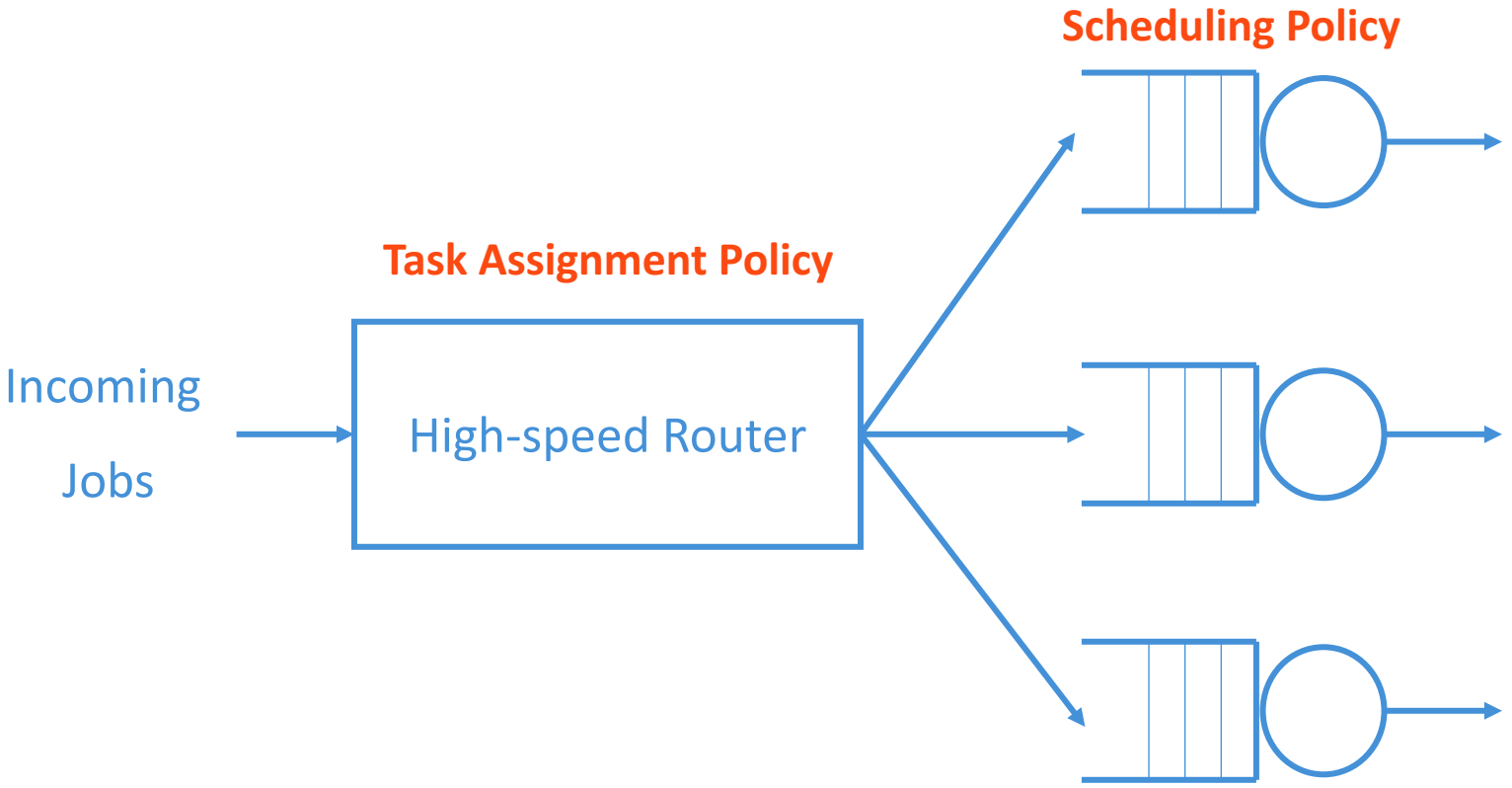
Task Assignment Policies for Server Farms

Reference: Chapter 24 of Prof. Mor Harchol-Balter's textbook

Slides by Jianyu Wang (jianyuw1@andrew.cmu.edu)

Server Farm Model



Rather than using a single, powerful server, it is more cost efficient to buy many slow, inexpensive servers.

Outline

A. **Policies Introduction**

B. Performance Comparisons

C. Optimal Analysis

D. Discussions

Assignment Policies

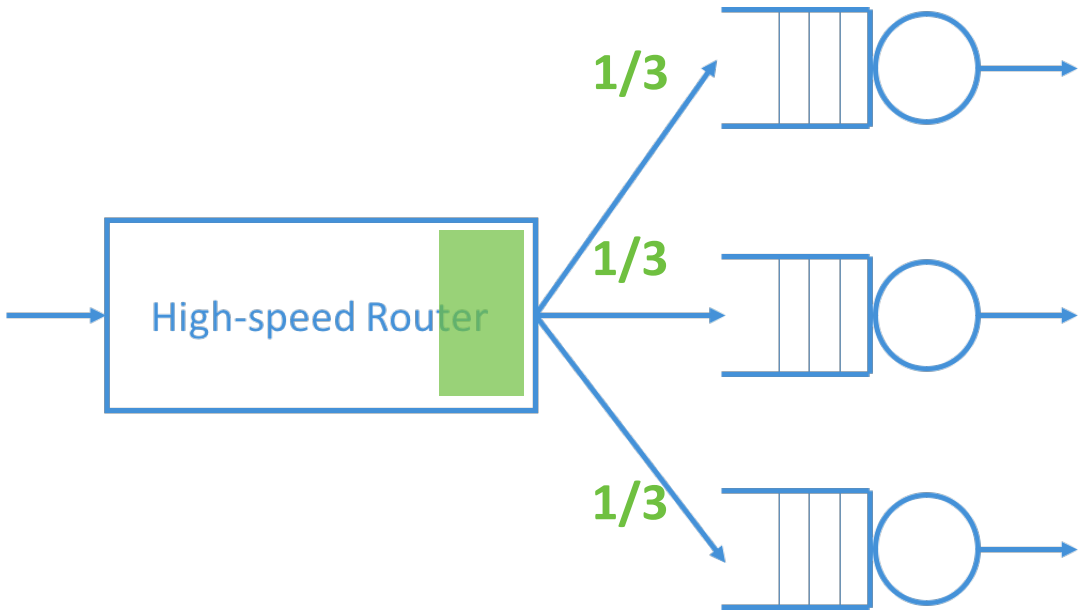
1) **Random**
2) **Round-Robin**
3) **JSQ (Join-the-Shortest-Queue)**
4) **M/G/k**
5) *** LWL (Least-Work-Left)**
6) *** SITA (Size-Interval-Task-Assignment)**

Assignment Policies

1. **Random** [Static]

Each job is assigned to one of the k hosts with equal probability.

The aim is to **equalize the expected number of jobs at each host**.

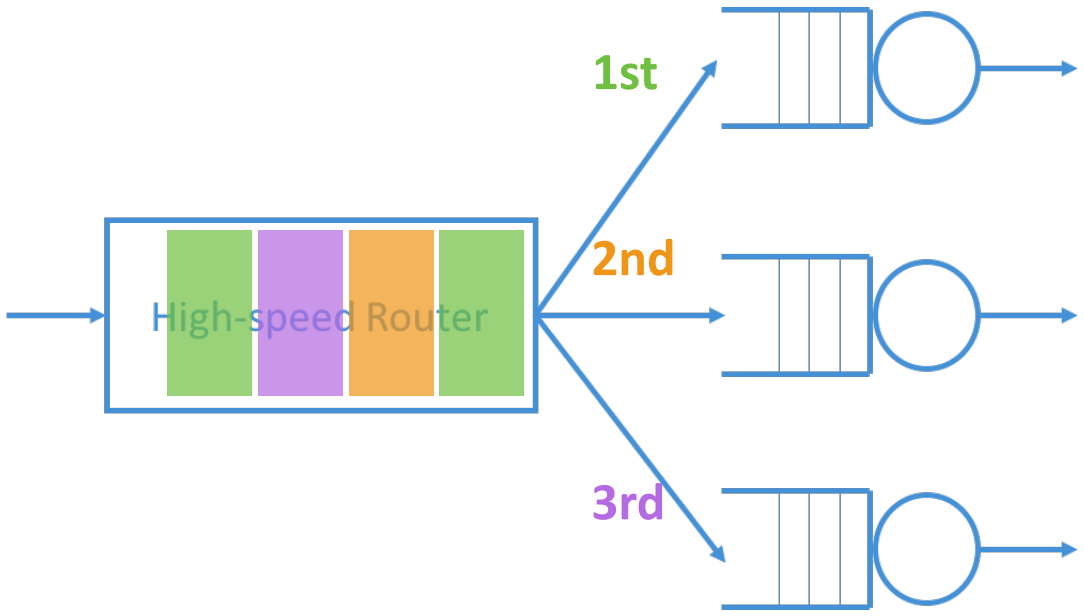


Assignment Policies

2. Round-Robin [Static]

Jobs are assigned to hosts in a cyclical fashion.

The aim is to **equalize the expected number of jobs at each host**.

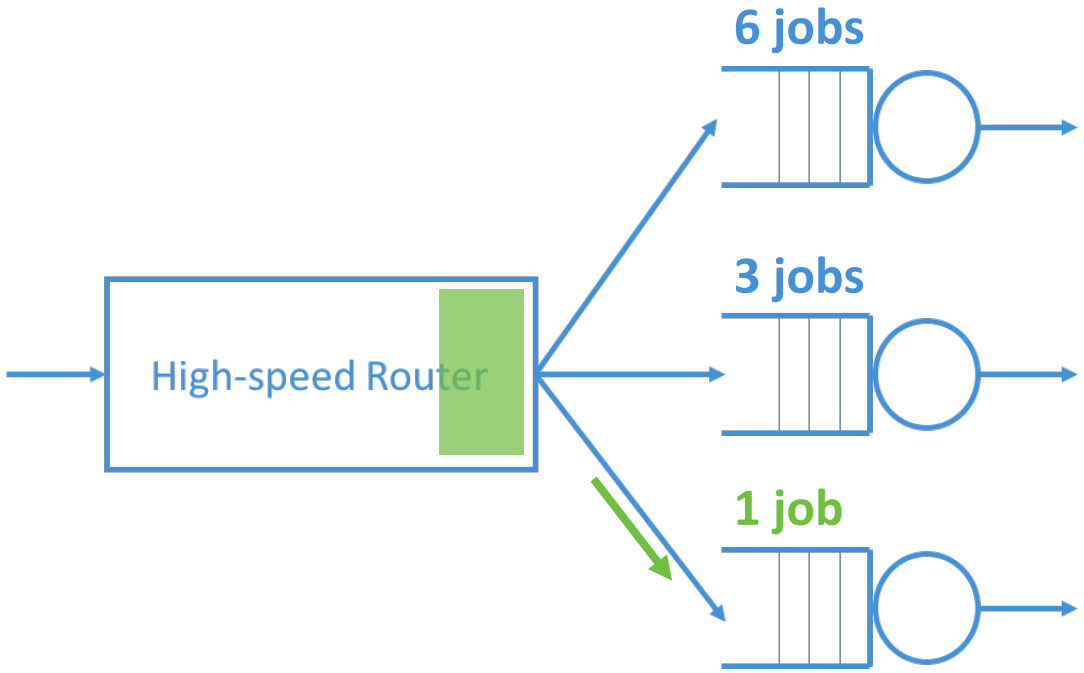


Assignment Policies

3. Join-the-Shortest-Queue (JSQ) [Dynamic]

Each incoming job is assigned to the host that has the shortest queue.

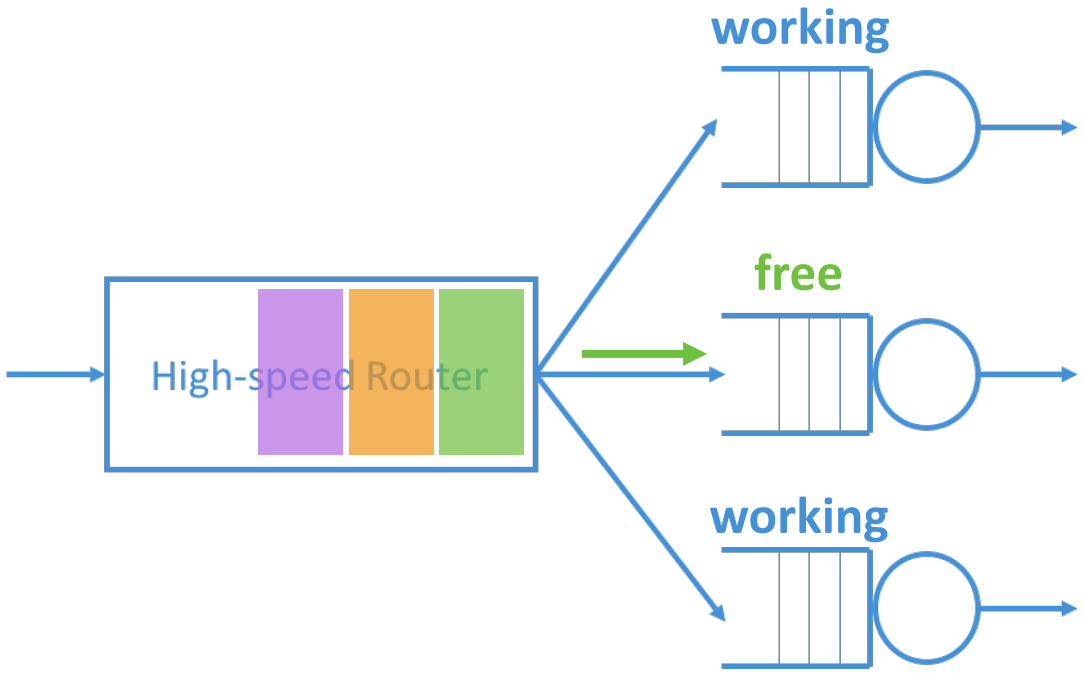
The aim is to **equalize the instantaneous number of jobs at each host**.



Assignment Policies

4. M/G/k [Dynamic]

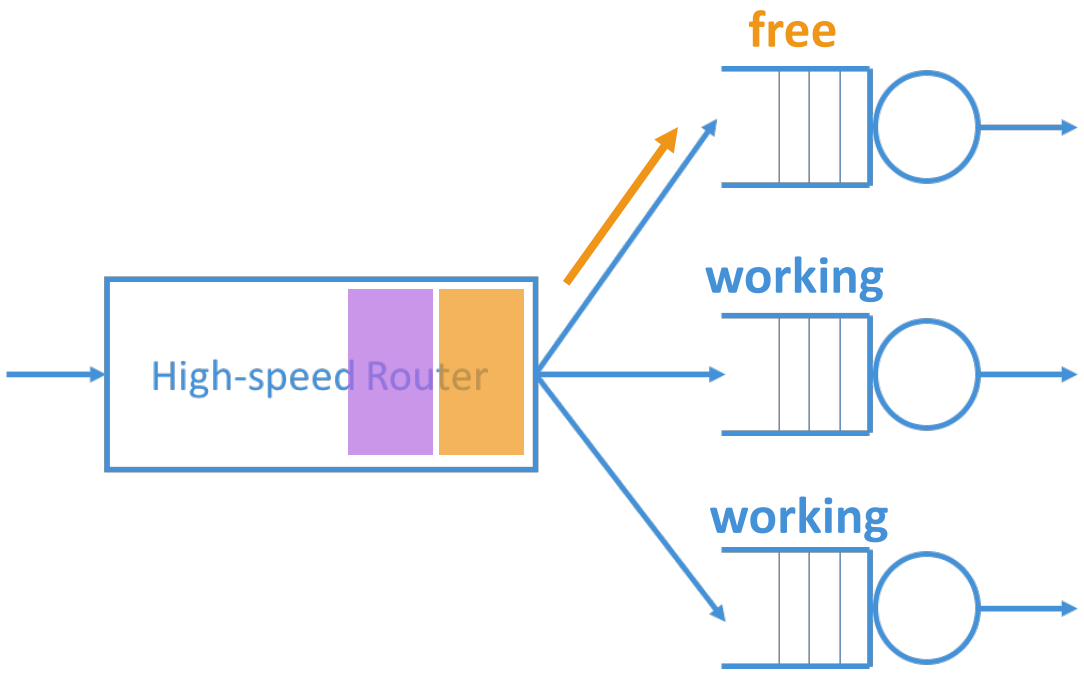
A host, when free, picks the job at the head of the central queue to run.



Assignment Policies

4. M/G/k [Dynamic]

A host, when free, picks the job at the head of the central queue to run.



Exercise: Which policy is best?

1. RANDOM
2. Round-robin
3. Join-the-shortest-queue
4. M/G/k

Q1: RANDOM vs Round-robin?

Q2: Round-robin vs Join-the-shortest queue?

Q3: Join-the-shortest-queue vs. M/G/k?

Assignment Policies

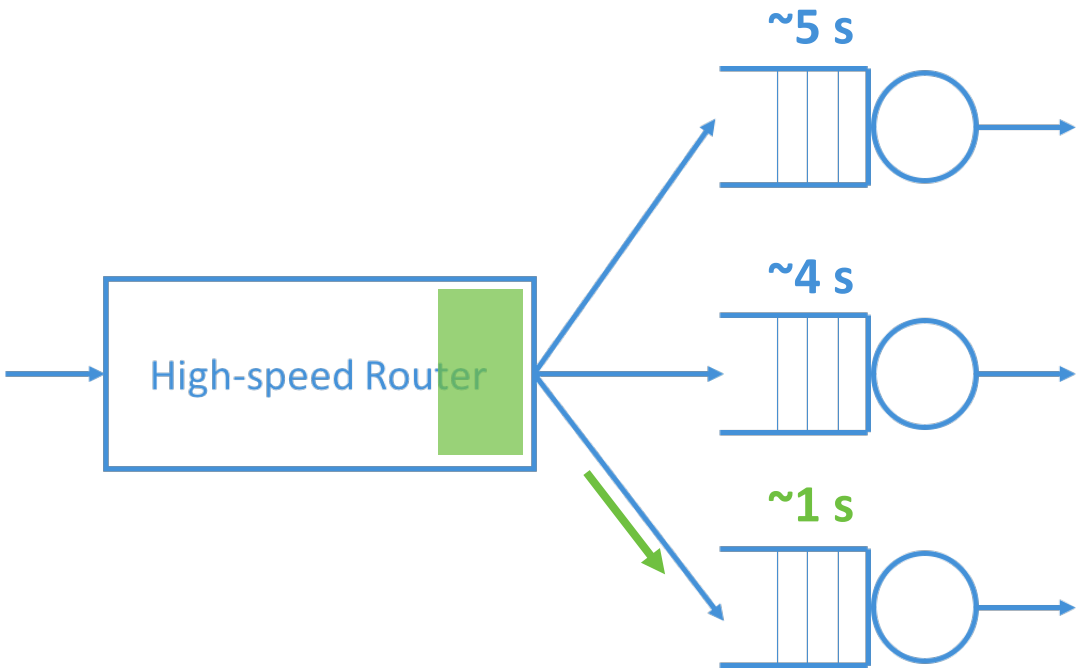
If we know the size of a job

Assignment Policies

If we know the size of a job

5. Least-Work-Left (LWL) [Dynamic]

Each job goes to the queue where it will achieve the lowest possible response time. This is a *greedy* policy. It aims to **equalize the total work at each host**.



Exercise: Which policy is best?

1. Join-the-shortest-queue
2. Least-work-left
3. M/G/k

Q1: Join-the-shortest-queue vs. Least-work-left?

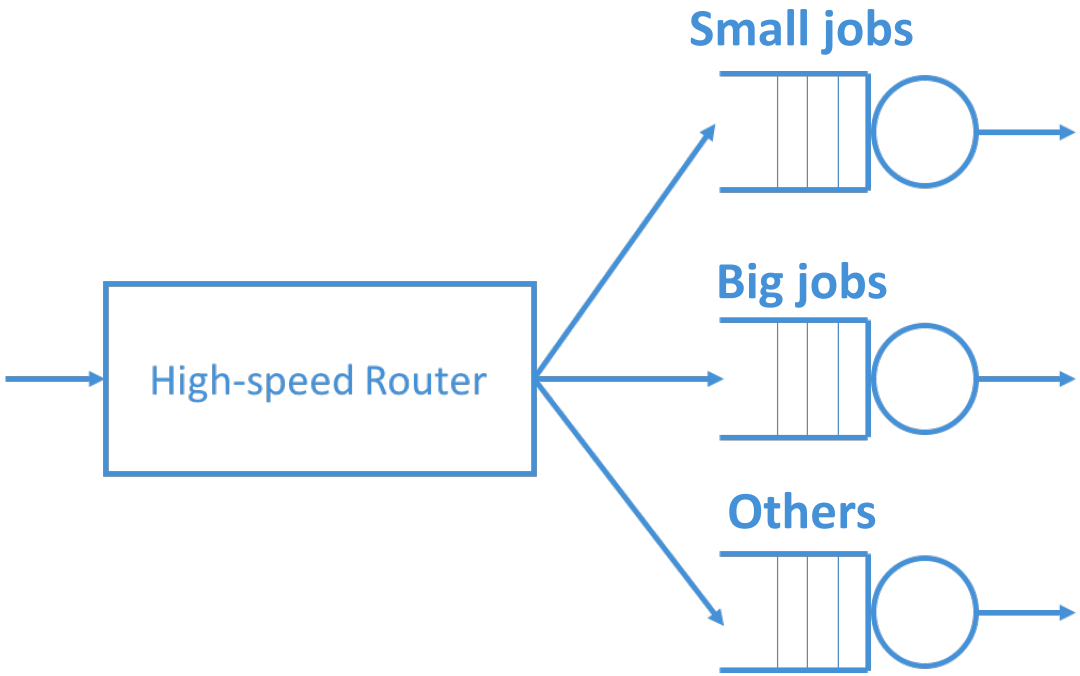
Q2: Least-work-left vs. M/G/k?

Assignment Policies

If we know the size of a job

6. Size-Interval-Task-Assignment (SITA)

Each host is assigned to a size interval, where the size intervals are non-overlapping and span the full range of possible job size.



Assignment Policies

If we know the size of a job

6. Size-Interval-Task-Assignment (SITA)

Each host is assigned to a size interval, where the size intervals are non-overlapping and span the full range of possible job size.

Reducing variability at each queue.

The problem of finding closed-form optimal cutoffs for general job size distributions is still wide open!

Outline

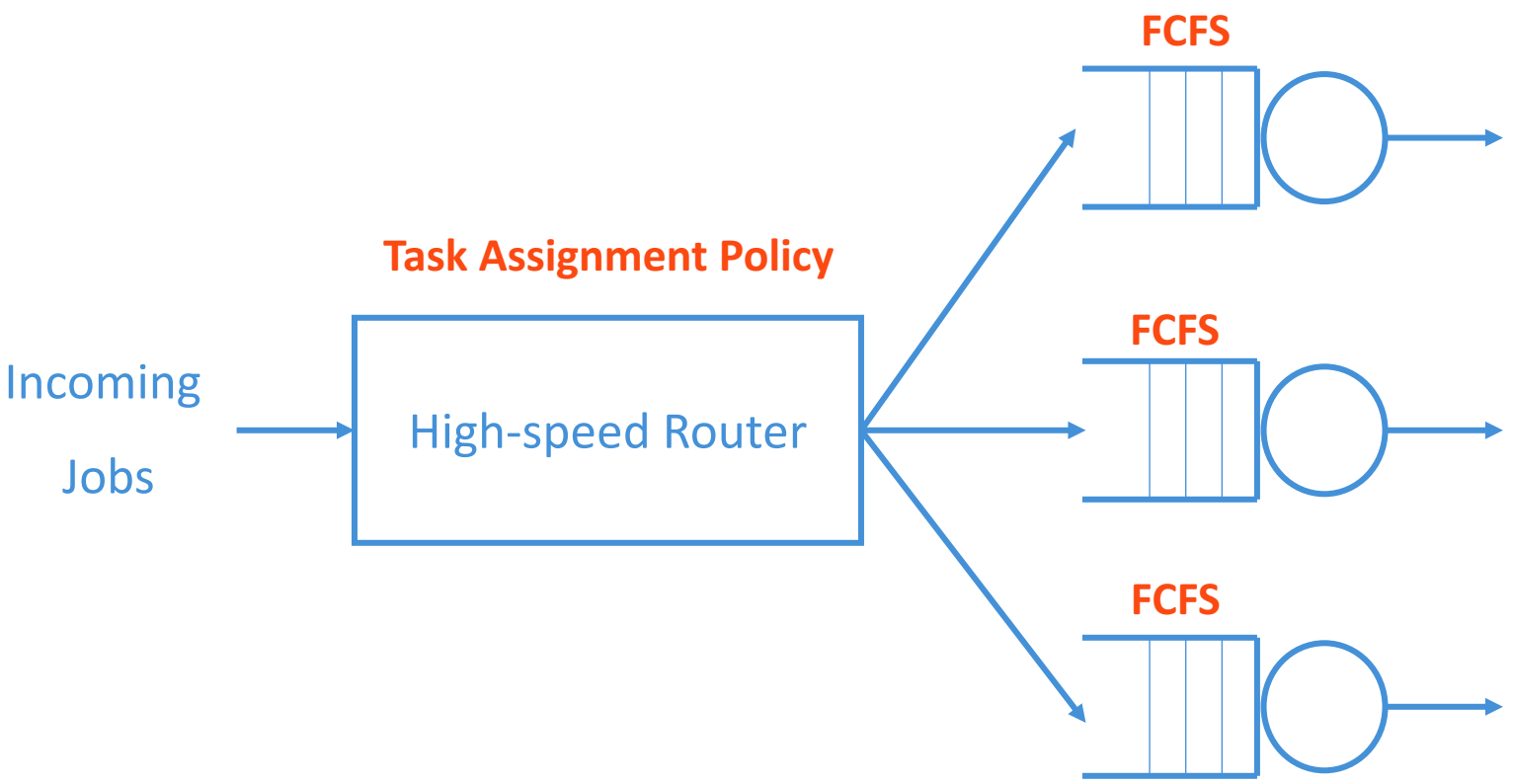
A. Policies Introduction

B. Performance Comparisons

C. Optimal Analysis

D. Discussions

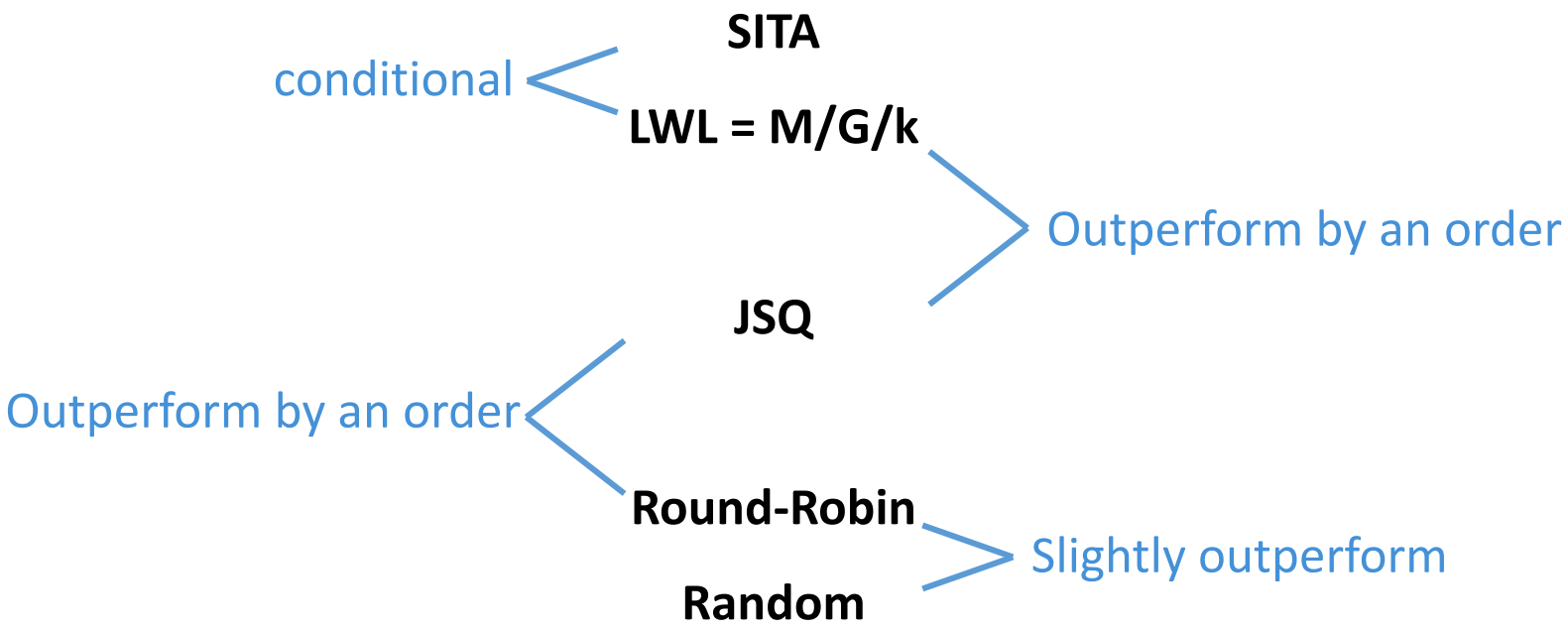
Server Farm Model



Question: Which assignment policy is the best?

Results

Under higher job size variability



Analysis

Under higher job size variability

SITA

LWL = M/G/k

JSQ

Round-Robin

Random

Under RANDOM policy,

the queue behaves like an M/G/1, with average arrival rate λ/k

Under Round-Robin policy,

the queue behaves like an E_k/G/1, with average arrival rate λ/k

Fact: E_k has lower variability, compared to the M (Exponential).

Analysis

Under higher job size variability

SITA

LWL = M/G/k

JSQ

Round-Robin

Random

Precisely analyzing JSQ is difficult. Based on approximations, it seems clear that JSQ is far superior to Round-Robin.

Intuition

When job size variability is high, queues can empty very suddenly. JSQ policy can quickly fill up the empty queue while Round-Robin needs to wait.

JSQ is a *dynamic policy*, which means it adapts based on changes in the state of the system.

Analysis

Under higher job size variability

SITA

LWL = M/G/k

JSQ

Round-Robin

Random

Empirical results show that M/G/k can outperform JSQ by an order of magnitude, with respect to the mean respond time.

Intuition

M/G/k holds off on assigning jobs to hosts as long as possible. The underutilization can never happen under M/G/k, because whenever there are >k jobs, every host is busy.

The analysis of M/G/k is a long-standing open problem!

Analysis

Under higher job size variability

SITA

LWL = M/G/k

JSQ

Round-Robin

Random

Many papers found that as job size variability is increased, SITA becomes far superior to LWL.

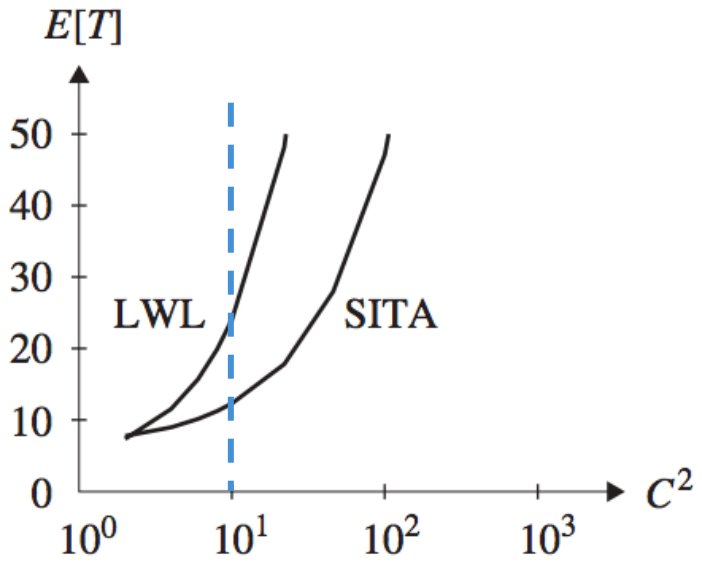


Figure 24.4. Expected response time, $E[T]$, for SITA and LWL versus C^2 in a 2-server system with $R = 1.8$ and job size distribution H_2 with unbalanced branches ($Q = 0.7$).

Analysis

Under higher job size variability

SITA

LWL = M/G/k

JSQ

Round-Robin

Random

Fact: There are cases where SITA is not superior to LWL under high C, and in fact, SITA is provably unboundedly worse than LWL.

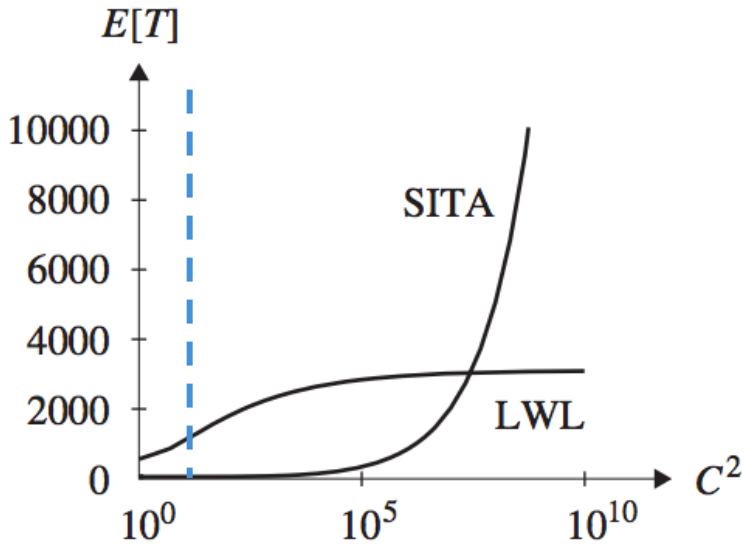


Figure 24.5. Expected response time, $E[T]$, for SITA and LWL versus C^2 in a 2-server system with Bounded Pareto job size distribution with $\alpha = 1.6$ and $R = 0.95$.

Outline

A. Policies Introduction

B. Performance Comparisons

C. Optimal Analysis

D. Discussions

Theoretical Question

How to optimally design a server farm?

if one is allowed to choose both the task assignment policy and the scheduling policy at the individual hosts?

Theoretical Question

There only exist worst-case analysis

One imagines an **adversary** who can generate **any arrival sequence**, where the arrival sequence consists of arrival times of jobs and their sizes.

Then, we **evaluate** our policy **on each possible** arrival sequence and is compared with the **optimal** policy **for that** arrival sequence.

Theoretical Question

There only exist worst-case analysis

Researchers care about

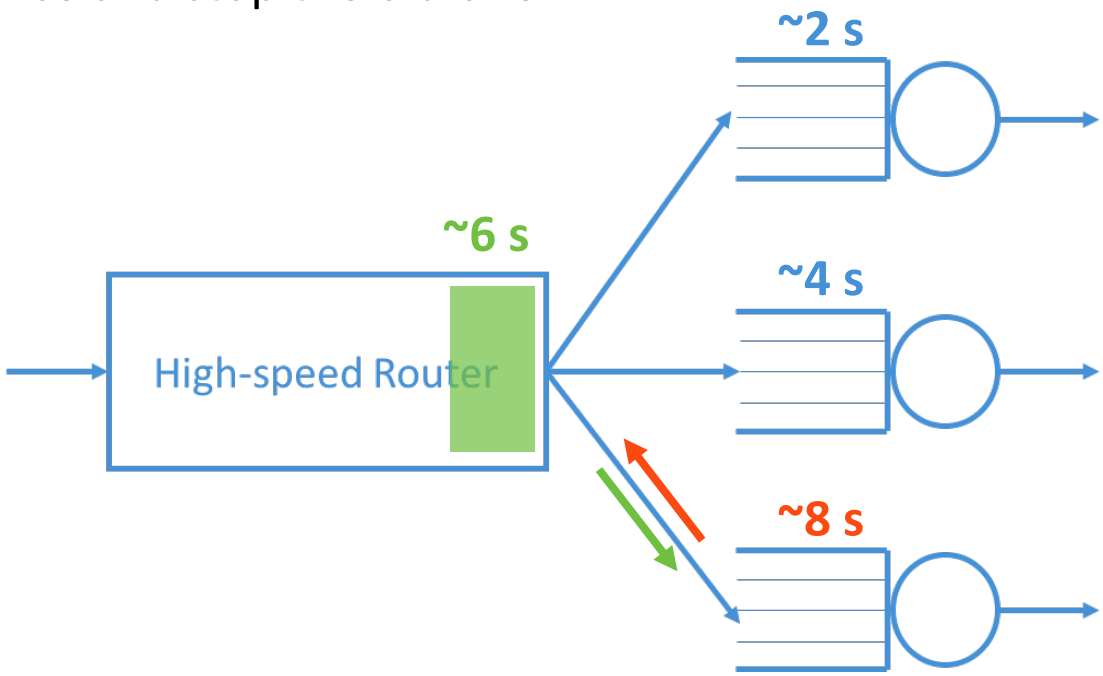
$$\begin{aligned}\text{Competitive ratio of } \mathcal{P} &= \max_{\mathcal{A}} r_{\mathcal{P}}(\mathcal{A}) \\ &= \max_{\mathcal{A}} \frac{\mathbb{E}\left[T(\mathcal{A})\right]^{\mathcal{P}}}{\mathbb{E}\left[T(\mathcal{A})\right]^{\mathrm{OPT}}}\end{aligned}$$

In this metric, a policy can be very poor just because it performs badly on one particular arrival sequence, though that sequence can be a very low-probability event.

Theoretical Question

Central-Queue-Shortest Remaining Process Time (SRPT)

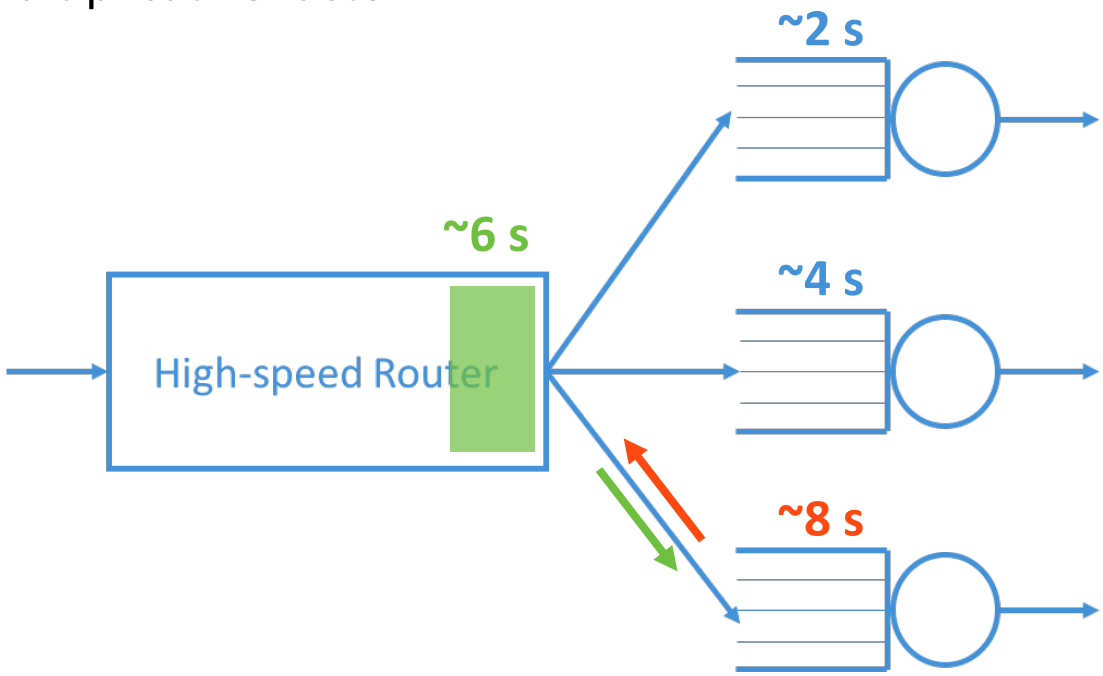
The system evaluates the remaining process time for incoming jobs, if the time is less than the maximal time in one server, that server will immediately put it into service and stop the old one.



Theoretical Question

Central-Queue-SRPT Policy

- **Optimal** with respect to mean response time for a single-server queue
- **No online algorithm can improve its competitive ratio** by more than a constant multiplicative factor.



Outline

A. Policies Introduction

B. Performance Comparisons

C. Optimal Analysis

D. Discussions