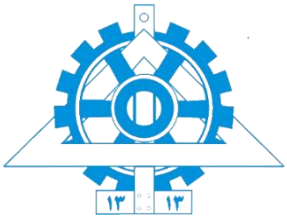


# تمرین برنامه نویسی شماره ۳

**عنوان:** پیاده سازی شبیه ساز شبکه برای بررسی الگوریتم های مسیریابی

**درس:** شبکه‌های کامپیوتری

**استاد راهنما:** دکتر ناصر یزدانی

**رشته:** مهندسی کامپیوتر

**دستیاران آموزشی:** سهراب مرادی[۱]، حسام رمضانیان[۲]، علی دارابی[۳]، امیرعلی وحیدی[۴]

**نیمسال اول سال تحصیلی ۰۴-۱۴۰۳**

[۱] نشانی پست الکترونیکی: m.moradi1998@ut.ac.ir

[۲] نشانی پست الکترونیکی: hesamhrf@gmail.com

[۳] نشانی پست الکترونیکی: adlidarabi9981@gmail.com

[۴] نشانی پست الکترونیکی: amir.ali.vahidi@ut.ac.ir

# فهرست



جدول تصاویر

## عنوان پروژه

**پیاده سازی شبیه ساز شبکه برای بررسی الگوریتم های مسیریابی**

## هدف

در تمرین ۲ شما با استفاده از یک Emulator با سطح Abstraction بالا اقدام به طراحی و پیاده سازی یک شبکه کامپیوتری ساده نمودید، در این تمرین بنا داریم تا در ادامه ی تمرین ۲، با پیاده سازی سطح پایین با سطح Abstraction کمتر یک شبیه ساز شبکه، با جزئیات بیشتری از شبکه های کامپیوتری آشنا شویم. بر این مبنا قصد داریم تا با استفاده از فریمورک Qt به پیاده سازی یک شبیه ساز رویداد محور (Event Driven) بپردازیم تا با استفاده از آن پاسخگویی الگوریتم های مسیریابی را در شرایط مختلف بررسی کنیم، همچنین با ساختار روترها و فرآیند های موجود در آنها، برخی از توپولوژی های شبکه و توزیع های آماری تولید داده در دنیای واقعی آشنا خواهید شد.
پس از پیاده سازی این تمرین، شما به سطح دانش بالا و دید عمیقی از شبکه های کامپیوتری دست پیدا خواهید کرد.

# ابزار و توضیح بخش های صورت تمرین

این تمرین تماما با استفاده از زبان برنامه نویسی C++ پیاده سازی خواهد شد لذا از کتابخانه ها و فریمورک های مطرح و محبوب این زبان استفاده خواهیم کرد. ابزار مورد نیاز به شرح زیر است:

- Qt framework

- ✓ در بخش ۱، مقدمه ای راجع به اهمیت مباحث پوشش داده شده در تمرین گفته شده است.
- ✓ در بخش ۲، تمرین را به ۶ بخش کلی شکسته و تا حد امکان به توضیح هر بخش و نکات آنها پرداخته ایم.
- ✓ بخش ۳ توضیحاتی است راجع به پیاده سازی تمرین و نکات مربوط به کد نویسی آن.
- ✓ بخش ۴ به جهت سنکرون شدن پیاده سازی ما و شما، توضیحات و تعاریف ما از مواردیست که قرار است در تمرین پیاده سازی شوند، انتظارات ما از الگوریتم ها و پروتکل ها، ویژگی ها روتر و توپولوژی ها.
- ✓ در بخش ۵ موارد امتیازی تمرین گردآوری شده است و برخی از آنها توضیح داده شده اند.
- ✓ بخش ۶ شامل فاز های اجرای شبیه ساز است.
- ✓ در بخش ۷ مواردی که باید پس از اجرای شبیه ساز آنها را بررسی کنید ذکر شده اند.
- ✓ بخش ۸ فاز های تحویل پروژه هستند، این پروژه به جهت کمک به شما فازبندی شده و هر فاز ددلاین جداگانه ای دارد.

## ۱- مقدمه

- **اهمیت الگوریتم های مسیریابی**

الگوریتم های مسیریابی سنگ بنای ارتباطات شبکه هستند که کارآمدترین مسیر را برای انتقال بسته های داده درسراسر شبکه از مبدا به مقصد تعیین می کنند. اهمیت این الگوریتم ها در توانایی آنها برای بهینه سازی عملکرد شبکه با کاهش تأخیر، متعادل کردن بار ترافیک و جلوگیری از از دست دادن بسته ها نهفته است. آنها برای حصول اطمینان از اینکه شبکه می‌تواند مقیاس رو به رشدی از کاربران و دستگاه‌ها را بدون کاهش کیفیت خدمات در خود جای دهد، ضروری هستند.

- **اهمیت توپولوژی**

توپولوژی ستون فقرات (backbone) یک شبکه به زیرساخت مرکزی اشاره دارد که بخش های مختلف شبکه را مانند ستون فقرات در بدن انسان به هم متصل می کند و انتخاب توپولوژی مناسب برای زیرساخت شبکه بسیار مهم است زیرا بخش عمده ای از ترافیک داده را مدیریت می کند و به عنوان مسیر اصلی برای ارتباطات بین شبکه ای عمل می کند. توپولوژی زیر ساخت با طراحی خوب، انتقال داده با سرعت بالا، قابلیت اطمینان و تحمل خطا را تضمین می کند. همچنین مدیریت شبکه را ساده می کند و با افزایش تقاضای شبکه امکان گسترش آسان تر را فراهم می کند.

- **شبیه سازی شبکه های تحت بار سنگین**

شبیه سازی شبکه یک عمل حیاتی برای نظارت بر الگوریتم های مسیریابی، به ویژه در شرایط بار سنگین شبکه است. با ایجاد یک مدل مجازی از یک شبکه، مهندسان شبکه می توانند رفتار الگوریتم های مسیریابی را بدون ریسک و هزینه آزمایش بر روی یک شبکه واقعی تجزیه و تحلیل کنند. شبیه سازی امکان مشاهده نحوه عملکرد الگوریتم ها در شرایط استرس (ترافیک بالا)، شناسایی تنگناهای بالقوه و ارزیابی استراتژی های مسیریابی مختلف را فراهم می کند. این عمل برای توسعه شبکه‌های قوی که می‌توانند عملکرد و قابلیت اطمینان را حتی در صورت مواجهه با حجم ترافیک بالا حفظ کنند، ضروری است.

- **توزیع های آماری در شبیه سازی شبکه**

توزیع های آماری نقش محوری در شبیه سازی شبکه ایفا می کنند زیرا به مدل سازی تصادفی و متغیر بودن ترافیک شبکه کمک می کنند. انواع مختلف ترافیک، مانند مرور وب، پخش ویدئو، یا تماس های VoIP، دارای الگوهای متمایزی هستند که می توانند با استفاده از توزیع های آماری مناسب نمایش داده شوند. برای مثال، زمان‌های بین ورود و اندازه بسته‌ها را می‌توان با استفاده از توزیع‌های نمایی یا پارتو مدل‌سازی کرد تا ماهیت انفجاری ترافیک شبکه های واقعی را منعکس کند. دخیل کردن توزیع های آماری در شبیه سازی شبکه اجازه می دهد تا نمایش دقیق تری از رفتار شبکه در شرایط بار مختلف ارائه شود.

- **ویژگی های روتر در شبیه سازی شبکه**

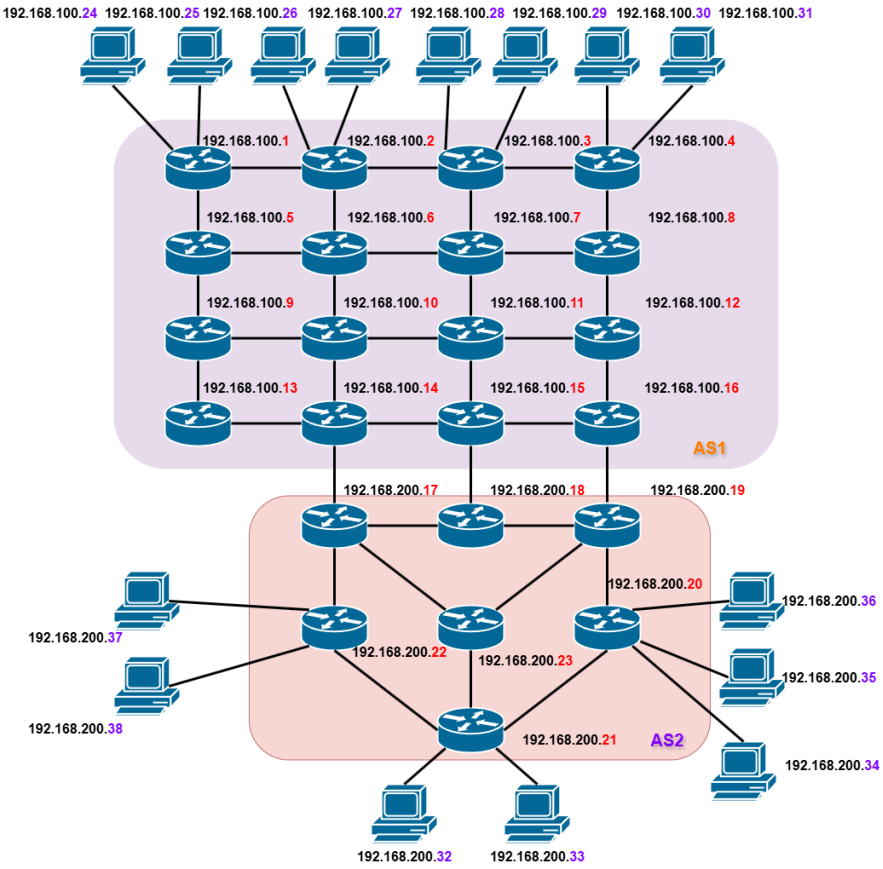
ویژگی های روتر در شبیه سازی شبکه نیز به همانند توزیع های آماری مهم هستند. ویژگی‌هایی مانند اندازه بافر، سیاست‌های مدیریت صف و پروتکل‌های مسیریابی (مانند OSPF یا RIP) مستقیماً بر نحوه مدیریت بسته‌ها توسط روترها تأثیر می‌گذارند، مخصوصاً تحت بار سنگین. شبیه‌سازی روترها با پیکربندی‌های واقعی، تجزیه و تحلیل عملکرد شبکه، شناسایی تنگناهای بالقوه و آزمایش الگوریتم‌ها یا سیاست‌های مسیریابی جدید را ممکن می‌سازد.

- **اهمیت توزیع های آماری در شبیه سازی**

ترکیب توزیع های آماری و ویژگی های روتر در شبیه سازی های شبکه به چند دلیل حیاتی است:

۱) تضمین می کند که شبیه سازی رفتار شبکه های واقعی را از نزدیک منعکس می کند.
۲) به محققان و دانشجویان اجازه می دهد تا تأثیر الگوهای مختلف ترافیک را بر عملکرد شبکه مشاهده کنند.
۳) به ارزیابی اثربخشی الگوریتم‌های مسیریابی و پیکربندی‌های مسیریاب تحت شرایط استرس، مانند زمان اوج استفاده یا حملات DDOS کمک می‌کند.
۴) با در نظر گرفتن این عوامل، شبیه سازی ها می توانند بینش های ارزشمندی را در مورد طراحی و مدیریت شبکه های قوی و کارآمد ارائه دهند.

## ۲- توضیح تمرین




*شکل ۱ - ساختار شبکه ی مورد بحث تمرین*

حل این تمرین نیازمنده پیاده سازی ۶ بخش مختلف (به انضمام کلاس های کمکی) است:

۱- **سیستم کنترل رویداد و داده:** این سیستم باید در بازه های زمانی قابل تنظیم (از طریق فایل کانفیگ، به عنوان مثال هر ۱۰ ثانیه) تولید سیگنال کند (یک سیگنال را در سطح برنامه emit کند)، از این سیگنال برای هماهنگی بخش های مختلف برنامه استفاده می شود. تولید داده ها بر اساس سیگنال های تولید شده از این ماژول کنترل خواهند شد، به این صورت که این سیستم با توجه به توزیع انتخابی، تعداد بسته هایی که در هر سیکل باید تولید شوند را به PC ها اعلام میکند. همچنین این ماژول کل مدت زمان شبیه سازی را نیز مدیریت خواهد کرد. این سیستم شامل دو کلاس DataGenerator و EventCoordinator میشود که در واقع EventCoordinator شامل یک instance از DataGenerator خواهد بود. توضیحات مربوط به توزیع های اماری و نحوه ی انتخاب آن در ادامه ی صورت تمرین آمده است.

**نکات مهم:**

- مقصد هر بسته میتواند به انتخاب شما، حتی به صورت رندوم تعیین شود اما توجه داشته باشید که بسته ای از شخص A برای خودش ارسال نمیشود!
- کلاس های دخیل: DataGenerator و EventCoordinator

۲- **روتر:** موجودیت روتر که عمل مسیریابی در شبکه را انجام میدهد. روتر ها دارای ویژگی های زیرخواهند بود:
۱- جدول مسیریابی (بر اساس نوع الگوریتم مورد استفاده).
۲- پورت ها.
۳- ورژن IP. (IP ورژن ۴ اجباری و IP ورژن ۶ امتیازی)
۴- بافر با اندازه ی محدود.
۵- DHCP.
**۶-** پروتکل BGP. (امتیازی)

**در دنیای واقعی روتر ها شامل ویژگی های بسیاری هستند که در این تمرین ما تا حد امکان ساده ترین نسخه ی روتر با برخی مفروضات غیر واقعی که پاسخگو نیاز تمرین باشد را انتخاب نموده ایم.**
به عنوان مثال روتر های شبکه به ازای هر NIC یک IP دارند اما در این تمرین هر روتر به عنوان یک موجودیت فقط یک IP دارد.

**نکات مهم:**

- هر روتر یک عدد منحصر به فرد به عنوان ID دارد که ID ها را ما بر روی شکل مشخص کرده ایم.
- تمامی روتر ها دارای DHCP سرور هستند اما این قابلیت در هر AS (Autonomous System) فقط برای یک روتر باید فعال باشد.
- هر DHCP Server هنگام تخصیص IP به هر Node شبکه، باید ID و IP و MACAddress آن Node به انضمام شماره ی AS مربوطه را در یک فایل بنویسد. این فایل جهت صحت عملکرد DHCP بررسی خواهد شد.
- کلاس های دخیل: OSPF, RIP, Router, DHCPServer, Port, Node, BGP.
- هر روتر دارای یک IP منحصر به فرد است.
- هر روتر یک بافر مشترک برای تمامی پورت های خود دارد.
- تمامی روتر ها دارای ۴ پورت هستند.
- برخی از روتر ها به عنوان as_gateway و برخی به عنوان user_gateway تعیین میشوند.
- تمامی روتر ها از کلاس Node ارث بری میکنند.
- میتوان هر تعداد کاربر را به یک پورت روتر متصل کرد اما این اتصال برای روتر ها یک به یک است، یعنی به هر پورت یک روتر، فقط یک روتر دیگر میتواند متصل شود.

- پیاده سازی BGP امتیازی است.

- پیاده سازی BGP بنا به تشخیص شما و با هر سطح پیچیدگی قابل قبول است اما نمره دهی بر اساس کمیت و کیفیت پیاده سازی صورت میگیرد.

- روتر ها در هر سیکل فقط میتوانند یک بسته ها به ازای هر پورت ارسال کنند، اگر روتری در بافر خود بسته هایی داشت که میتوانستند از پورت های مختلف ارسال شوند، میتواند بسته ها را از پورت های مختلف در یک سیکل ارسال کند ولی فقط یک بسته به ازای هر پورت! به عنوان مثال از ۶ بسته در بافر روتر وجود داشت و تمام بسته ها باید از پورت ۱ ارسال میشدند، در این حالت روتر در هر سیکل فقط ی بسته را میتواند ارسال کند ولی اگر ۴ بسته داشت و این ۴ بسته به ترتیب از پورت های ۱ تا ۴ میتوانستند ارسال شوند، روتر میتواند در یک سیکل هر ۴ بسته را ارسال کند.

- بسته های داده و بسته های کنترلی هردو در یک بافر مشترک قرار میگیرند، برای مدیریت بسته های داده از FIFO استفاده کنید و هنگامی که بافر پر بود، بسته ای که اکنون رسیده را drop کنید ( dropped flag را true کنید و بسته را در بافر ذخیره نکنید)

- اگر بافر پر بود و بسته ی کنترلی به روتر رسید، این بسته اولویت بالایی دارد، آخرین بسته ی داده در صف را drop کنید (حذف از لیست و true کردن dropped flag) و بسته ی کنترلی را در ابتدای صفر insert کنید.

- ***مهم*** ID شماره ی روتر است که ما برای سادگی در اتصال آنها و تشکیل توپولوژی ها و AS ها بر روی تصاویر تمرین مشخص کرده ایم، در واقعیت این ID وجود ندارد و صرفا برای ساده سازی مرحله ی صفر اجرای برنامه است (برای مشاهده ی جزئیات مرحله های اجرای برنامه به بخش ۶ مراجعه کنید). این ID متفاوت از MACAddress است و برای توصیف و شبیه سازی دقیق تر از شرایط واقعی، نیاز داریم تا هر روتر یک MACAddress داشته باشد که این MAC Address ها در حین اجرای فاز صفر شبیه ساز تولید و به روتر ها assign میشوند! درحالی که ID ها را ما برای هر نود شبکه برای شما از قبل مشخص کرده ایم!

## ۳- موجودیت Packet

در شبکه انواع مختلفی از بسته ها وجود دارند که ما در این تمرین، بسته ها را به دو دسته ی کلی بسته های کنترلی و بسته های داده تقسیم بندی میکنیم.
بسته های داده بسته هایی هستنده که کاربران در شبکه برای یکدیگر ارسال میکنند.
بسته های کنترلی بسته هایی هستند که برای مدیریت شبکه از طرف روتر ها برای یکدیگر ارسال میشوند و هرکدام برای هدف خاصی استفاده میشوند.
بسته های کنترلی در این تمرین شامل موارد زیر هستند:

۱- بسته های DHCP که شامل Discovery, Offer, Request و Ack هستند.
۲- بسته های مربوط به الگوریتم های مسیریابی.
۳- بسته های مربوط به BGP.
۴- بسته های مورد نیاز شما در پیاده سازی خودتان.

هر بسته شامل بخش های مختلفی است، این بخش ها شامل:

۱- IPHeader: برای IP ورژن ۴ و ۶.
۲- BGPHeader: برای بسته های BGP.
۳- TCPHeader
۴- DataLinkHeader
۵- Payload
۶- sequenceNumber
۷- waitingCycles: تعداد سیکل هایی که در صف منتظر بوده
۸- totalCycles: کل سیکل هایی که طول کشیده تا بسته به مقصد برسد.
۹- Path: آی پی هاپ هایی که طی کرده.
۱۰- و....

دقت داشته باشید که در صورت پیاده سازی IP ورژن ۶، بسته ها حین انتقال از روتر ورژن ۶ به روتر ورژن ۴ باید encapsulate شوند. (تغییر هدر از ورژن ۶ به ۴ و ...)

**نکات مهم:**

- کلاس های دخیل: Packet, IP و انواع Header

## ۴- موجودیت AS (Autonomous System)

در پیاده سازی backbone شبکه، ما از دو AS استفاده میکنیم تا بتوانیم الگوریتم های مسیریابی خارج از شبکه ی محلی را نیز پوشش بدهیم، هر AS شامل تعدادی روتر است (تعداد روتر ها جهت پیاده سازی در شکل ۱ مشخص شده است) که این روتر ها در قالب یک توپولوژی خاص به یکدیگر متصل شده است، همچنین روتر های مشخصی نیز باید به عنوان درگاه این AS معرفی شوند. یکی از AS ها از توپولوژی Mesh و دیگری از توپولوژی Ring-Star (یک توپولوژی ابداعی، ترکیبی از توپولوژی Ring و Star) استفاده خواهد کرد (توضیحات توپولوژی ها + شکل در بخش تعاریف تمرین وجود دارند). برای ساده سازی شرایط تمرین، طول لینک ها در ارتباطات میان روتر ها (چه داخلی و چه خارجی) یکسان در نظر گرفته خواهد شد. لازم است تا شبکه ی Mesh شامل ۱۶ روتر و شبکه ی Ring-Star شامل ۷ روتر باشد.

**نکات مهم:**

- هر AS به تعداد تعریف شده روتر و PC میسازد (تعریف در فایل کانفیگ)، ابتدا روتر ها را مطابق با توپولوژی در نظر گرفته شده به یک دیگر متصل کرده و سپس PC ها را به Gateway ها (مطابق با فایل کانفیگ) متصل میکند.
- کلاس های دخیل: AS, Router, TopologyBuilder, TopologyController, Port, PortBindingManager, PC
- هر AS مطابق با فایل کانفیگ یک روتر را به عنوان DHCP Server میسازد (ویژگی DHCP روتر مورد نظر را enable میکند)
- هر AS بر اساس کانفیگ gateway های خود با کاربران و دیگر AS ها را میشناسد.

۵- **موجودیت PC:** این موجودیت در واقع کاربران شبکه هستند که در هماهنگی با سیستم تولید رویداد و داده، اقدام به تولید بسته های داده و ارسال آنها به کاربران مقصد میکنند.

**نکات مهم:**

- در هر سیکل به تعدادی که سیستم کنترل رویداد بر اساس توزیع آماری مشخص میکند بسته ساخته و ارسال میکند.
- مقصد هر بسته میتواند به انتخاب شما، حتی به صورت رندوم تعیین شود اما توجه داشته باشید که بسته ای از شخص A برای خودش ارسال نمیشود!
- مانند روتر ها، PC ها نیز از کلاس Node ارث بری میکنند.
- هر کاربر یک پورت دارد که از طریق آن برای gateway دیتا ارسال میکند.
- هر کاربر IP تمامی کاربر های دیگر شبکه را دارد.

۶- **موجودیت Network:** این موجودیت AS ها را در خود نگه میدارد و آنها را به یک دیگر متصل میکند (از طریق gateway ها و متناظر با فایل کانفیگ).

۷- **موجودیت Simulator:** این موجودیت وظیفه ی مدیریت فاز های اجرای برنامه را بر عهده دارد (توضیحات مربوط به فاز ها و دستورات فاز ۴ اجرای برنامه در ۶ ذکر شده اند)

ممکن است شما بسته به سبک پیاده سازی خودتان نیاز به بخش های دیگری نیز داشته باشید، بنابراین لطفا این بخش های جدید را در گزارش خود مفصل شرح دهید.

## ۳- برنامه نویسی و نکات پیاده سازی

کد شما باید ساختارمند و تماما ماژولار باشد، برای پیاده سازی سیستم کنترل رویداد از Signal-Slot System در Qt استفاده کنید. لازم هست تا تمامی کلاس های شما مستقیم یا غیرمستقیم از QObject ارث بری کنند. برای شبیه سازی عملکرد موازی سیستم (بسته ها نباید به ترتیب در شبکه حرکت کنند، زیرا در این صورت هیچگاه کارکرد واقعی شبکه شبیه سازی نخواهد شد) هر روتر را باید در Thread مخصوص به خودش اجرا کنید، برای این منظور دو راه پیش روی شماست:

۱- میتوانید از تابع moveToThread در QObject استفاده کنید، به این شکل که به تعداد روتر ها از QThread یک instance بسازید و هر روتر را به یک Thread منتقل کنید.
۲- میتوانید در هنگام پیاده سازی روتر از QThread ارث بری کنید؛ توجه داشته باشید که هر روتر باید از کلاس Node ارث بری کند پس در واقع Node باید از QThread ارث بری کند.(ارث بری غیر مستقیم)

نکته ۱: توجه داشته باشید که تمامی توابعی که برای کلاس روتر پیاده سازی میکنید باید Reentrancy داشته باشند و برخی توابع نیز نیاز است تا Thread-Safe باشند، همچنین امکان استفاده از Qt Concurrent و دیگر ابزار های Qt که کمک به شبیه سازی حالت موازی می کنند مجاز است. برای ساده سازی تمرین، استفاده از تمامی کتابخانه های فریمورک Qt مجاز است و میتوانید از هر کتابخانه ای که فرآیند حل تمرین را برای شما ساده تر کند استفاده کنید.

نکته ۲: به مانند کلاس روتر، کلاس PC و EventCoordinator نیز باید در Thread مخصوص به خود اجرا شوند.

نکته ۳: روش پیشنهادی ما برای پیاده سازی، روش دوم یعنی ارث بری از QThread است، در این روش میتوانید کلاس Node را به صورت یک زیرکلاس از QThread تعریف کنید.

نکته ۴: در صورت استفاده از روش ۲، میتوانید به طریقی پیاده سازی را انجام دهید که PC و Router هردو از Node ارث بری کنند. (منطبق بر پیاده سازی دستیاران آموزشی)

در پیاده سازی شبیه ساز های شبکه مبحث Memory Management بسیار مهم است، برای انتقال بسته ها میان روتر ها، یک بسته در سیستم تولید داده شود و یک QSharedPointer از آن بین روتر ها و ترد های مختلف در جریان باشد (تا از کپی ها ناخواسته جلوگیری شود). همچنین توجه شود که instance های QSharedPointer درقالب const reference بین توابع مختلف پاس داده شوند.

پیشنهاد میشود تا روابط Object Trees & Ownership میان QObject ها مطالعه شود تا به مدیریت بهتر حافظه کمک کند.

برای پیاده سازی راحت تر توپولوژی ها، میتوانید از روی شکل ۱ روتر ها را شماره گذاری کنید و با یک شمارنده ی static به هر نود یک id اختصاص داده و از روی شماره گذاری خودتان مشخص کنید که هر روتر به کدام روتر های دیگر متصل است. در شکل ۱ عدد اکتت آخر هر IP نشان دهنده ی شماره روتر است.

در روتر تابع printRoutingTable را پیاده سازی کنید، همچنین این قابلیت باید وجود داشته باشید که بعد از اتمام فرآیند شبیه سازی، با انتخاب هر روتر دلخواهی، بتوانید Routing Table آن را مشاهده کنیم.

از Hard Code کردن مقادیر و اعداد و ... در کد جدا پرهیز کنید، قطعا در نمره ی شما تاثیر خواهد داشت.

نیاز است تا ویژگی های روتر همانند آنچه در بخش تعاریف آمده است پیاده شود، برای ساده سازی تمرین ویژگی ها در ساده ترین سطح شان تعریف شده اند.

**جریان پیاده سازی:**

ابتدا به سراغ بخش های کوچک تر تمرین بروید، به عنوان مثال در ابتدا کلاس MACAddress را پیاده سازی کنید. ما برای تولید MACAddress های یکتا، کلاسی به نام MACAddressGenerator نیز پیاده سازی کردیم که پیشنهاد میکنیم شما نیز این کار را انجام دهید.

در ادامه به سراغ کلاس هایی بروید که کمترین نیاز را به بخش های دیگر تمرین دارند، مثلا بعد از پیاده سازی مک، به سراغ پیاده سازی IP بروید. این کلاس نیز مانند MACAddress پیش نیازی ندارد.

در پیاده سازی کلاس IP نهایت دقت را داشته باشید، به افرادی که مایلند تا Ipv۶ را نیز به عنوان بخش امتیازی پیاده سازی کنند، پیشنهاد میشود تا یک Abstract Class به عنوان پایه ی IP داشته باشند و کلاس های IPV۴ و IPV۶ را با ارث بری از این کلاس پیاده سازی کنند.

یک سبک پیاده سازی پیشنهادی برای کلاس IP در اختیار شما قرار گرفته است.

سپس به سراغ EventCoordinator بروید، این کلاس یکی از مهم ترین کلاس های برنامه است و در پیاده سازی آن وسواس بالایی داشه باشید. تفکر رویداد محور و پیش بینی شما از فرآیندی که قرار است در برنامه اتفاق بیوفتد، در پیاده سازی هرچه بهتر این کلاس کمک خواهند کرد.

برای زمان بندی و شبیه سازی کلاک از std::chrono::milliseconds یا Qchrono استفاده کنید، این متغییر های زمان با کلاس Qtimer سازگار هستند.

بهتر است تا این کلاس را Singleton پیاده سازی کنید.

در هماهنگی با کلاس EventCoordinator کلاس DataGenerator را پیاده سازی کنید. این کلاس تعداد داده هایی که در هر سیکل براساس توزیع آماری تولید خواهد شد را حساب میکند و EventCoordinato از طریق سیگنال nextTick آن را به PC ها اطلاع میدهد.

در ادامه میتوانید به سراغ کلاس Packet بروید و آن را به همراه انواع Header های مورد نیاز پیاده سازی کنید.

در فایل Globals.h ما Enum Class هایی را در اختیار شما قرار داده ایم که میتوانید از آنها در پیاده سازی خود استفاده کنید.

سپس به سراغ کلاس های Node و Router بروید و طبق توضیحات قبلی آنها را پیاده سازی کنید، ترجیح ما بر این است که الگوریتم های مسیریابی مانند OSPF , RIP و پروتکل های شبیه مانند BGP را نیز هر کدام را در یک کلاس پیاده سازی کنید و در روتر استفاده کنید.

برای پیاده سازی DHCP Server نیز اکنون ابزار لازم برای پیاده سازی این کلاس را در اختیار دارند.

در انتها کلاس های AS, PC, Network و باقی کلاس هایی که فکر میکنید جای خالیشان حس میشود را پیاده سازی کنید.

ممکن است در پیاده سازی خود راه بهتر را این برگزینید که از template class ها استفاده کنید، دقت کنید که این نوع از کلاس ها با ماکرو Q_OBJECT سازگاری ندارند، به همچین جهت برای پیاده سازی این دست کلاس ها، ابتدا یک کلاس پایه که از QObject ارث بری میکند و ماکرو Q_OBJECT را نیز دارد پیاده سازی کنید، سیگنال های مدنظر

خود را در این کلاس تعریف کنید، سپس از این کلاس در template class خود ارث بری کنید، به این شکل میتوانید قابلیت های Q_OBJECT و template class ها را ترکیب کنید.

نکته ی دیگر اینکه برای تمامی پوینتر هایی که میسازید (منظور چیز هایی مثل بسته ها و پوینتر هایی که قرار است در جریان برنامه دست به دست شوند) از QSharedPointer استفاده کنید، گاهی لازم است تا پوینتر ها را از base class ها بسازید و در جریان برنامه آنها را cast کنید که برای این منظور میتوانید از توابع مربوط به cast در QSharedPointer استفاده کنید. داکیومنتیشن QSharedPointer

لازم است تا شما یک کلاس به عنوان ApplicationContext داشته باشید تا بتوانید به آن در سراسر برنامه دسترسی داشته باشید که پیشنهاد ما این است که این کلاس singleton باشد تا در نقطه ار برنامه بتوانید به یک instance مشترک دسترسی پیدا کنید.

در این کلاس state های مختلف برنامه، فاز های مختلف شبیه سازی (مانند فاز شناسایی شبکه و ساخت Routing Table ها و تخصیص IP ها و ...) را در آن نگهداری کنید.

همچنین کانفیگ شبیه ساز که از فایل خوانده میشود را نیر در این کلاس به صورت ساختارمند ذخیره کنید تا به سادگی بتوانید از آن استفاده کنید.

برای دوستانی که کمتر با CPP آشنایی دارند: اگر مجبور به استفاده از template function ها شدید، این توابع را inline و در بدنه کلاس پیاده سازی کنید، پیاده سازی انها را به فایل cpp منتقل نکنید.

همواره سعی کنید تا دیزاین شما متناظر با دیزاین پیشنهادی Qt باشد اما اگر جایی مجبور به استفاده از دیزاین های متفاوت بودید حتما از ابزار های استاندارد CPP استفاده کنید، به عنوان مثال اگر جایی نیاز شد تا تابعی را به عنوان callback پاس بدید یا ذخیره کنید حتما از std::function استفاده کنید.

ورودی های template ها را verify کنید، با استفاده از concept ها در استاندارد CPP20 این مورد میسر خواهد شد.

در جریان پیاده سازی برای عدم گمراهی همواره از دستیاران آموزشی سوال بپرسید و نظرات آنان را درباره نحوه ی پیاده سازی و تفکر سیستمی جویا شوید.

# ۴- تعاریف

**۱- تعریف توزیع پوآسون:**

توزیع پواسون (Poisson Distribution) یکی از توزیع‌های احتمال گسسته است که برای مدل‌سازی تعداد رخدادهای یک رویداد در یک بازه زمانی یا مکانی مشخص استفاده می‌شود، به شرطی که این رخدادها مستقل از هم باشند و با نرخ ثابتی رخ دهند. این توزیع معمولاً در سیستم‌هایی استفاده می‌شود که رخدادها به صورت تصادفی اما با نرخ مشخصی رخ می‌دهند.

$$P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad k = 0, 1, 2, \ldots$$

Where:

- $k$ is the number of events (a non-negative integer).
- $\lambda$ is the **rate parameter** (average number of events in the interval).
- $e$ is Euler's number ($e \approx 2.71828$).

**۲- تعریف توزیع پارتو:**

توزیع پارتو (Pareto Distribution) یک توزیع احتمال پیوسته است که برای مدل‌سازی داده‌هایی استفاده می‌شود که در آن مقدار کمی از داده‌ها بخش بزرگی از کل را تشکیل می‌دهند. ویژگی اصلی آن این است که مقادیر بزرگ‌تر با احتمال کمتری رخ می‌دهند.

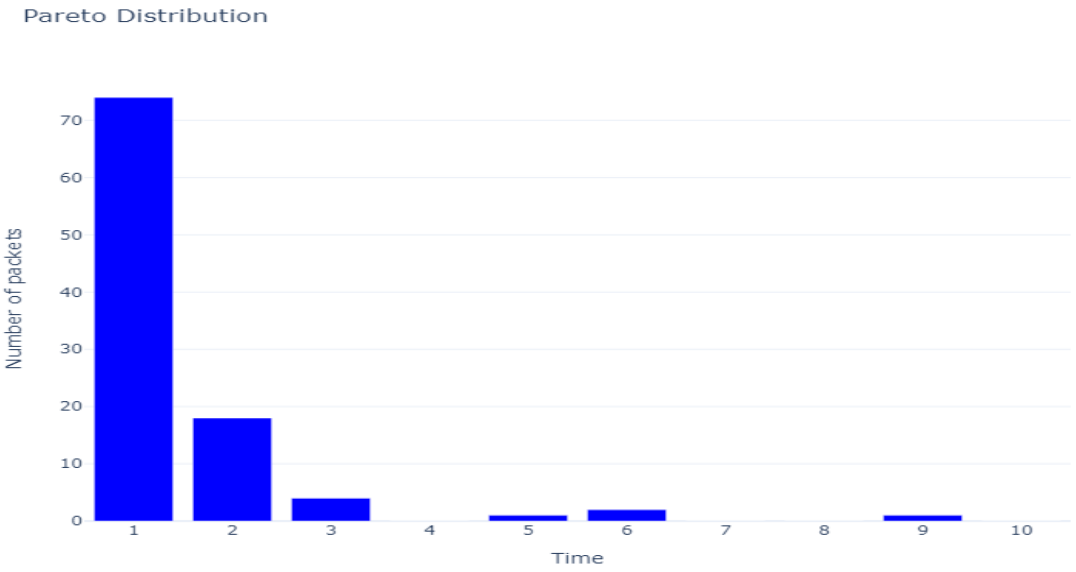
## Probability Density Function (PDF):

$$f(x) = \begin{cases} \frac{\alpha x_m^\alpha}{x^{\alpha+1}} & \text{if } x \geq x_m, \\ 0 & \text{if } x < x_m. \end{cases}$$

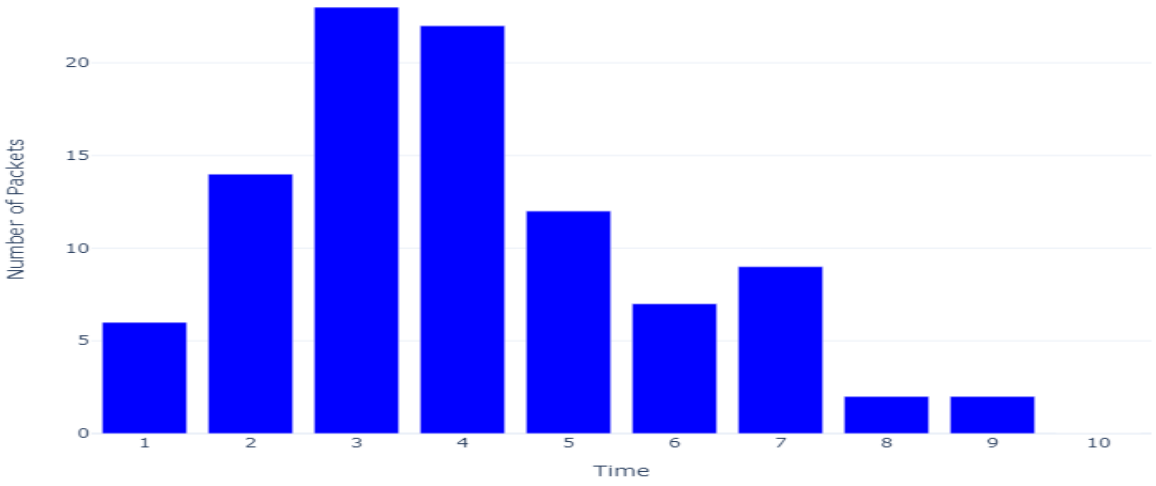
## Parameters:

- $x_m$ (scale): Minimum possible value of $X$.
- $\alpha$ (shape): Determines the "steepness" or tail heaviness of the distribution.

مثال: یک نمونه از توزیع ارسال پکت ها در شبکه بر اساس توزیع پارتو و پواسون برای ۱۰ واحد زمانی شبیه‌سازی به شکل زیر است:




*شکل 2 - توزیع پارتو*




*شکل 3 - توزیع پوآسون*

## ۳- ویژگی های روتر:

### جدول روتینگ:

جدول روتینگ یکی از مؤلفه‌های اصلی در هر روتر است که به مدیریت ترافیک شبکه کمک می‌کند. جدول روتینگ داده‌ها و اطلاعاتی را در خود ذخیره می‌کند که برای تعیین بهترین مناسب ترین مسیر برای ارسال بسته‌های داده به مقصد خاصی استفاده می‌شود. در این جدول، اطلاعاتی که وجود دارد شامل موارد زیر است:

- **آدرس شبکه مقصد (Destination Network Address)**: این آدرس نشان‌دهنده شبکه‌ای است که بسته‌ها باید به آن فرستاده شوند. معمولاً به صورت آدرس IP و ماسک شبکه نمایش داده می‌شود.
- **ماسک شبکه (Subnet Mask)**: این ماسک کمک می‌کند تا روتر بتواند تشخیص دهد که کدام بخش از آدرس IP مربوط به شبکه است و کدام بخش مربوط به میزبان داخل آن شبکه.
- **دروازه (Gateway)**: دروازه یا همانNext Hop ، آدرسی است که بسته‌ها باید ابتدا به آن فرستاده شوند تا از آنجا به مقصد نهایی هدایت شوند. این آدرس معمولاً آدرس IP روتری است که بسته‌ها باید ابتدا به آن بروند.
- **متریک (Metric)**: متریک یک عدد است که برای تعیین "هزینه" یا "فاصله" استفاده می‌شود تا بهترین مسیر به مقصد تعیین شود.
- **پروتکل (Protocol)**: نشان می‌دهد که این مسیر از طریق کدام پروتکل مسیریابی در جدول روتینگ قرار گرفته است. این می‌تواند شامل پروتکل‌های مسیریابی مانندRIP ، OSPF یا BGP باشد.

زمانی که یک روتر بسته‌ای را دریافت می‌کند، ابتدا آدرس مقصد بسته را بررسی می‌کند و سپس با مقایسه آن با موارد ذخیره‌شده در جدول روتینگ، بهترین مسیر را برای ارسال بسته تعیین می‌کند. این فرآیند شامل بررسی تمام مسیرهای ممکن و انتخاب مسیری است که کمترین متریک را دارد یا بر اساس سیاست‌های تعیین‌شده دیگر انتخاب می‌شود.

## الگوریتم های مسیریابی:

**Comparison Between RIP and OSPF**

| RIP (Routing Information Protocol) | OSPF (Open Shortest Path First) |
|---|---|
| Uses the Bellman-Ford algorithm for path calculation. | Uses the Dijkstra algorithm for path calculation. |
| Maintains only a distance vector for routes. | Maintains a complete topology of the network. |
| Simple to implement but less efficient for large networks. | More complex but highly efficient in large and dynamic networks. |
| Routing updates are shared with neighbors. | LSAs (Link-State Advertisements) are flooded throughout the network. |
| Limited scalability due to hop count restriction (maximum 15). | No hop count restriction; suitable for larger networks. |
| Periodic updates increase bandwidth usage. | Updates are triggered by changes, reducing bandwidth usage. |
| Convergence is slower as updates propagate gradually. | Convergence is faster due to immediate flooding of LSAs. |

*شکل ۴ - مقایسه ی اجمالی OSPF و RIP*

## الگوریتم RIP: (Routing Information Protocol)

### بررسی اجمالی پروتکل:

پروتکل RIP یک پروتکل مسیریابی بر پایه distance-vector است که از الگوریتم Bellman-Ford استفاده می‌کند. این پروتکل به مسیریاب‌ها این امکان را می‌دهد تا توپولوژی شبکه را به صورت دینامیک کشف کرده و کوتاه‌ترین مسیرها به گره‌های مقصد را محاسبه کنند.

### نحوه عملکرد پروتکل:

#### ۱- شناسایی همسایگان:

در ابتدای کار، روترها هیچ اطلاعی از همسایگان خود ندارند. برای شناسایی همسایگان، هر روتر از طریق کلاس پورت، سیگنال‌هایی به شبکه ارسال می‌کند. در صورتی که پاسخی از یک نود دریافت شود، روتر لیست همسایگان خود را با اطلاعات جدید به‌روزرسانی می‌کند. در این حالت اگر روتر همسایه‌ای خراب باشد، با دریافت نکردن جواب از آن می‌توانیم متوجه خرابی آن بشویم.

#### ۲- مقداردهی اولیه:

هر روتر با دانستن مستقیم همسایگان خود و هزینه‌های متصل به آن‌ها شروع می‌کند که این هزینه‌ها در سرتاسر شبکه یکسان است. هزینه‌های همه نودهای غیرهمسایه به‌طور پیش‌فرض بی‌نهایت در نظر گرفته می‌شود.

#### ۳- به‌روزرسانی مسیریابی:

یک به‌روزرسانی مسیریابی به همسایگان ارسال می‌شود تا اطلاعات مسیریابی در شبکه منتشر شود. در این تمرین جهت سادگی، این به‌روزرسانی تنها یک‌بار انجام می‌شود و نیازی به به‌روزرسانی دوره‌ای (مانند پروتکل اصلی که هر ۳۰ ثانیه به‌روزرسانی می‌شود) نیست.

به‌روزرسانی با استفاده از الگوریتم Bellman-Ford انجام می‌شود تا مسیرهای کوتاه‌تر به صورت تدریجی محاسبه شوند.

#### ۴- مدیریت جدول مسیریابی:

همانطور که پیشتر گفته شد، هر روتر یک جدول مسیریابی (routing table) نگه می‌دارد که شامل مقصد، ماسک زیرشبکه (subnet mask)، next-hop gateway و هزینه (metric) می‌شود.

این جدول به صورت دینامیک هر زمان که مسیرهای بهتر (با هزینه کمتر) پیدا شوند یا همسایگان قابل دسترسی نباشند، به‌روزرسانی می‌شود.

#### ۵- مدیریت Packet ها:

بسته‌های RIP بین همسایگان در شبکه رد و بدل می‌شوند که شامل اطلاعات مسیریابی در payload هستند. نوع این بسته‌ها همانطور که گفته شد از نوع control می‌باشند

### ۶- نهایی شدن مسیریابی:

در پیاده‌سازی RIP (بدون به‌روزرسانی‌های دوره‌ای)، جدول‌های مسیریابی پس از تبادل اولیه اطلاعات مسیریابی نهایی می‌شوند.به طور خاص، زمانی که یک روتر به‌روزرسانی‌ها را از همه همسایگان دریافت کرده و جدول مسیریابی خود را متناسب با آن به‌روزرسانی کند، اطلاعات مسیریابی پایدار تلقی می‌شود.

شبکه زمانی برای ترافیک عادی آماده است که همه روترها به‌روزرسانی‌ها را پردازش کرده و هیچ تغییری در جدول‌ها مشاهده نشود.

برای اعلام پایان انجام پروتکل مسیریابی، باید چک شود که:

پس از مدت زمان معقول، هیچ به‌روزرسانی جدیدی از همسایگان دریافت نمی‌شود و تغییری در هزینه مسیرهای موجود ایجاد نمی‌شود.

نحوه ی اطلاع سیستم از اتمام فرآیند شناسایی (فاز ۲) در بخش ۶ توضیح داده شده است

## الگوریتم OSPF (Open Shortest Path First):

### بررسی اجمالی پروتکل:

پروتکل OSPF یک پروتکل مسیریابی بر اساس (link-state) است که از الگوریتم Dijkstra استفاده می‌کند. این پروتکل برای همگرایی سریع‌تر و تصمیم‌گیری دقیق‌تر مسیریابی طراحی شده است.

### نحوه عملکرد پروتکل:

#### ۱- شناسایی همسایگان:

در ابتدای کار، روترها هیچ اطلاعی از همسایگان خود ندارند. برای شناسایی همسایگان، هر روتر از طریق کلاس پورت، سیگنال‌هایی به شبکه ارسال می‌کند. در صورتی که پاسخی از یک نود دریافت شود، روتر لیست همسایگان خود را با اطلاعات جدید به‌روزرسانی می‌کند. در این حالت اگر روتر همسایه‌ای خراب باشد، با دریافت نکردن جواب از آن می‌توانیم متوجه خرابی آن بشویم.

#### ۲- مقداردهی اولیه:

روترها تنها همسایگان و هزینه‌های متصل به آن‌ها را می‌شناسند. هزینه‌های همه گره‌های غیرهمسایه به‌طور پیش‌فرض بی‌نهایت در نظر گرفته می‌شود.

#### ۳- انجام Link-State Advertisements (LSAs):

هر روتر اطلاعات توپولوژی خود را از طریق LSAs منتشر می‌کند که به سایر روترها اجازه می‌دهد یک دیدگاه همگن و ثابت از شبکه داشته باشند.

#### ۴- الگوریتم Dijkstra:

با استفاده از LSAs، روترها یک پایگاه داده linkstate یا (LSDB) ایجاد می‌کنند که نمایی از شبکه به‌طور کامل است.

سپس الگوریتم Dijkstra برای محاسبه کوتاه‌ترین مسیرها به هر مقصد از این پایگاه داده استفاده می‌شود.

#### ۵- مدیریت جدول مسیریابی:

با استفاده از کوتاه‌ترین مسیرهای محاسبه‌شده تا دیگر نودها، جدول مسیریابی پر می‌شود که برای هر مقصد، next-hop gateway را مشخص می‌کند.

#### ۶- مدیریت Packetها:

بسته‌های OSPF برای انجام LSAs و همگام‌سازی جداول مسیریابی بین روترها استفاده می‌شوند. این بسته‌ها نیز از نوع control می‌باشند.

**۷- نهایی شدن مسیریابی:**

جدول‌های مسیریابی OSPF پس از تبادل و پردازش همه (LSA)ها و هماهنگ‌سازی پایگاه داده (LSDB) بین همه روترها نهایی می‌شوند.

هر روتر سپس درخت کوتاه‌ترین مسیر را با استفاده از الگوریتم Dijkstra محاسبه کرده و جدول مسیریابی خود را به‌روزرسانی می‌کند.

پس از اتمام این فرآیند، جدول‌های مسیریابی پایدار تلقی می‌شوند.

برای اعلام پایان انجام پروتکل مسیریابی، باید چک شود که:

همه روترها LSAها را از همسایگان دریافت کرده و تأیید کرده‌اند و محاسبات کوتاه‌ترین مسیر به اتمام رسیده و برای مدت زمان معقولی اعلان جدیدی تولید نمی‌شود.

نحوه ی اطلاع سیستم از اتمام فرآیند شناسایی (فاز ۲) در بخش ۶ توضیح داده شده است

**پورت ها:**

پورت‌های یک روتر نقطه ورود و خروج ارتباطات در شبکه است. این پورت‌ها برای اتصال به دستگاه‌های دیگر یا به شبکه خارجی استفاده می‌شوند و وظیفه ارسال و دریافت داده‌ها را بر عهده دارند.

برای این تمرین نیاز است تا پورت ها شماره گذاری شوند. مطابق با موارد ذکر شده در ابتدای تمرین، بافر میان پورت های روتر مشترک است.

پورت ها را دو طرفه بهم هم متصل کنید (یعنی اگر پورت ۱ از روتر X به پورت ۴ از روتر Y متصل شد، لازم به انتصال برعکس نباشد)

در پیاده سازی پورت ها از Signal-slot system استفاده کنید.

**DHCP**

Dynamic Host Configuration Protocol (DHCP) یک پروتکل شبکه است که استفاده می‌شود تا به دستگاه‌هایی که به یک شبکه متصل شده‌اند، به طور خودکار آدرس IP و تنظیمات دیگر شبکه از جمله ماسک زیرشبکه، آدرس دروازه، و آدرس DNS را اختصاص دهد. ویژگی DHCP در روتر به طور کلی به شرح زیر است:

- **تخصیص آدرس IP به دستگاه‌ها**:
- یکی از وظایف اصلی DHCP در روتر این است که به دستگاه‌هایی که به شبکه متصل شده‌اند، به طور خودکار آدرس IP اختصاص دهد. این امر از دستکاری دستی و مداوم در تنظیمات آدرس IP برای هر دستگاه جلوگیری می‌کند و به مدیران شبکه اجازه می‌دهد تا شبکه را به راحتی مدیریت کنند.

## Ipv6

Ipv۶در روترها به طریق مشابهی با Ipv4 کار می‌کند اما با تفاوت‌هایی در ساختار آدرس‌دهی و پردازش داده‌ها. در ادامه، نحوه کارکرد Ipv6 در روترها به شرح زیر است:

۱. **آدرس‌دهی Ipv6:**

- در Ipv۶ ، آدرس‌ها به صورت ۱۲۸ بیتی نمایش داده می‌شوند که باعث افزایش انتقال پذیری آدرس‌ها و افزایش تعداد دستگاه‌ها در شبکه می‌شود. روترها مسئول تخصیص آدرس‌های Ipv6 به دستگاه‌های متصل به شبکه هستند و از پروتکل‌هایی مانند DHCPv6 استفاده می‌کنند تا آدرس‌ها را به دستگاه‌ها اختصاص دهند.

۲. **پردازش بسته‌های Ipv6:**

- بسته‌های داده Ipv6 شامل سرآیندهایی هستند که شامل آدرس‌های Ipv6 مبدأ و مقصد می‌شوند. روترها به طور مشابهی با Ipv4 ، از اطلاعات در سرآیند برای تصمیم‌گیری در مورد مسیریابی بسته‌ها استفاده می‌کنند و بر اساس جدول مسیریابی خود، بسته‌ها را به مقصد مورد نظر هدایت می‌کنند.

**درباره نحوه encapsulation در کتاب توضیحات لازم اورده شده است.**

**بافر:**

ویژگی‌های بافر در روترها عمدتاً شامل موارد زیر می‌شود:

۱. **اندازه بافر (Buffer Size):**

- این ویژگی مشخص می‌کند که چه تعداد داده را می‌توان در بافر ذخیره کرد. اندازه بافر بر اساس نیازهای شبکه و ظرفیت دستگاه تنظیم می‌شود.

۲. **نحوه مدیریت بافر (Buffer Management):**

- این ویژگی مشخص می‌کند که چگونه داده‌ها در بافر مدیریت می‌شوند. این شامل روش‌های مختلفی از جمله FIFO (First In, First Out) ، Tail Drop، RED (Random Early Detection) و WRED (Weighted Random Early Detection) می‌شود که هر کدام ویژگی‌ها و عملکرد خاص خود را دارند.

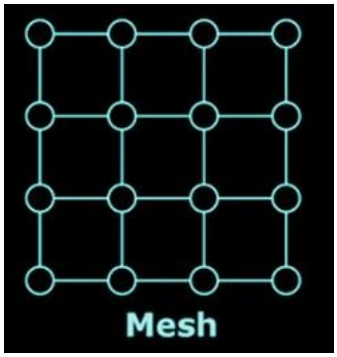
۳. **نحوه نگهداری داده‌ها (Data Retention):**

- این ویژگی مشخص می‌کند که چگونه داده‌ها در بافر نگهداری می‌شوند و چه مدت زمانی در بافر باقی می‌مانند. برخی از سیاست‌های نگهداری می‌توانند شامل FIFO یا LIFO (Last In, First Out) باشند.

## توپولوژی های شبکه:

### ۱. توپولوژی مش (Mesh Topology):

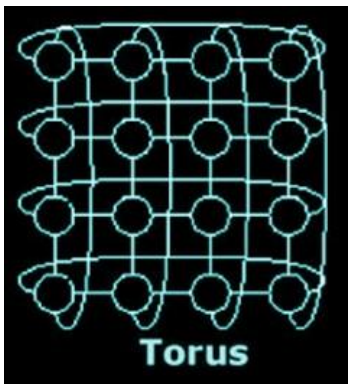
در توپولوژی مش، هر دستگاه به دستگاه های مجاور خود در راستای محور های مختصات وصل میباشد. به شکل زیر:




*شکل ۵ – توپولوژی مش*

### ۲. توپولوژی تورس (Torus Topology):

این توپولوژی الهام گرفته از توپولوژی مش میباشد با این تفاوت که رئوسی که در ابتدا و انتهای هر سطر و ستون قرار دارند، به هم دیگر وصل هستند. به طور ساده، می‌توان توپولوژی تورس را به صورت یک شبکه مربعی دوبعدی تصور کرد که مرزهای آن به یکدیگر متصل هستند.




*شکل ۶ - توپولوژی تورس*

### ۳. توپولوژی رینگ استار (Ring Star Topology):

ابتدا به توضیح توپولوژی رینگ و استار میپردازیم:

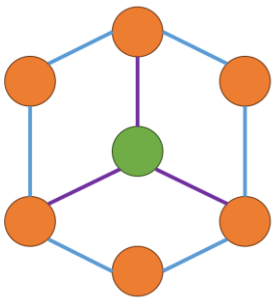
توپولوژی رینگ: در توپولوژی رینگ، دستگاه ها به صورت دایره ای مرتبط هستند، به طوری که هر دستگاه با دو دستگاه مجاور خود ارتباط دارد.


*شکل ۷ - توپولوژی رینگ*

توپولوژی استار: در توپولوژی استار، یک دستگاه مرکزی وجود دارد که با همه‌ی دستگاه‌های دیگر به صورت مستقیم متصل است. دستگاه‌های دیگر فقط به دستگاه مرکزی متصل هستند و نه به یکدیگر.


*شکل ۸ - توپولوژی استار*

حال در توپولوژی رینگ استار، ترکیب این دو توپولوژی را استفاده میکنیم. برای مثال شکل زیر را در نظر بگیرید:



*شکل ۹ - توپولوژی رینگ استار*

یک شبکه متشکل از تعدادی نود با توپولوژی رینگ میسازیم و در مرکز این رینگ یک نود دیگر قرار میدهیم، حال نود وسط به صورت یکی در میان به نود های موجود در رینگ متصل میشود.

## هدر IP ورژن ۴:

هدر مورد نظر ما برای IPv4 دارای این ویژگی ها است:

```
private:     // members
    uint8_t   m_versionHeaderLength;
    uint8_t   m_typeOfService;
    uint16_t  m_totalLength;
    uint16_t  m_identification;
    uint16_t  m_flagsFragmentOffset;
    uint8_t   m_ttl;
    uint8_t   m_protocol;
    uint16_t  m_headerChecksum;
    IPv4Ptr_t m_sourceIp;
    IPv4Ptr_t m_destIp;
```

*شکل ۱۰ - ویژگی های هدر IPv4*

## هدر IP ورژن ۶:

هدر مورد نظر ما برای IPv6 دارای این ویژگی ها است:

```
private:     // members
    uint32_t  m_versionTrafficClassFlowLabel;
    uint16_t  m_payloadLength;
    uint8_t   m_nextHeader;
    uint8_t   m_hopLimit;
    IPv6Ptr_t m_sourceIP;
    IPv6Ptr_t m_destIP;
```

*شکل ۱۱ - ویژگی های هدر IPv6*

به ویژگی های هر دو ورژن IP، version را نیز اضافه کنید (اگر از abstract class به عنوان کلاس base استفاده میکنید)

## هدر TCP:

```
private:
    uint16_t m_sourcePort;
    uint16_t m_destPort;
    uint32_t m_sequenceNumber;
    uint32_t m_acknowledgmentNumber;
    uint8_t  m_dataOffset;
    uint8_t  m_flags;    // URG, ACK, PSH, RST, SYN, FIN
    uint16_t m_windowSize;
    uint16_t m_checksum;
    uint16_t m_urgentPointer;
```

شکل ۱۲ - ویژگی های هدر TCP

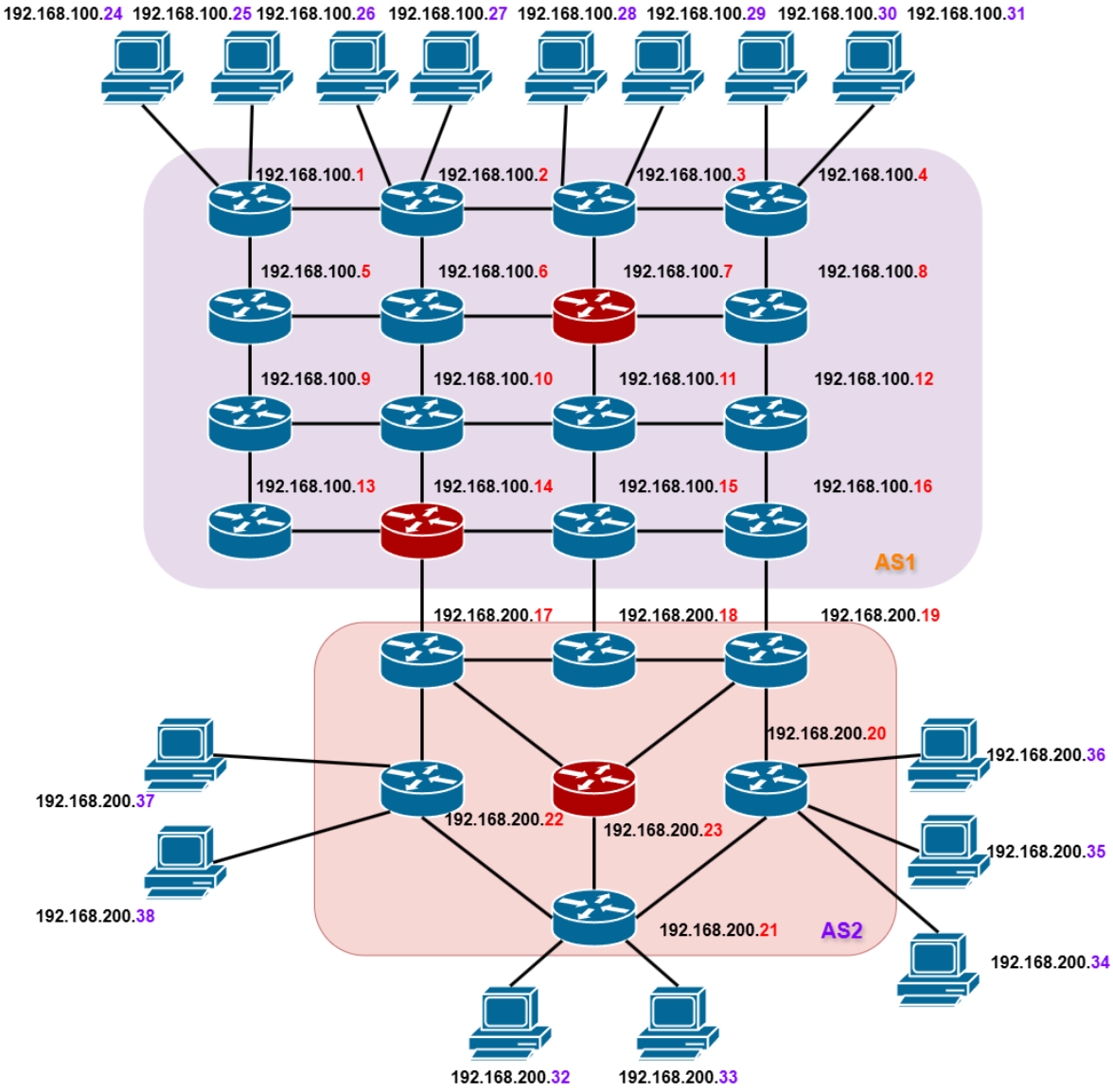
## هدر DataLink:

```
private:
    MACAddress m_sourceMACAddress;
    MACAddress m_destinationMACAddress;
```

شکل ۱۳ – ویژگی های هدر Data Link

## ۵- بخش های امتیازی

۱- IP version 6
۲- BGP
۳- تغییر توپولوژی Mesh به Torus در AS اول (یک بار مش و بار دیگر تورس)
۴- بررسی رفتار شبکه وقتی روتری خراب است




*شکل ۱۴ - روتر های خراب شبکه*

در تصویر بالا سه روتری که با رنگ قرمز مشخص شده اند خراب هستند (از اولین فاز اجرای برنامه این سه روتر خراب هستند) بررسی های هدف تمرین را در این حالت نیز انجام دهید و با حالت اول مقایسه کنید.

# ۶- فاز های اجرای شبیه ساز

اجرای شبیه ساز شامل ۴ فاز کلی میشود:

### فاز اول (نقطه صفر)

در این فاز از کلاس های پیاده سازی شده به تعداد لازم instance ساخته میشود و فایل کانفیگ در برنامه خوانده میشود. برمبنای فایل کانفیگ AS ها و Router ها و ... مطابق با توپولوژی ها ساخته و به یک دیگر متصل میشود. انتهای این فاز مانند آن است که شبکه ای ساخته شده اما تمام روتر ها خاموش هستند.

### فاز دوم (شناسایی)

در این فاز EventCoordinator شروع به کار میکند، روتر ها شروع به ارسال پیام های کنترلی به یک دیگر میکنند تا Routing Table های خود را بسازند و شبکه را شناسایی کنند.(در این فاز PC ها نباید کاری انجام بدهند!) DHCP Server ها به تمامی نود های شبکه IP تخصیص میدهند و شبکه آماده ی خدمت رسانی است.

باتوجه به توضیحات مندرج در بخش تعاریف راجع به الگوریتم های مسیریابی، در هر سیکل اگر جدول مسیریابی router به روزرسانی نشد، یک سیگنال مربوط به اتمام فرآیند emit میکند، شما باید تعداد دفعات emit شدن این سیگنال در سیکل را بشمارید، اگر در چند سیکل (به انتخاب شما) تعداد دفعات emit شدن این سیگنال برابر با تعداد روتر های شبکه بود، این فاز به اتمام میرسد. این شمارش را در کلاس Network یا Simulator انجام دهید.

### فاز سوم (اجرا)

در این فاز PC ها شروع به ارسال بسته در شبکه میکنند. مدت زمان این فاز از روی فایل کانفیگ تنظیم میشود.

### فاز چهارم (تحلیل)

در این فاز شبیه سازی تمام شده و شبیه ساز آماده است تا اطلاعات درخواستی ما که در بخش ۹ (ارزشیابی) توضیح داده شده اند را نمایش دهد، برنامه آماده است تا دستوراتی را از ورودی بخواند و اجرا کند، مجموعه دستورات شامل:

- **Packet-loss: چند درصد packet loss داریم؟**
- **Hop-count-avg: میانگین تعداد گام هایی که بسته ها طی کرده اند تا به مقصد برسند.**
- **Waiting-cycles-stat: مجموع کل، میانگین و کمینه و بیشینه ی انتظار بسته ها در بافر روتر ها.**
- **Used-routers: لیست روتر هایی که بسته ی کاربران به آنها رسیده است.**
- **Poor-routers: روتر هایی که بیشترین تعداد بسته های کاربر را مسیریابی کرده اند.**
- **Print-routing-table <N>: چاپ روتینگ تیبل نود N (آیدی روتر)**
- **Exit: اتمام اجرای شبیه ساز**
- **Reset: بازگشت به فاز یک (نقطه صفر)**
- **Clean: پاک کردن تمام فایل های مربوط به لاگ ها و ... (بجز فایل کانفیگ)**

## ۷- ارزیابی

۱- **شما باید دو بار فرآیند شبیه سازی را انجام دهید، بار اول همه ی روتر ها از RIP و بار دوم از OSPF برای مسیریابی استفاده کنند، بررسی کنید که کدام الگوریتم جواب بهتری گرفته است. معیار های مقایسه:**

- **چند درصد بسته ها به مقصد نرسیده اند؟**
- **میانگین تعداد گام هایی که بسته ها طی کرده اند تا به مقصد برسند.**
- **مقایسه ی مجموع کل، میانگین و کمینه و بیشینه ی انتظار بسته ها در بافر روتر ها.**
- **کدام الگوریم از فضای گسترده تری از شبکه استفاده کرده است (بر اساس مسیر حرکت بسته بین روتر ها).**
- **در نقطه ی اوج بار ترافیک کدام الگوریتم بهتر کار کرده است (از منظر شما با معیار های شما)**
- **در هر الگوریتم کدام روتر ها بیشترین تعداد بسته را route کرده اند؟ (چهار روتر اول)**

نحوه ی تخصیص توزیع آماری بدین شرح است:

۱- گروه هایی که مجموع رقم انتهایی شماره دانشجویی آنها زوج است از توزیع پوآسون برای تولید داده استفاده کنند

۲- گروه هایی که مجموع رقم انتهایی شماره دانشجویی آنها فرد است از توزیع پارتو برای تولید داده استفاده کنند.

# ۸- فاز های تحویل پروژه

فاز ۱ -> ددلاین: ۱۶ آذر

موارد تحویلی

- EventsCoordinator
- DataGenerator
- MACAddress
- MACAddressGenerator
- Packet (without IP & IP Header)
- DataLinkHeader
- TCPHeader
- Report

فاز ۲ -> ددلاین: ۲۳ آذر

موارد تحویلی

- IP
- IPHeader
- Port
- PortBindingManager
- Node
- PC
- Router (without routing algorithms)
- TopologyBuilder
- TopologyController
- AutonomousSystem
- Test
- Report

فاز ۳ -> ددلاین: ۴ دی

موارد تحویلی

- RIP
- OSPF
- DHCPServer
- Network
- ApplicationContext
- Test
- Report

فاز ۴ -> ددلاین: ۷ دی

موارد تحویلی

- Simulator
- Test
- Report

رعایت این فازبندی اجباری است، برای هر فاز یک محل آپلود در سامانه قرار خواهد گرفت.

برای ددلاین های ۱ و ۲ و ۳ در مجموع ۶ روز Grace دارید که میتوانید از آن استفاده کنید، ددلاین ها به هیچ عنوان تمدید نخواهند شد.

ددلاین فاز ۴ ددلاین نهایی تمرین است و شامل Grace نمیشود!

پس از انجام هر فاز، علاوه بر آن که کد ها در گیت هاب هستند، فایل های تمرین را در یک آرشیو rar از اول تمرین تا انتهای آن ددلاین روی سامانه آپلود کنید.

گزارش هر فاز را در همان فاز تکمیل کنید و در نهایت در ددلاین تمرین گزارش کاملی از کل فرآیند تمرین خواهد داشت.

# جمع بندی و نکات پایانی

- مهلت تحویل: **۷ دی ماه ۱۴۰۳**
- پروژه در گروه‌های ۲ نفره انجام می‌شود. (گروه بندی در سامانه ایلرن نیز انجام می‌شود و تحویل تمرین به صورت گروهی خواهد بود)

- هر ۲ نفر می بایست کار را تقسیم کنند. همچنین از Git برای ساختن branch و تقسیم issue ها استفاده نمایید. (با استفاده از commit ها و تعیین issue ها میزان مشارکت هر نفر مشخص می‌شود). بعد از انجام این کار کدها را در یک repository به نام CN_CHomeworks_۳ در اکانت‌های GitHub/GitLab خود قرار دهید(به صورت private). همچنین در یک فایل README.md می‌توانید report و داکیومنت خود را کامل کنید و در کنار repository قرار دهید. در نهایت لینک این repository را در محل پاسخ تمرین قرار دهید. (از فرستادن فایل به صورت زیپ جدا خودداری نمایید.) اکانت تی ای‌های این تمرین رو به Repo خودتون به پروژه اضافه کنید.

  **Git Accounts:** @TheSohrabX - @Mohta3b - @hesamhrf - @AliDarabi9

- برای پیاده سازی این تمرین از C++ استفاده کنید.
- دقت کنید گزارش نهایی شما می‌بایست همانند یک Document باشد و شامل توضیح کد و ساختار کد، همچنین نتیجه نهایی اجرای کد و اسکرین شات‌های دقیق از تمام مراحل باشد. (در فایل Readme.md کنار فایل های اصلی خود و در Repo مربوطه قرار دهید.) این نکته حائز اهمیت است که فایل PDF به هیچ عنوان مورد پذیرش قرار نخواهد گرفت.
- ساختار صحیح و تمیزی کد برنامه، بخشی از نمره‌ی این پروژه شما خواهد بود. بنابراین در طراحی ساختار برنامه دقت به خرج دهید.
- برای هر قسمت کد، گزارش دقیق و شفاف بنویسید. کدهای ضمیمه شده بدون گزارش مربوطه نمره‌ای نخواهند داشت.
- هدف این تمرین یادگیری شماست. لطفا تمرین را خودتان انجام دهید. در صورت مشاهده‌ی مشابهت بین کدهای دو گروه، مطابقت سیاست درس با گروه متقلب و تقلب دهنده برخورد خواهد شد. همچنین توجه داشته باشید استفاده از ابزارهای AI توجیهی برای شباهت کدهای تحویل داده شده توسط گروه های مختلف نمی باشد.
- سؤالات خود را تا حد ممکن در گروه درس مطرح کنید تا سایر دانشجویان نیز از پاسخ آن بهره‌مند شوند. در صورتی که قصد مطرح کردن سؤال خاص‌تری دارید، از طریق ایمیل زیر ارتباط برقرار کنید. توجه داشته باشید که سایر شبکه‌های اجتماعی راه ارتباطی رسمی با دستیاران آموزشی نیست و دستیاران آموزشی مؤظف به پاسخگویی در محیط‌های غیررسمی نیستند.
- ایمیل دستیاران طراح:

  - m.moradi1998@ut.ac.ir
  - amir.ali.vahidi@ut.ac.ir
  - alidarabi9981@gmail.com
  - hesamhrf@gmail.com

موفق باشید.