# Scalable Load Balancing in the Presence of Heterogeneous Servers

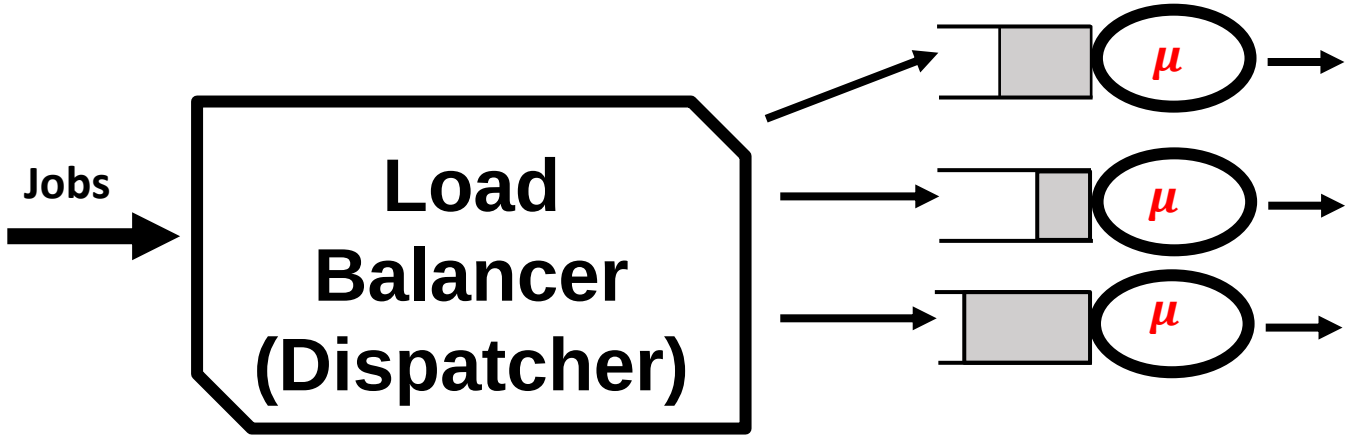
Kristen Gardner[1], Jazeem Abdul Jaleel[2], Alexander Wickeham[2], Sherwin Doroudi[2]

**Performance 2020, November 3rd 2020**

[1] Computer Science Department, Amherst College,
[2] Industrial and Systems Engineering, University of Minnesota

# Introduction

Scalable Load Balancing in the Presence of Heterogeneous Servers




- Load Balancing – homogeneous servers

| Load Balancing Policies | |
|---|---|
| Random | Round Robin |
| **J**oin **S**hortest **Q**ueue (JSQ) | **J**oin **I**dle **Q**ueue (JIQ) |

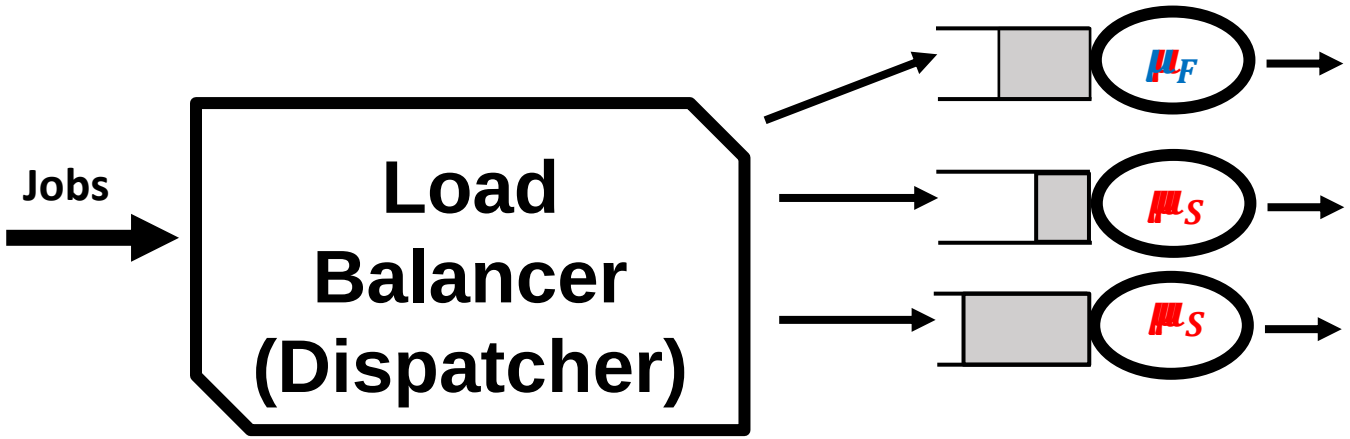
Weber 1978, Winston 1977, Lu *et al* 2011, Wang *et al* 2018

Examples
- AWS
- IBM network dispatcher
- Citrix ADC
- Cisco Local director

***Important***

# Introduction

Scalable Load Balancing in the Presence of Heterogeneous Servers

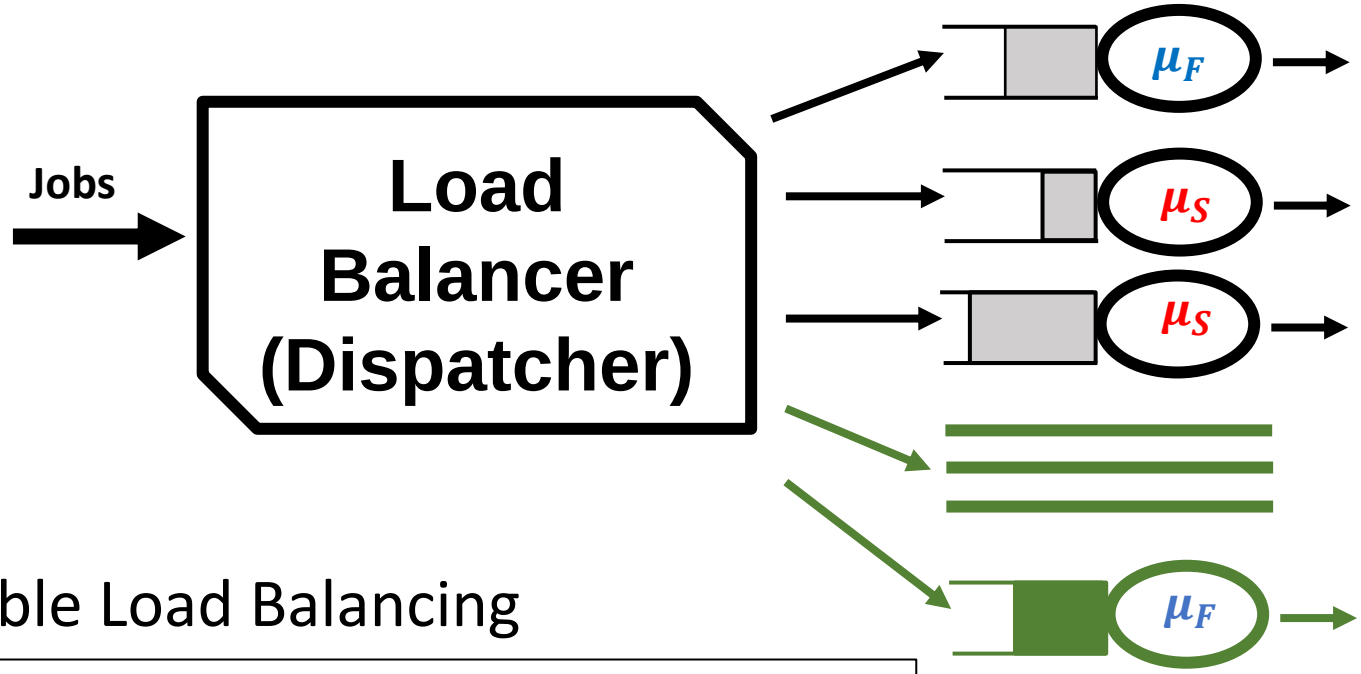



- Load Balancing of heterogeneous Server Systems
  (Slow server problems)

Load balancing policies
- Threshold type
  Hyytiä 2013, Luh et al 2002

# Introduction

Scalable Load Balancing in the Presence of Heterogeneous Servers




Difficulties
- Complex
- Error prone
- Costly

- Scalable Load Balancing

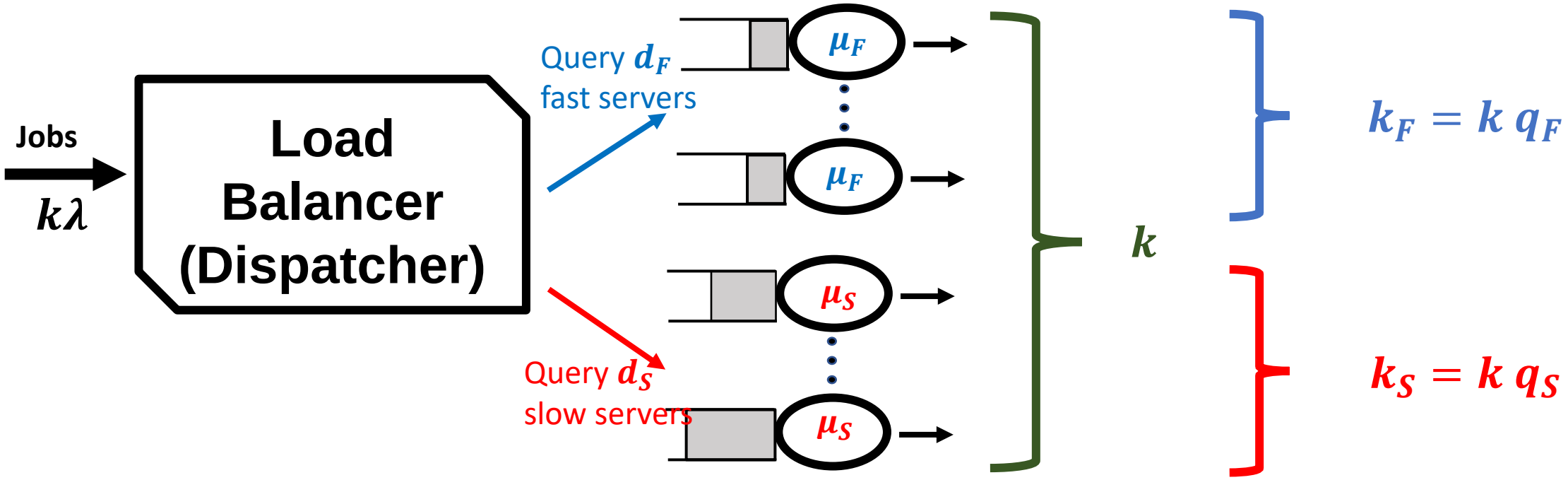
Power of *d* policies (query d)
- **J**oin **S**hortest **Q**ueue – query **d (JSQ-d)**
- **J**oin **I**dle **Q**ueue – query **d (JIQ-d)**

Mitzenmacher 2001
Bramson et al 2012
Li et al. 2014

# Outline

- System model
- Explain the policies:
  - $\mathbf{J}\text{oin } \mathbf{I}\text{dle } \mathbf{Q}\text{ueue} - (\boldsymbol{d_F}, \boldsymbol{d_S})$
  - $\mathbf{J}\text{oin } \mathbf{S}\text{hortest } \mathbf{Q}\text{ueue} - (\boldsymbol{d_F}, \boldsymbol{d_S})$
- Stability of the proposed policies
- Techniques used and the optimization problem
- Performance comparison

# Model



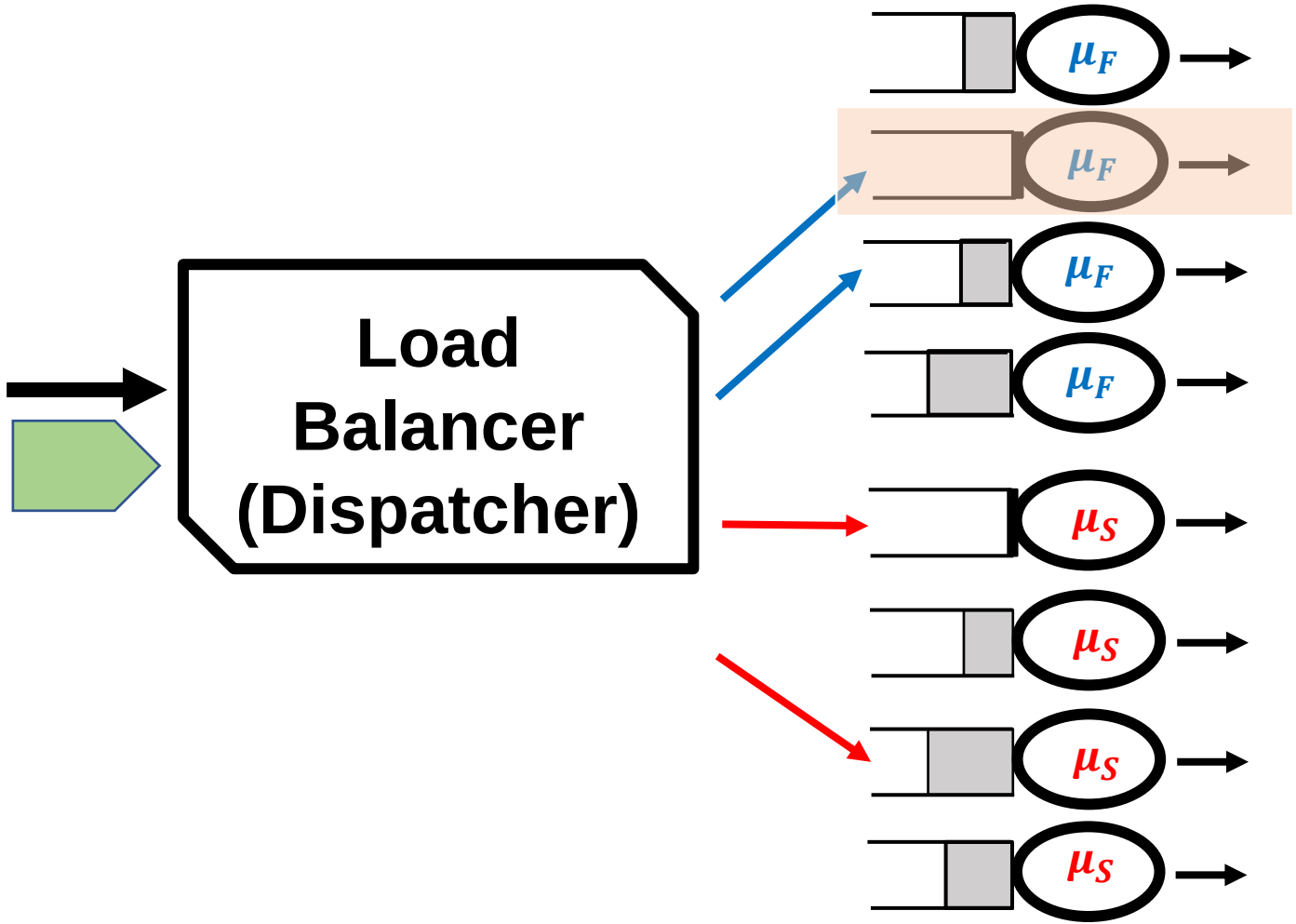

**Policies:**

1. Join Idle Queue $-(d_F, d_S)$ — JIQ $-(d_F, d_S)$
2. Join Shortest Queue $-(d_F, d_S)$ — JSQ $-(d_F, d_S)$

# JIQ – $(d_F, d_S)$

Query randomly $d_F$ fast servers and $d_S$ slow servers.

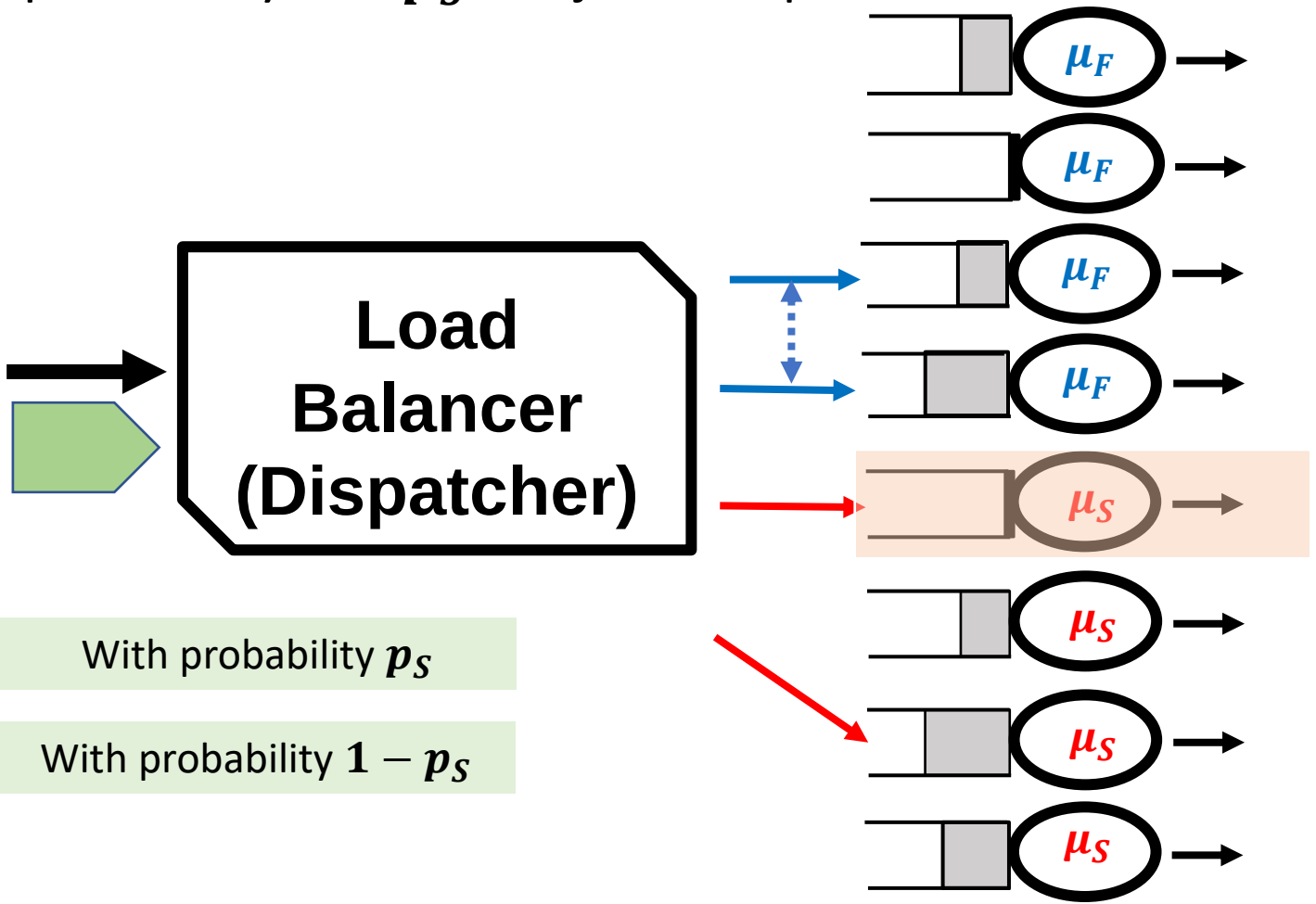
1. If any of the $d_F$ fast servers are idle, the job begins service on one of them.




$d_F = 2$
$d_S = 2$

# JIQ $-(d_F, d_S)$

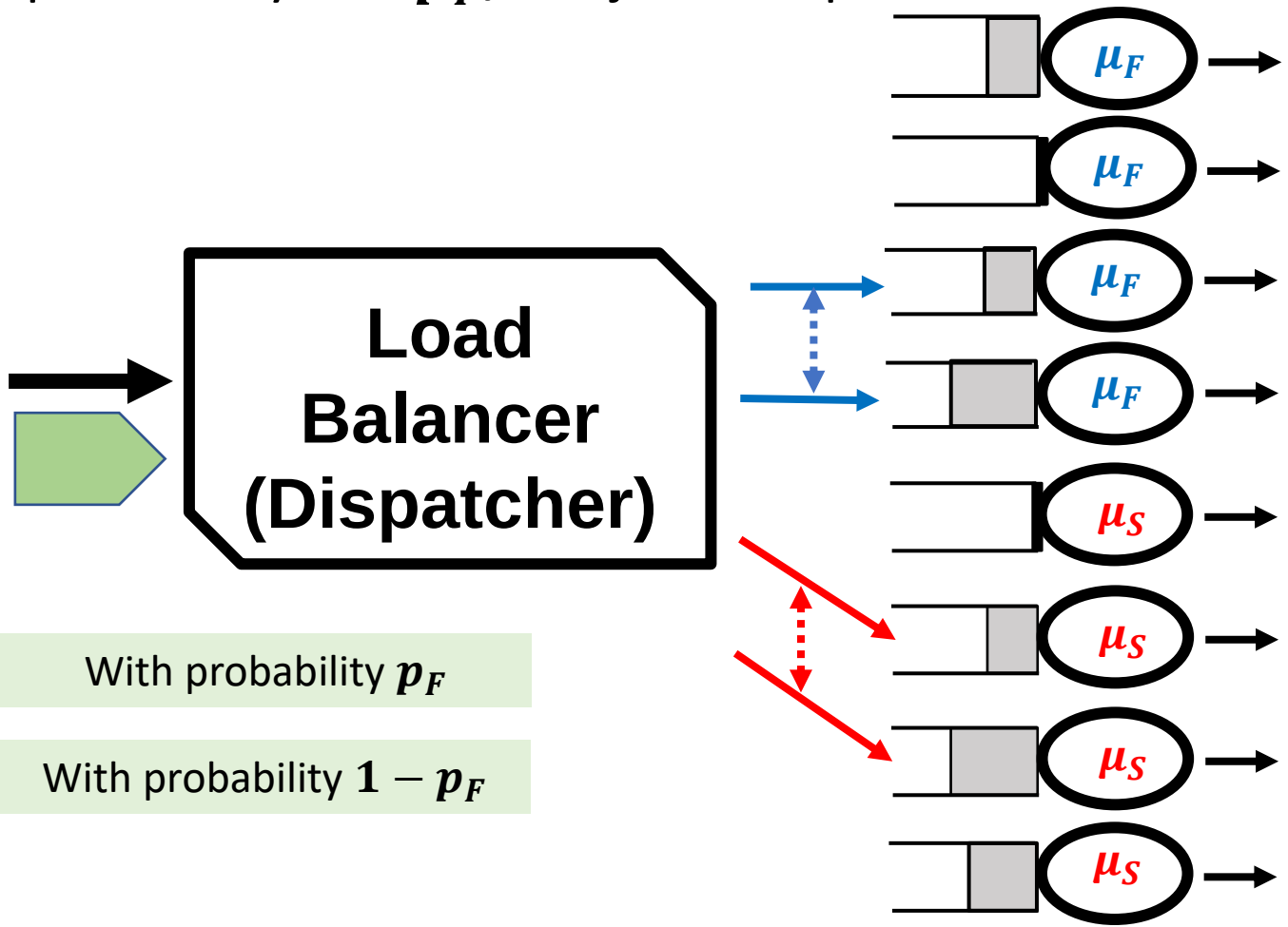
2. If all $d_F$ fast servers are busy and any of the $d_S$ slow servers are idle,
   - with probability $p_S$, the job begins service on a random queried idle slow server.
   - with probability $1 - p_S$ the job is dispatched to a random queried busy fast server




$d_F = 2$
$d_S = 2$

# JIQ $-(d_F, d_S)$

3. If all $d_F + d_S$ queried servers are busy,
   - with probability $p_F$, the job is dispatched to a random queried busy fast server.
   - with probability $1 - p_F$, the job is dispatched to a random queried busy slow server.




$d_F = 2$
$d_S = 2$

# JIQ $-(d_F, d_S)$

Query randomly $d_F$ fast servers and $d_S$ slow servers.

1. If any of the $d_F$ fast servers are idle, the job begins service on one of them.

2. If all $d_F$ fast servers are busy and any of the $d_S$ slow servers are idle,
   - with probability $p_S$, the job begins service on a random queried idle slow server.
   - with probability $1 - p_S$ the job is dispatched to a **random** queried busy fast server

3. If all $d_F + d_S$ queried servers are busy,
   - with probability $p_F$, the job is dispatched to a **random** queried busy fast server.
   - with probability $1 - p_F$, the job is dispatched to a **random** queried busy slow server.

# JSQ $-(d_F, d_S)$

When job arrives, query at random $d_F$ fast servers and $d_S$ slow servers.

1. If any of the $d_F$ fast servers are idle, the job begins service on one of them.

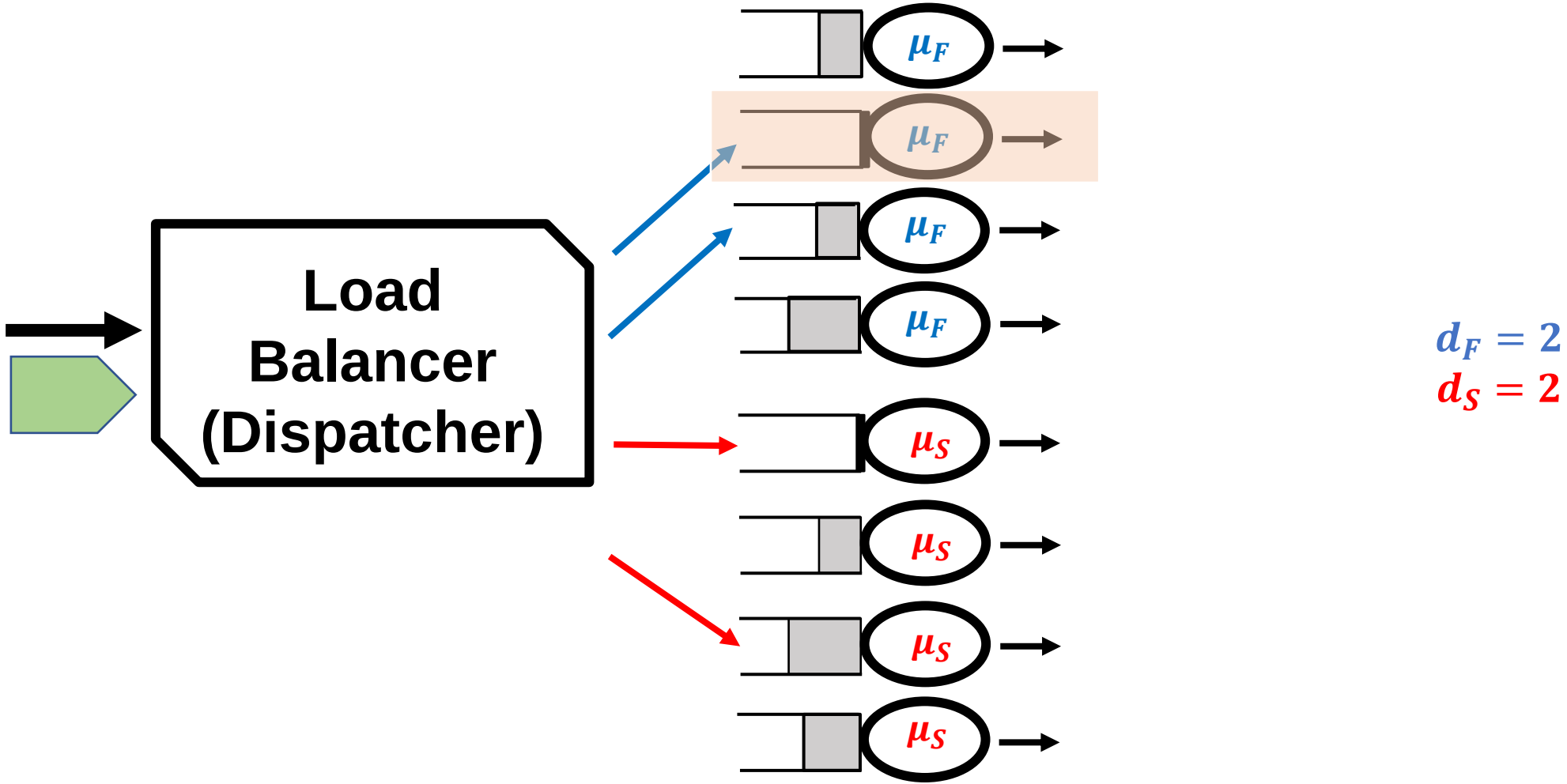
2. If all $d_F$ fast servers are busy and any of the $d_S$ slow servers are idle,
   - with probability $p_S$, the job begins service on a random queried idle slow server.
   - with probability $1 - p_S$, the job is dispatched to the fast server with **shortest queue** among the $d_F$ queried fast servers

3. If all $d_F + d_S$ queried servers are busy,
   - with probability $p_F$, the job is dispatched to the fast server with **shortest queue** among the $d_F$ queried fast servers
   - with probability $1 - p_F$, the job is dispatched to the slow server with **shortest queue** among the $d_S$ queried slow servers

# JSQ $-(d_F, d_S)$

Query randomly $d_F$ fast servers $d_S$ slow servers.

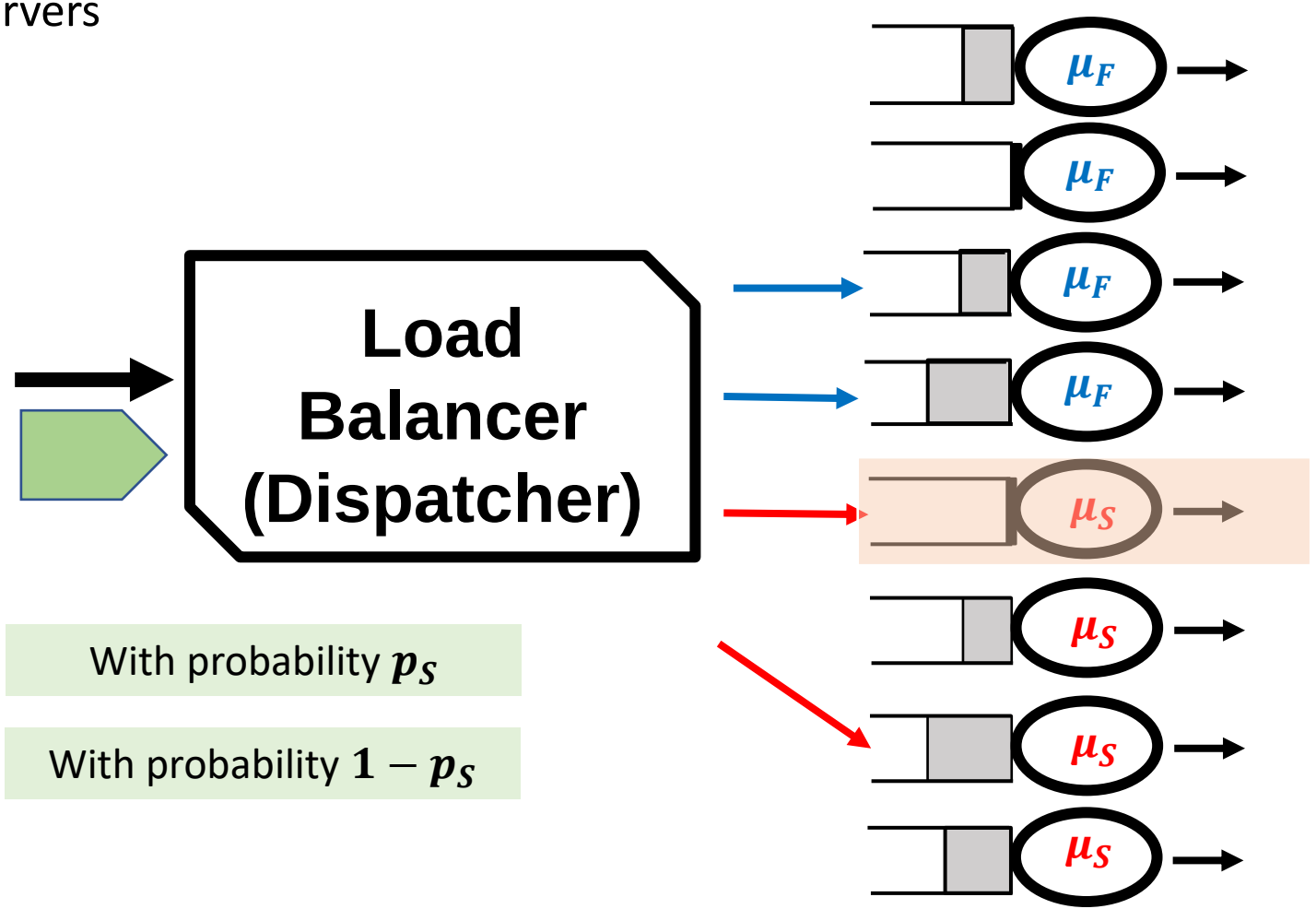
1. If any of the $d_F$ fast servers are idle, the job begins service on one of them.




$d_F = 2$
$d_S = 2$

# JSQ $-(d_F, d_S)$

2. If all $d_F$ fast servers are busy and any of the $d_S$ slow servers are idle,
   - with probability $p_S$, the job begins service on a random queried idle slow server.
   - with probability $1 - p_S$, the job is dispatched to the fast server with shortest queue among the $d_F$ queried fast servers

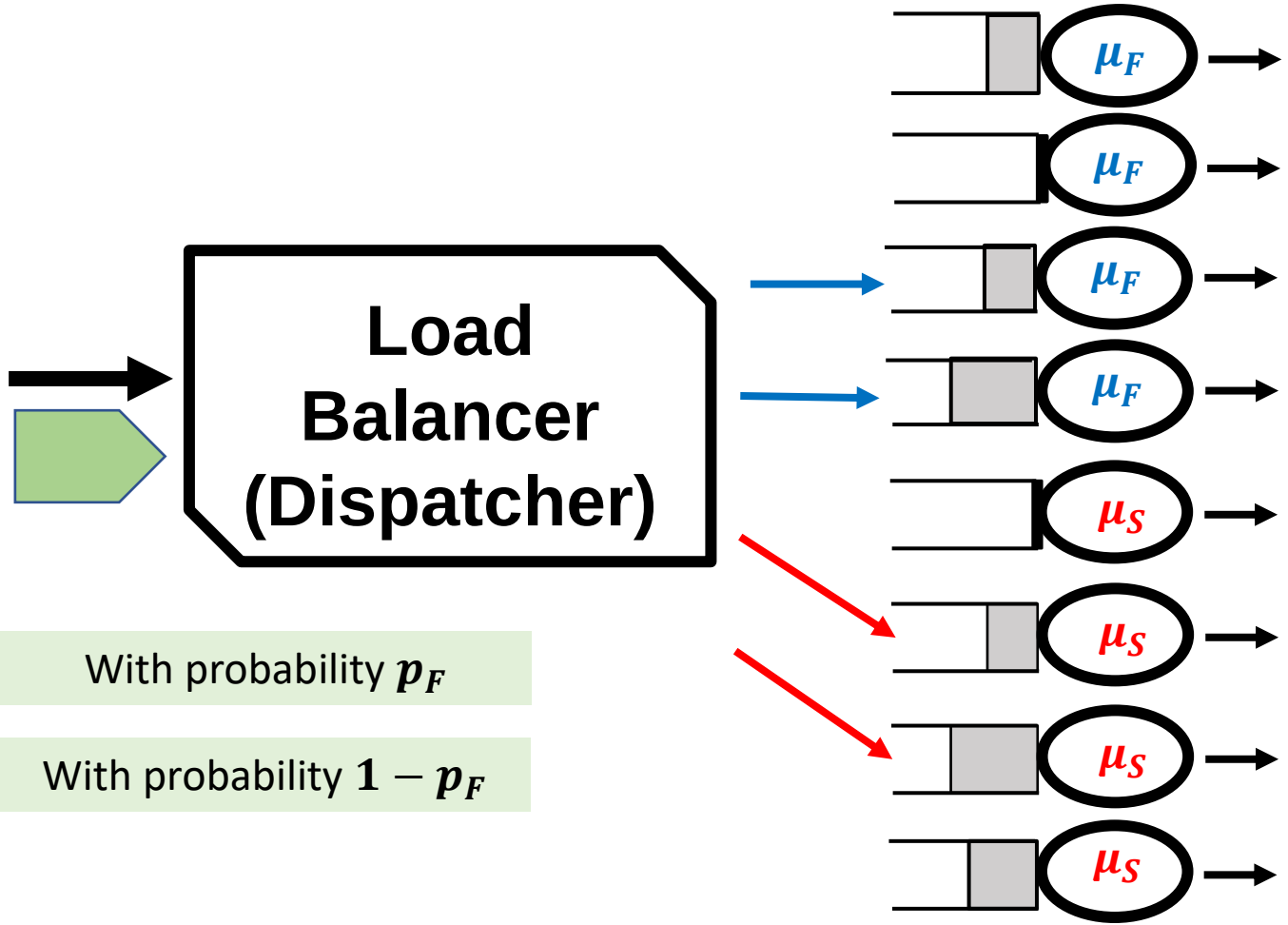



$d_F = 2$
$d_S = 2$

# JSQ $-(d_F, d_S)$

3. If all $d_F + d_S$ queried servers are busy,

- with probability $p_F$, the job is dispatched to the fast server with shortest queue among the $d_F$ fast servers
- with probability $1 - p_F$, the job is dispatched to the slow server with shortest queue among the $d_S$ slow servers




$d_F = 2$
$d_S = 2$

# Theorem 1 - Stability

*Theorem*

Under both $\mathbf{JIQ} - (\boldsymbol{d_F}, \boldsymbol{d_S})$ and $\mathbf{JSQ} - (\boldsymbol{d_F}, \boldsymbol{d_S})$, for any values of $\boldsymbol{d_F}, \boldsymbol{d_S} \geq 1$, there exist choices of $\boldsymbol{p_F}$ and $\boldsymbol{p_S}$ such that the system is stable for $\lambda < \mu_F q_F + \mu_S q_S$.

# Calculating Mean Response Time

- **Mean Field Analysis:** asymptotic independence assumption

- $\mathbf{JIQ} - (\boldsymbol{d_F}, \boldsymbol{d_S})$:
  - Tagged server approach
  - General Service time distributions

- $\mathbf{JSQ} - (\boldsymbol{d_F}, \boldsymbol{d_S})$:
  - Solved differential equations to obtain the limiting probabilities of the servers
  - Exponential Service time distribution

# Optimization Problem

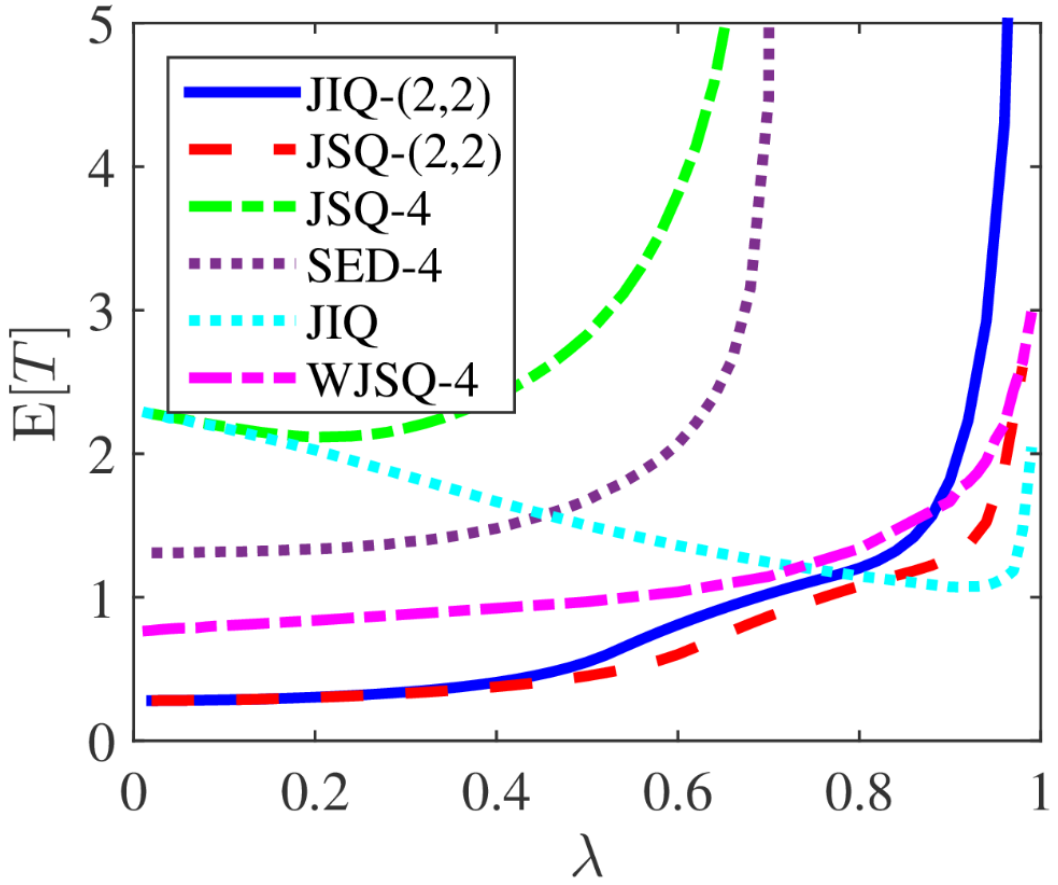
- Minimize Expected Response Time
- Problem Parameters $\lambda, \mu_F, \mu_S, q_F, q_S, d_F, d_S$
- Decision Variables $p_F, p_S$

# Performance comparison

$\mathbf{JIQ} - (\boldsymbol{d_F}, \boldsymbol{d_S})$ and $\mathbf{JSQ} - (\boldsymbol{d_F}, \boldsymbol{d_S})$ against:

- $\mathbf{J}$oin $\mathbf{S}$hortest $\mathbf{Q}$ueue $- (\boldsymbol{d})$
- $\mathbf{S}$hortest $\mathbf{E}$xpected $\mathbf{D}$elay $- (\boldsymbol{d})$
- $\mathbf{W}$eighted $\mathbf{J}$oin $\mathbf{S}$hortest $\mathbf{Q}$ueue $- (\boldsymbol{d})$
- $\mathbf{J}$oin $\mathbf{I}$dle $\mathbf{Q}$ueue

# Performance comparison

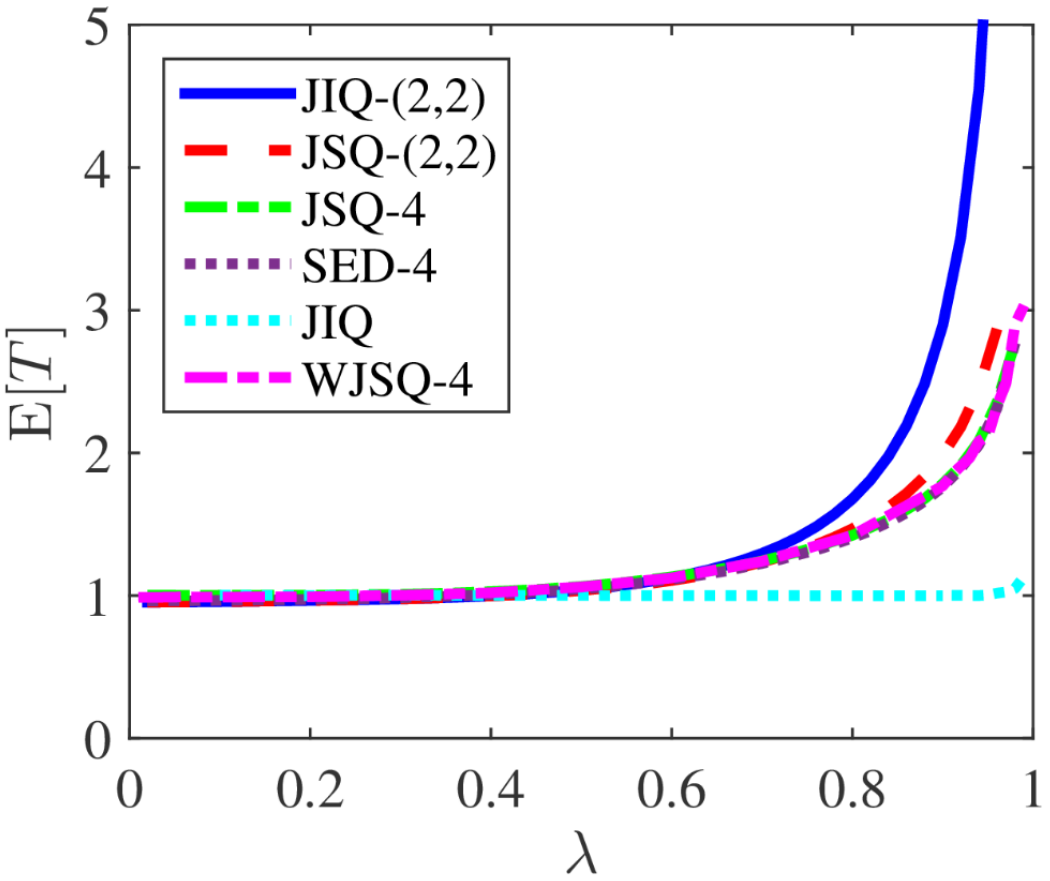



Percentage of fast servers= 20%

$$\frac{\mu_F}{\mu_S} = 10$$

- Apparent instability of $\mathbf{JSQ-(4)}$ and $\mathbf{SED-(4)}$
- Stability of $\mathbf{JIQ-(2,2)}$ and $\mathbf{JSQ-(2,2)}$ by Theorem 1
- At low loads, **JIQ** does not efficiently utilize the fast servers

# Performance comparison




Percentage of fast servers= 50%

$$\frac{\mu_F}{\mu_S} = 1.1$$

**Main Takeaways**

- Stability of $\mathbf{JIQ} - (\boldsymbol{d_F}, \boldsymbol{d_S})$ and $\mathbf{JSQ} - (\boldsymbol{d_F}, \boldsymbol{d_S})$ by Theorem 1
- The best dispatching policy will depend on the problem parameters.

# Conclusion

- Introduced and analyzed new scalable power of d load balancing policies for heterogeneous systems
- Established the stability of the proposed policies
- Identified parameter settings where the policies outperform others
- Future Work: A more general framework
  - see MAMA 2020

# Thank you