

## گزارش پروژه شبیه‌سازی شبکه‌ی صف باز

### مرور کلی

در این گزارش، پیاده‌سازی شبیه‌سازی یک شبکه صف باز ارائه شده است. این شبیه‌سازی با استفاده از زبان برنامه‌نویسی پایتون و با کمک کتابخانه‌ی SimPy پیاده‌سازی شده است. این شبیه‌سازی، سیستمی با سه صف را مدل‌سازی می‌کند که هرکدام دارای زمان سرویس منحصر به فردی هستند. توزیع زمان بین ورود و هم‌چنین زمان سرویس همه‌ی صف‌ها به صورت نمایی در نظر گرفته شده است که رویکردی معمول در نظریه صف برای شبیه‌سازی فرآیندهای تصادفی و بدون حافظه است.

### اجزای کد شبیه‌سازی

#### پیکربندی (Config)

پارامترهای شبیه‌سازی را تعیین می‌کند. این‌ها شامل نرخ زمان بین ورود (inter\_arrival\_rate) و نرخ زمان سرویس (mu\_1, mu\_2, mu\_3) برای هر صف و هم‌چنین زمان کل شبیه‌سازی (simulation\_time) هستند.

```
class Config:
    def __init__(self):
        self.inter_arrival_rate = 1.0
        self.mu_1 = 2.0
        self.mu_2 = 4.0
        self.mu_3 = 3.0
        self.simulation_time = 1000
```

#### صف

هر صف در سیستم را نمایندگی می‌کند. این صف‌ها هم‌چنین تعداد مشتریان خدمت‌رسانی شده، تعداد مشتریان در صف، تأخیر کلی و آمارهای مختلف دیگر را ردیابی می‌کنند. هر صف با استفاده از متد process، مشتریان را پردازش می‌کند که مکانیزم سرویس را شبیه‌سازی می‌کند.

```
class Queue:
    def __init__(self, env, mu, id):
        self.env = env
        self.mu = mu
        self.id = id
        self.server = simpy.Resource(env, capacity=1)
        self.num_served = 0
        self.num_in_queue = 0
        self.total_delay = 0
        self.times_of_arrival = []
        self.times_of_arrival_debug = []
        self.service_times = []
        self.area_under_b = 0 # sum active time
        self.area_under_q = 0 # sum in queue by time
        self.last_event_time = 0
        self.server_status = 0
```

### مشتری

مشتریان را با نرخ‌ی که از توزیع نمایی پیروی می‌کنند، تولید می‌کند. هر مشتری ابتدا توسط صف ۱ پردازش می‌شود، سپس بر اساس یک احتمال به صف ۲ یا ۳ هدایت می‌شود.

```
class Customer:
    def __init__(
        self, env: simpy.Environment, queues: Dict[str, Queue], config: Config
    ):
        self.env = env
        self.queues = queues
        self.config = config
        self.inter_arrival_times = []

    def process(self):
        while True:
            inter_arrival_time = random.expovariate(self.config.inter_arrival_rate)
            self.inter_arrival_times.append(inter_arrival_time)

            yield self.env.timeout(inter_arrival_time)
            self.env.process(self.process_customer())

    def process_customer(self):
        queue = self.queues["1"]
        yield self.env.process(queue.process())

        if random.random() < 0.4:
            next_queue = "2"
        else:
            next_queue = "3"

        yield self.env.process(self.queues[next_queue].process())
```

### شبیه‌سازی

کل فرآیند را هماهنگ می‌کند، صف‌ها را مقداردهی اولیه می‌کند و مشتریان را طی زمان مشخص شبیه‌سازی پردازش می‌کند. در پایان شبیه‌سازی، چندین معیار کلیدی عملکرد محاسبه و نمایش داده می‌شوند.

```
class Simulation:
    def __init__(self, config: Config, env=simpy.Environment()):
        self.env = env
        self.config = config
        self.queues = {
            "1": Queue(self.env, config.mu_1, 1),
            "2": Queue(self.env, config.mu_2, 2),
            "3": Queue(self.env, config.mu_3, 3),
        }

    def run(self):
        c = Customer(self.env, self.queues, self.config)
        self.env.process(c.process())
        self.env.run(until=self.config.simulation_time)
```

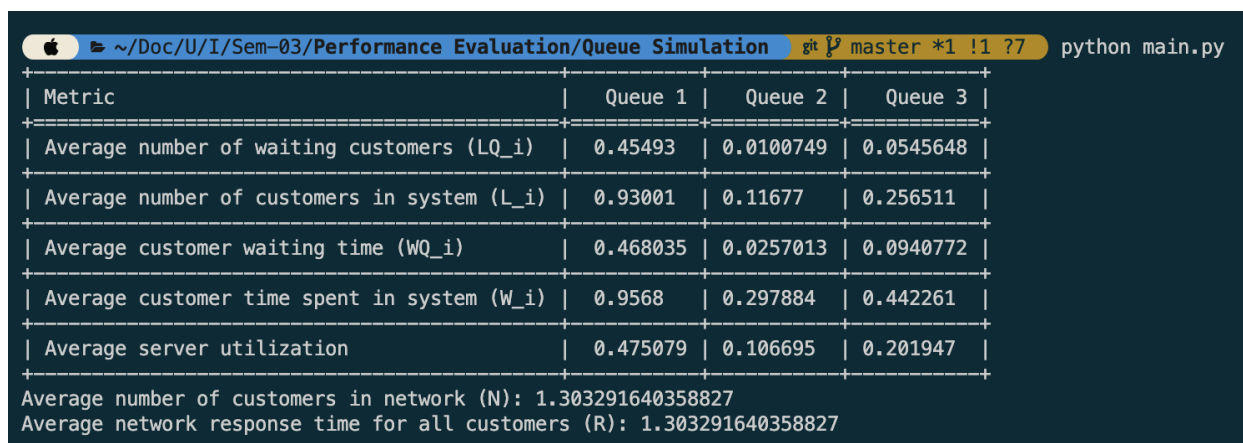
## محاسبه معیارها

در پایان شبیه‌سازی، چندین معیار عملکرد کلیدی به ازای هر صف محاسبه و نمایش داده می‌شوند:

- میانگین تعداد مشتریان منتظر ( $LQ_i$ )
- میانگین تعداد مشتریان در سیستم ( $L_i$ )
- میانگین زمان انتظار مشتری ( $WQ_i$ )
- میانگین زمانی که مشتری در سیستم می‌گذراند ( $W_i$ )
- بهره‌وری سیستم ( $\rho$ )

علاوه بر این، شبیه‌سازی میانگین کل تعداد مشتریان در شبکه ( $N$ ) و میانگین زمان پاسخ شبکه برای همه مشتریان ( $R$ ) را محاسبه می‌کند.

خروجی برنامه‌ی خواسته شده به شکل زیر است:



Metric	Queue 1	Queue 2	Queue 3
Average number of waiting customers ( $LQ_i$ )	0.45493	0.0100749	0.0545648
Average number of customers in system ( $L_i$ )	0.93001	0.11677	0.256511
Average customer waiting time ( $WQ_i$ )	0.468035	0.0257013	0.0940772
Average customer time spent in system ( $W_i$ )	0.9568	0.297884	0.442261
Average server utilization	0.475079	0.106695	0.201947

Average number of customers in network ( $N$ ): 1.303291640358827  
Average network response time for all customers ( $R$ ): 1.303291640358827

## نتیجه‌گیری

این شبیه‌سازی بینش‌های ارزشمندی در مورد دینامیک‌های صف در شبکه‌ای با نرخ‌های زمان سرویس متفاوت ارائه می‌دهد. با تنظیم پارامترها در کلاس Config، مدل را می‌توان برای سناریوهای مختلف سازگار کرد که آن را به ابزاری انعطاف‌پذیر برای تحلیل و بهینه‌سازی سیستم‌های صف تبدیل می‌کند. استفاده از SimPy، یک محیط شبیه‌سازی انعطاف‌پذیر و کارآمد را فراهم می‌کند که برای هر دو منظور آموزشی و حرفه‌ای در درک سیستم‌های صف پیچیده مناسب است.