گزارش پروژه درس شبیه‌سازی کامپیوتری

دکتر بردیا صفایی

بهار ۱۴۰۲

تهیه کنندگان:

محمدمهدی قیدی ۹۸۱۰۵۹۷۶

زهرا رحمانی ۹۹۱۷۰۴۳۴

مقدمه

در این گزارش به شرح اجرا و پیاده‌سازی پروژه پایانی درس شبیه‌سازی خواهیم پرداخت. در مستند پروژه از ما خواسته شده بود که سناریویی از یک شبکه کامپیوتری، تشکیل شده از یک روتر و دو دیوایس پیاده‌سازی کنیم که یکی از آن‌ها پکت‌های مختلفی را ارسال میکند که روتر دریافت کرده و روی آن‌ها پردازش انجام می‌دهد. ما برای پیاده‌سازی این پروژه از زبان پایتون استفاده کردیم و سعی کردیم کدی روان و خوانا بنویسیم تا مطالعه و بررسی آن برای خواننده آسان باشد. این داک هم در کنار کد به عنوان مستندی خواهد بود که این امر را تسهیل خواهد کرد.

معماری و ساختار کد

ما در پیاده‌سازی و نوشتن کد پروژه از رویکرد Object-Oriented بهره بردیم و هر کدام از ماهیت‌های موجود در شبکه را با استفاده از یک کلاس پیاده‌سازی کردیم. برای مثال پکت، روتر، پراسسور همگی آبجکت هستند که می‌توانند به طرز خاصی کار انجام بدهند. مثلا کلاس پکت بدین صورت تعریف شده است:

```
class Packet:
 def __init__(self, arrival_time, priority, process_time):
  self.arrival_time = arrival_time
  self.priority = priority
  self.process_time = process_time * 10
  self.queue_time = 0
 def __str__(self):
  return f'Arrival {self.arrival_time}, Priority {self.priority}, Process time {self.process_time}'
```

چنانچه در کد هم میتوانیم ببینیم، هر پکت دارای زمانی برای پردازش، یک اولویت و یک arrival time است که زمانی که از host مبدا خود خارج شده و به سمت روتر می‌رود را نشان می‌دهد. همچنین هر پکت یک queue time نیز دارد که نشان دهنده این است که این پکت چه مقدار در صف روتر معطل شده تا به پردازنده/پراسسور برسد.

برای پیاده‌سازی روتر، پارامترهایی که در اجرای کار یک روتر، مثل صف، حداکثر طول صف و پراسسورهای موجود در روتر کاربرد دارند در کلاس تعریف شده اند. همچنین توابعی که کارهای مورد نیاز ما در یک روتر را هندل میکنند اضافه شده‌اند.

برای پراسسورها متغیرهایی مثل مجموع زمان مورد استفاده بودن پردازنده و وضعیت فعلی‌شان در کلاس تعبیه شده اند.

تولید داده

همانطور که در داک اصلی پروژه نوشته شده بود، تعداد پکت‌های ارسالی توسط host مبدا از توزیع پوآسون پیروی می‌کند. همانطور که از درس میدانستیم، بین توزیع پوآسون و توزیع نمایی با پارامتر لامبدا، یک رابطه برقرار می‌باشد. بدین صورت که اگر تعداد رخدادهای یک رویداد در یک بازه زمانی از توزیع پوآسون پیروی بکند، فاصله‌های زمانی بین رخدادها(interarrival times) از توزیع نمایی پیروی خواهد کرد. در اینجا ما از این فکت استفاده کردیم و برای تولید زمان‌های arrival time هر پکت، که در واقع نشان دهنده زمان ارسالشان از host مبدا به روتر هستند، با استفاده از پارامتر X که در بالای کد تعریف کردیم و می‌تواند در هر experiment متغیر باشد، به تولید یک interarrival تایم پرداختیم. بدین صورت تا زمانی که مجموع این interarrival time ها از زمانی که برای شبیه‌سازی (T) اختصاص دادیم بزرگتر نشود پکت‌های تولید شده با interarrival تایم برآمده از توزیع نمایی را میپذیریم.

همچنین در داک گفته شده بود که زمان پردازش هر پکت توسط روتر به صورت تصادفی و از توزیع نمایی با پارامتر Y انتخاب خواهد شد. که این رفتار هم در زمان تولید داده‌ها شبیه‌سازی شد و به هر پکت یک زمان مورد نیاز برای پردازش اختصاص داده شد.

اولویت هر کدام از پکت‌ها هم به کمک تابع رندوم که موجود در StdLib خود پایتون است و با قرار دادن در محدوده احتمالاتی ارائه شده توسط مستند پروژه انتخاب شد.

اجرای گذر زمان و شبیه‌سازی عملکرد شبکه

پس از آن که مقادیر و دیتای مورد نیاز برای شبیه‌سازی را تولید کردیم، به شبیه‌سازی گذر زمان پرداختیم. برای این منظور واحد زمانی را به ۱۰۰۰ قسمت تقسیم کردیم. مثلا فرض کنید میخواستیم در زمان T=100 روند کار شبکه را

شبیه‌سازی کنیم. کاری که انجام می‌دهیم بدین صورت است که یک حلقه خواهیم داشت که با گام های یک هزارم واحد زمانی و به صورت یک هزارم یک هزارم جلو خواهد رفت.
همچنین در هر لحظه‌ای از زمان چک خواهیم کرد و ببینم که آیا event ای در این لحظه رخ خواهد داد یا خیر. مثلا در این لحظه آیا پکتی وارد روتر خواهد شد؟ آیا کار یکی از پردازنده‌های روتر در این لحظه تمام می‌شود و آزاد شود؟ آیا پکتی در صف روتر ممکن است وارد پردازنده شود و کارش شروع شود؟ و تمامی این اتفاقات را بررسی خواهیم کرد. مثلا ممکن است یک پکت arrival time ای برابر ۶.۳۰۷ داشته باشد. در این صورت هنگامی که متغیر حلقه به این عدد برسد، پکت از host مبدا dispatch شده و به سمت روتر می‌رود و داخل روتر می‌شود. همچنین تمام عملیات‌هایی مثل گرفته شدن یک پکت و پراسس شدن آن و ... را به صورت توابعی در کلاس Router درون کد قرار دادیم تا این موارد را در هر مرحله مدیریت کند.

جمع‌آوری و مدیریت داده‌های شبکه

همانطور که از هدف پروژه نیز مشخص است، نیاز به اندازه‌گیری تعدادی معیار و مدیریت آن‌ها داشتیم. بدین منظور متغیرهایی را برای اندازه گیری طول صف روتر، اندازه گیری میانگین زمان صرف شده در صف‌های روتر، میزان بهره‌وری هر کدام از پردازنده‌ها، تعداد پکت‌های دراپ شده و ... در کلاس‌های روتر و پراسسور و پکت گنجاندیم. در هر واحد زمانی، با توجه به وضعیت موجود در شبکه، این داده‌ها را آپدیت میکنیم. مثلا تابع update_router_queue_stats برای به روز رسانی مقادیر مربوط به طول صف‌های روتر در هر iteration استفاده می‌شود و در نهایت مقادیری که داشتیم را برای گزارش در خروجی چاپ میکنیم.
پارامترهای خواسته شده بدین صورت بودند:

- میانگین طول صف‌ها
- میانگین زمان صرف شده در تمامی صف‌ها
- میانگین زمان صرف شده در هر یک از صف‌ها
- میانگین بهره‌وری هر یک از پردازنده‌ها
- تعداد تمامی بسته‌های drop شده

- نمودار CDF مربوط به مدت زمان صرف شده در تمامی صف‌ها برای packet های با اولویت بالا(high priority)

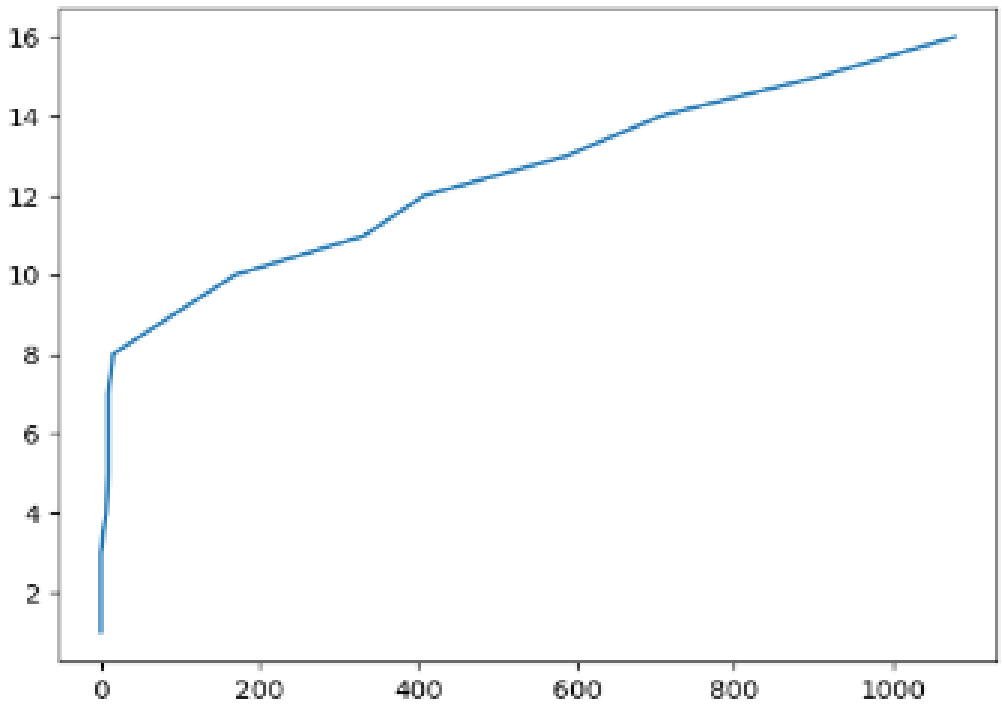
تمامی موارد بالا در کد محاسبه شده‌اند و با هر بار اجرای کد تمامی اطلاعات و داده‌های مربوط به آن اجرا در خروجی چاپ خواهند شد.

برای مثال یک نمونه از اجرای شبیه‌سازی با پارامترهای زیر را در عکس نشان داده‌ایم:

- PROCESSORS_NUM = 4
- X = 10
- Y = 5
- T = 1000
- HIGH_PRIORITY_QUEUE_LENGTH_LIMIT = 4
- MID_PRIORITY_QUEUE_LENGTH_LIMIT = 6
- LOW_PRIORITY_QUEUE_LENGTH_LIMIT = 10
- SERVICE_POLICY = WRR

```
cafebazaar@cafebazaar:~/univ/Computer_Simulation/network_project$ python3.10 simulator.py
Serving policy of the router:  ServicePolicyTypes.WRR
Total simulation time:  1000
Total number of packets sent in the network:  105
Number of processed packets:  84
Number of dropped packet:  21
Processor 1 was utilized in 937.55: 93.75 percent
Processor 2 was utilized in 931.31: 93.13 percent
Processor 3 was utilized in 981.48: 98.15 percent
Processor 4 was utilized in 926.49: 92.65 percent
All processors utilization:  94.42075
Avg length of the high priority queue:  1.71
Avg length of the mid  priority queue:  3.99
Avg length of the low  priority queue:  6.04
Avg length of all router queues:  3.91
Avg time spent in the high priority queue:  114.24
Avg time spent in the mid  priority queue:  231.01
Avg time spent in the low  priority queue:  120.41
Avg time spent in all queues:  134.89
```

همانطور که میبینیم در این شبیه‌سازی تمام اطلاعات خواسته شده قرار گرفته و همچنین نمودار CDF پکت‌های با اولویت بالا نیز در زیر آورده شده است(کد به صورتی نوشته شده که پس از اجرای آن فایل عکس حاوی این نمودار در کنار کد قرار خواهد گرفت).



همچنین ۲ سوال برای نتیجه‌گیری از شبیه‌سازی‌های اجرا شده پرسیده شده که آن‌ها را در زیر پاسخ خواهیم داد:

1. چطور می‌توان بهره‌وری سیستم را افزایش داد؟

برای افزایش بهره‌وری سیستم می‌توان با توجه به میزان بهره‌وری هر کدام از پردازنده‌ها، تصمیم گرفت که آیا بهتر است تعداد پردازنده‌ها را افزایش داد یا خیر. مثلا اگر تعداد پکت های دراپ شده بالا باشد و همچنین بهره‌وری تعداد بالایی از پردازنده‌ها روی ۱۰۰ درصد باشد، این یعنی سیستم در حال عمل روی فشار بالایی است و بهتر است چند پردازنده برای کمک به سیستم اضافه شود. پس از افزایش این پردازنده‌ها تعداد کمتری پکت‌های دراپ شده را شاهد خواهیم بود. همچنین لود بین پردازنده‌ها پخش شده و پکت‌های بیشتری پردازش می‌شوند که باعث افزایش بهره وری سیستم‌خواهد بود.

همچنین از طرفی اگر بهره‌وری پردازنده‌ها میزان پایینی است میتوان تعدادی از آن‌ها را کم کرد تا هزینه و انرژی بیهوده صرف نشود چرا که در این شرایط تعداد کمتری پردازنده می‌توانند کار مورد نیاز لود روی سیستم را انجام بدهند.

2. با توجه به داده‌های بدست آمده از شبیه‌سازی و با در نظر گرفتن تعداد پکت‌های دراپ شده و تعداد پکت‌های رسیده به مقصد، استفاده از کدام سیاست‌های نوبت‌دهی منطقی‌تر است و مناسب‌تر است که از آن به عنوان سیاست نوبت‌دهی استفاده کنیم؟

با توجه به مقادیر dropped packets که از اجرای شبیه‌سازی با پارامترهای مختلف دیده شد، به نظر می‌آید پالیسی Weighted Round Robin بازدهی بهتر و تعداد پکت‌های لاس شده کمتری را ایجاد خواهد کرد. در این حالت وزن صف‌های الگوریتم، با میزان احتمال رخداد اولویت پکت‌ها تنظیم شده است و همین امر باعث می‌شود به طور کلی و در long run عملکرد خوبی نشان دهد.

البته باید این موضوع را مد نظر داشت که اگر burst های بزرگ از پکت‌هایی با یک اولویت داشته باشیم این متود عملکرد ضعیفی نشان خواهد داد.

به طور کلی و در یک نتیجه گیری صحیح تر می‌توان گفت که متود مورد استفاده ما باید منطبق بر نوع کاربردی که از شبکه داریم باشد. اگر یک شبکه معمولی با پکت‌هایی که از احتمالات بیان شده و اولویت‌هایشان پیروی می‌کنند داشته باشیم WRR انتخاب مناسبی خواهد بود.

اما اگر ترافیک با رفتار پیک زدن و burst شکل داشته باشیم احتمالا FIFO انتخاب مناسب تری خواهد بود.

همچنین اگر میخواهیم به بعضی مشتری‌ها و بعضی پکت‌ها اولویت بدهیم و شبکه را مثلا در یک بیمارستان داریم طراحی میکنیم که پکت‌های حیاتی با اولویت بالاتر ارسال می‌شوند و نیاز به پردازش سریعتر آنان هست از non-preemptive priority scheduling استفاده خواهیم کرد که پکت‌های با اولویت بالاتر را زودتر مورد پردازش قرار دهند.