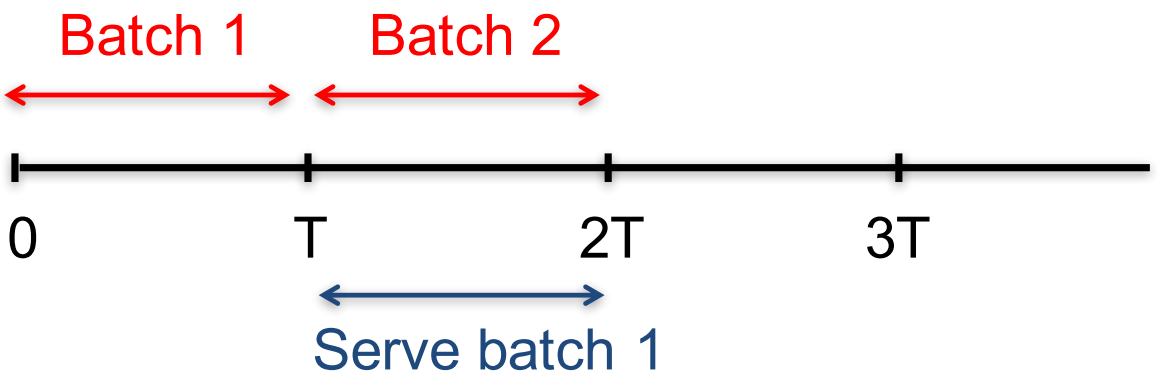
$$k^* \leq \min \left\{ \min_i \sum_j q_{ij}, \ \min_j \sum_i q_{ij} \right\} \leq pmn - \sqrt{pmn \log n}$$

- Proof based on Gale-Ryser Theorem (extension of max-flow min-cut) + Large deviation analysis

A Standard Batching Policy [NMC'07]

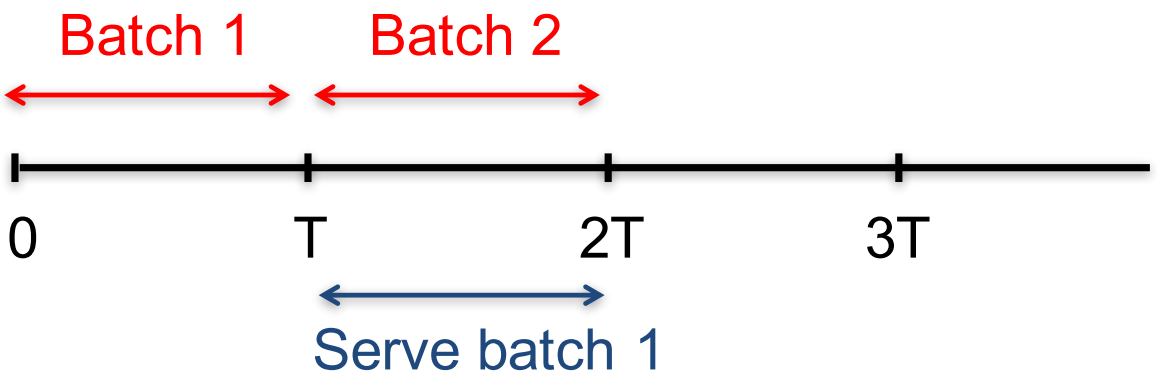


A Standard Batching Policy [NMC'07]



- No. of arrivals to any input/output port in time T $\sim \mathrm{Binom}(nT, \rho/n)$
- Max no. of arrivals to any input/output port in time T $\approx \rho T + \sqrt{T \log n}$

A Standard Batching Policy [NMC'07]

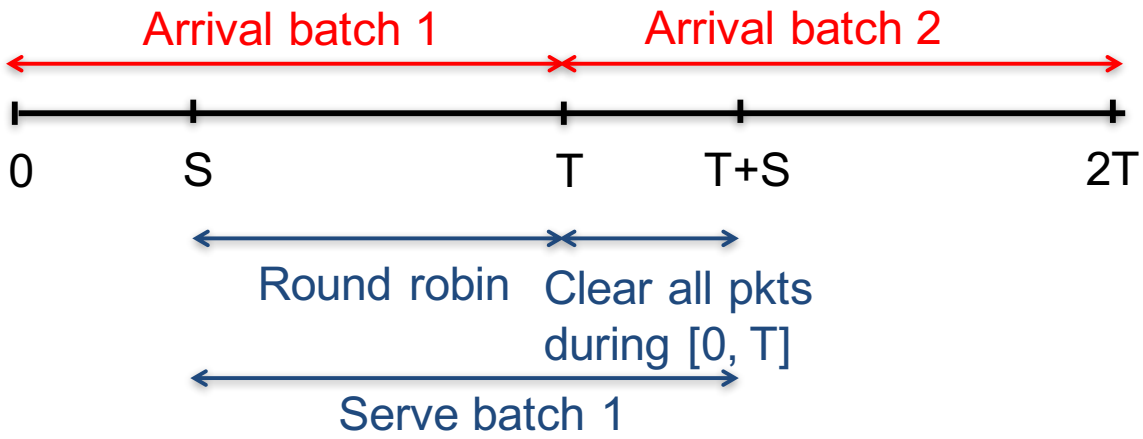


- No. of arrivals to any input/output port in time T $\sim \text{Binom}(nT, \rho/n)$
- Max no. of arrivals to any input/output port in time T $\approx \rho T + \sqrt{T \log n}$
- Finishing serving a batch in time T needs

$$T \geq \rho T + \sqrt{T \log n} \iff T \geq \frac{\log n}{(1-\rho)^2}$$

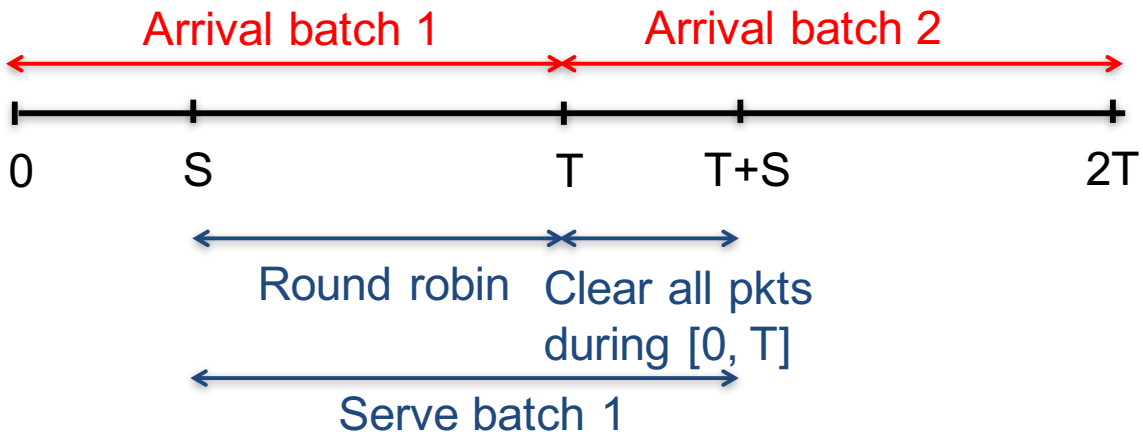
- Expected total queue size $\approx nT \geq \frac{n \log n}{(1-\rho)^2}$

An Impatient Batching Policy [STZ'16]



- Start serving before the arrival of entire batch
  1. Wait for S time slots
  2. Simple round-robin for $T - S$ time slots

An Impatient Batching Policy [STZ'16]

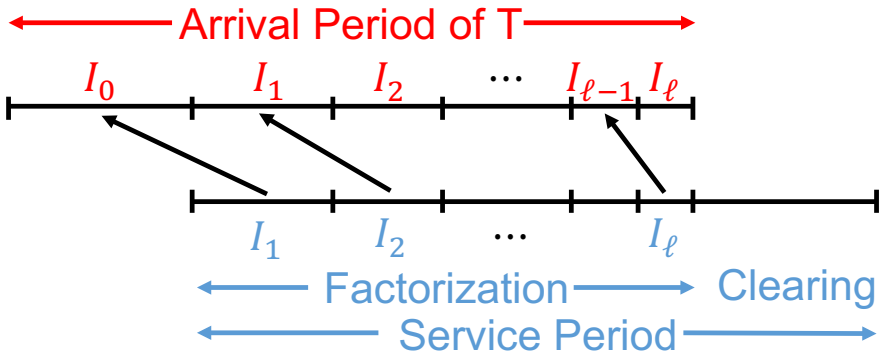


- Start serving before the arrival of entire batch
  1. Wait for S time slots
  2. Simple round-robin for $T - S$ time slots
- Need to ensure no waste of service opportunities during round-robin

$$\frac{T-S}{n} \leq \rho\frac{T}{n} - \sqrt{\frac{T\log n}{n}} \iff S \geq (1-\rho)T + \sqrt{nT\log n}$$

- $T \asymp \frac{\log n}{(1-\rho)^2} \Rightarrow$ Expected total queue size $\approx nS \geq \frac{n^{1.5}\log n}{1-\rho}$

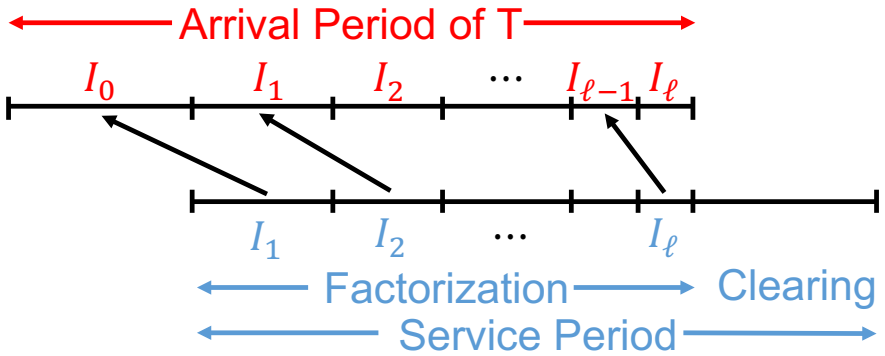
Our Improved Batching Policy



- Start serving even earlier
  1. Wait for I_0 time slots
  2. Serve packets via factorization for $T - I_0$ time slots:

$$I_u \text{ serves arrivals in } I_{u-1} \text{ for } 1 \leq u \leq \ell$$

Our Improved Batching Policy



- Start serving even earlier
  1. Wait for I_0 time slots
  2. Serve packets via factorization for $T - I_0$ time slots:

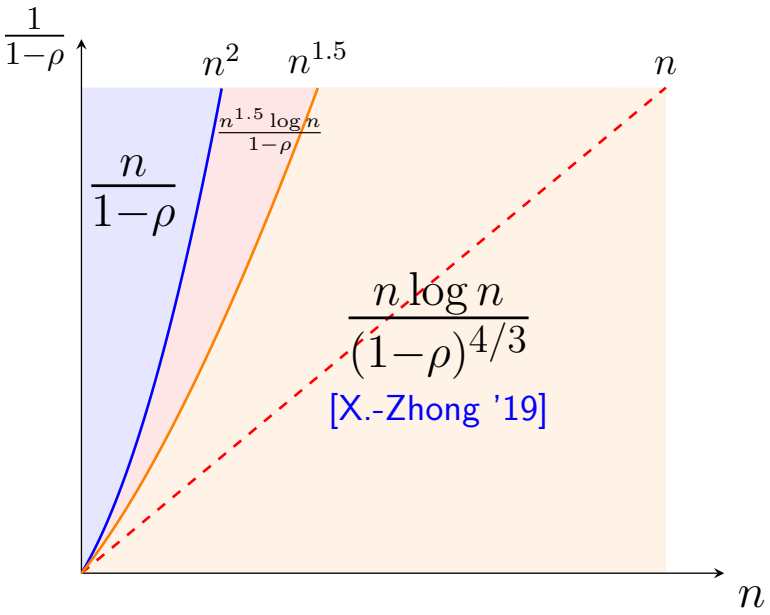
$$I_u \text{ serves arrivals in } I_{u-1} \text{ for } 1 \le u \le \ell$$

- To ensure no waste of service opportunities during factorization

$$\begin{cases} I_u \le \rho I_{u-1} - \sqrt{I_u \log n} \\ I_0 \ge I_1 \ge \cdots \ge I_\ell \asymp I_0 \\ I_0 + I_1 + \cdots + I_\ell = T \end{cases} \iff I_0 \asymp T^{2/3} \log^{1/3} n$$

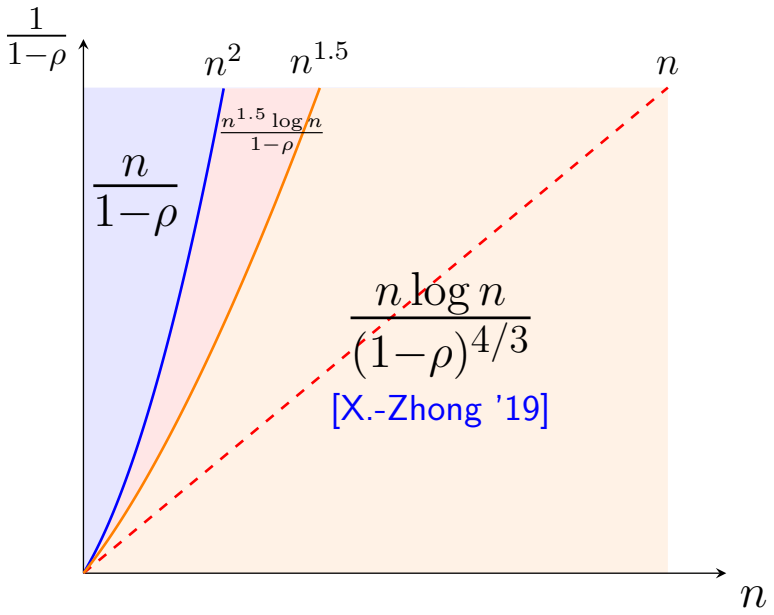
- $T \asymp \frac{\log n}{(1-\rho)^2} \Rightarrow$ Expected total queue size $\approx nI_0 \asymp \frac{n \log n}{(1-\rho)^{4/3}}$

Conclusion and remarks



1. Improved queue-size scalings via graph factorization
2. A tight characterization of the largest k-factor in random bipartite multigraphs

Conclusion and remarks



1. Improved queue-size scalings via graph factorization
2. A tight characterization of the largest k-factor in random bipartite multigraphs

Open problem

- Achieving the universal lower bound $\frac{n}{1-\rho}$?

References

- X. & Yuan Zhong (2019). *Improved Queue-Size Scaling for Input-Queued Switches via Graph Factorization*, ACM SIGMETRICS 2019, arXiv:1903.00398.