بسم الله الرحمن الرحیم

تمرین برنامه نویسی شماره ۰۳

عنوان: پروژه برنامه نویسی- شماره ۰۳

درس: شبکه‌های کامپیوتری

استاد راهنما: دکتر ناصر یزدانی[۱]

رشته: مهندسی کامپیوتر

دستیاران آموزشی: اسامه ایراندوست[۲]، احمدرضا پارسی‌زاده[۳]

نیمسال دوم سال تحصیلی ۰۲-۱۴۰۱

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

[۲] نشانی پست الکترونیکی: osameh.irandoust@ut.ac.ir

[۳] نشانی پست الکترونیک:

فهرست مطالب

عنوان پروژه

هدف از این تمرین آشنایی با الگوریتم های مسیریابی در شبکه است. در این تمرین با زبان C++ به پیاده‌سازی الگوریتم‌های Border Gateway و distance vector و link state و مقایسه کردن آنها خواهد پرداخت.

مقدمه

در این تمرین یک توپولوژی عمومی که از تعدادی روتر تشکیل شده است طراحی خواهید کرد. هر کدام از لینک‌های میان روترها هزینه‌ای خواهندداشت و هر روتر جدول روتینگ و ارسال بسته‌های خود را با استفاده از الگوریتم‌های Border Gateway و distance vector و link state محاسبه می‌کند. در آخر این جدول‌ها چاپ می‌شوند.

دستورات مربوطه توسط خط فرمان (CLI) به برنامه داده می‌شوند. در ادامه شرح این دستورات را خواهیم داشت:

۱- تعیین توپولوژی شبکه

با این دستور ارتباط بین گره‌های شبکه و نیز هزینه مسیریابی بین دو گره مشخص می‌شود.

توجه: ارتباط بین گره‌ها بدون جهت می‌باشد .

گره‌های مبدا و مقصد نمی‌توانند یکسان باشند و درصورت یکسان بودن باید خطای مناسب چاپ شود.

برای مثال ۱۷ - ۲ - ۱ به این معنی است که مسیری از نود ۱ به نود ۲ و با هزینه ۱۷ وجود دارد .

ورودی
topology <s1>-<d1>-<c1> <s2>-<d2>-<c2> …
خروجی
OK
ورودی نمونه
topology 1-2-19 1-3-9 2-4-3
خروجی نمونه
OK

۲- نمایش توپولوژی شبکه

با استفاده از این دستور، ارتباط بین نودهای مختلف به همراه هزینه بین نودها در قالب یک جدول نمایش داده می‌شود .

ورودی
show
خروجی
Adjacency Matrix
ورودی نمونه
show
خروجی نمونه

```
  | 1 2 3 4
---------------------------------
1 | 0 19 9 -1
2 | 19 0 -1 3
3 | 9 -1 0 -1
4 | -1 3 -1 0
```

۳- تغییر هزینه مسیریابی

با استفاده از این دستور می‌توان هزینه مسیریابی بین دو گره در شبکه را تغییر داد. همچنین اگر مسیری بین دو گره مشخص شده وجود نداشته باشد، یک مسیر جدید ایجاد خواهد شد. گره‌های مبدا و مقصد نمی‌توانند یکسان باشند و درصورت یکسان بودن باید خطای مناسب چاپ شود.

ورودی
modify <s>-<d>-<c>
خروجی
OK | Error
ورودی نمونه
modify 1-3-4
خروجی نمونه
OK

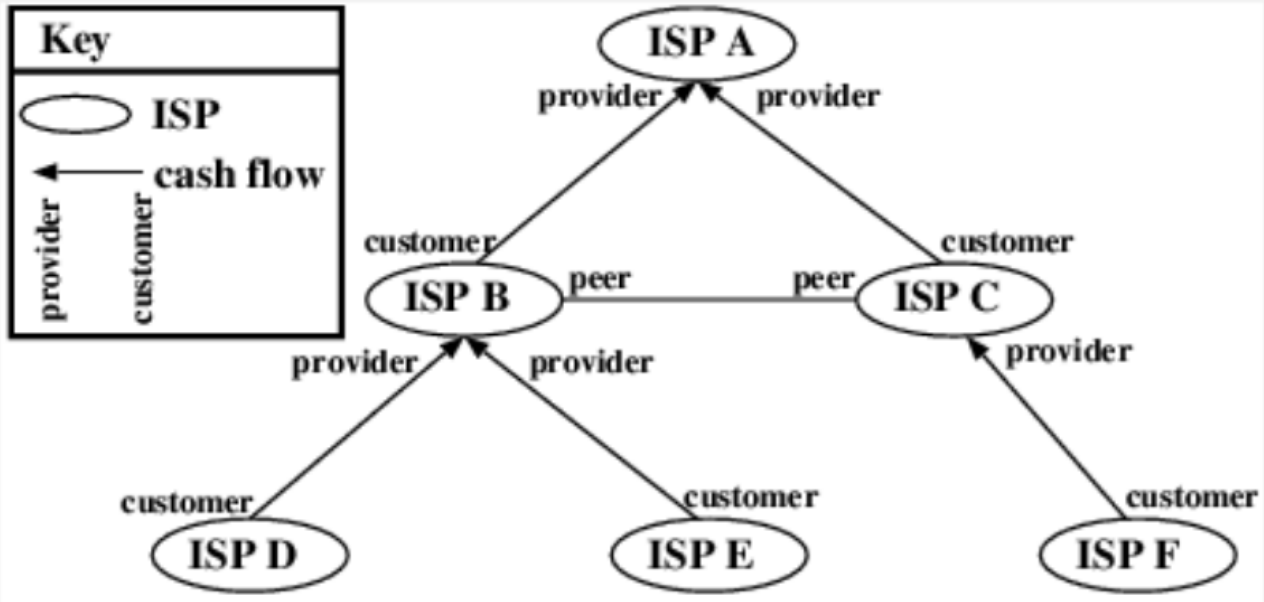
۴- حذف ارتباط بین دو گره

با استفاده از این دستور، مسیر میان دو گره مشخص شده از بین می‌رود (اگر گره یا مسیر وجود نداشت ارور دلخواه بدهد).

ورودی
remove <s>-<d>
خروجی
OK | Error
ورودی نمونه
remove 1-3
خروجی نمونه
OK

۵- پیاده سازی الگوریتم Border Gateway (BGP)

در این بخش الگوریتم BGP پیاده‌سازی می‌شود. شما باید کاری که یک AS (Autonomous System) در این الگوریتم انجام می‌دهد را پیاده‌سازی کنید.



رابطه‌ی بین AS ها دو حالت مشتری-تامین‌کننده یا همتا-همتا می‌باشد. در هنگام ساخت AS توپولوژی همسایه‌ها و نوع رابطه به او داده می‌شود. شما باید قوانین موجود در الگوریتم و نحوه تبلیغات و اولویت‌بندی‌ها را پیاده‌سازی کنید. همچنین در صورت یکسان شدن اولویت‌ها ابتدا به مسیر کوتاه‌تر و سپس مسیر با شماره کم‌تر اولین واسط اولیت می‌دهیم.

در مسیر BGP ممکن است مشکلات امنیتی رخ دهد. مثلا یک AS پیشوندی را تبلیغ کند که متعلق به خودش نیست. با یک AS اقدام به کوتاه‌تر یا بلندتر نشان‌دادن یک مسیر نماید. یا اینکه یک AS خاص را از مسیری حذف یا به آن اضافه کند. باید این موارد را در نظر بگیرید و تشخیص دهید.

در این بخش که یک شبیه‌سازی است نیازی به دانستن جزئیات و نحوه ساختن بسته‌ها ندارید. اطلاعات مورد نیاز از طریق پارامترهای توابع ارسال می‌شود. اما توصیه می‌شود نگاهی به RFC 4271 بیاندازید. همچنین شماره AS ها از ۰ تا ۳۱ است.

اگر چند مسیر به یک پیشوند پیدا کردید باید اولویت آنها به‌ترتیب: مسیرها از مشتری، مسیرهای از همتا، مسیرهای از سرویس‌دهنده قرار دهید و بر مبینای آن یک مسیر گزارش کنید. مسیرهایی از همتا به‌دست ما می‌رسد به دیگر همتاها گزارش نمی‌کنیم. همچنین یک مشتری از مسیرهایی که از یک سرویس‌دهنده گرفته باشد را به سرویس‌دهنده دیگر گزارش نمی‌کند.

در صورتی که یک پیشوند را از قبل متعلق به یک AS بدانیم و پیامی با مبدا متفاوت برای آن پیشوند دریافت کنیم آن را hijack تشخیص می‌دهیم.

توجه: به طور دلخواه می‌توانید این الگوریتم و توابع آن را طراحی کنید (قالب دستورات این بخش به دلخواه خودتان است و می‌توانید با بیش از یک دستور این بخش را پیاده‌سازی کنید).

۶- پیاده سازی الگوریتم Link State (LSRP)

این دستور شماره نود مبدا را گرفته و جدول مسیریابی را برای آن رسم می‌کند. همچنین جزئیات جدول در هر iteration را نیز چاپ کنید.

در صورتی که این دستور بدون آرگومان وارد شود، نتایج را برای تمامی گره‌های شبکه چاپ کنید.

ورودی

```
lsrp <source>
```

خروجی

```
Cost per destination in iterations. In end It shows final result.
```

ورودی نمونه

```
lsrp 1
```

خروجی نمونه

```
Iter 1:
Dest 1 | 2 | 3 | 4 |
cost 0 | 19 | 9 | -1 |
------------------------
Iter 2:
Dest 1 | 2 | 3 | 4 |
cost 0 | 19 | 9 | -1 |
------------------------
Iter 3:
Dest 1 | 2 | 3 | 4 |
cost 0 | 19 | 9 | 22 |
------------------------
Path [s] -> [d] | Min-Cost | Shortest-Path
1->2 | 19 | 1->2
1->3 | 9 | 1->3
1->4 | 22 | 1->2->4
```

۷- پیاده سازی الگوریتم Distance Vector (DVRP)

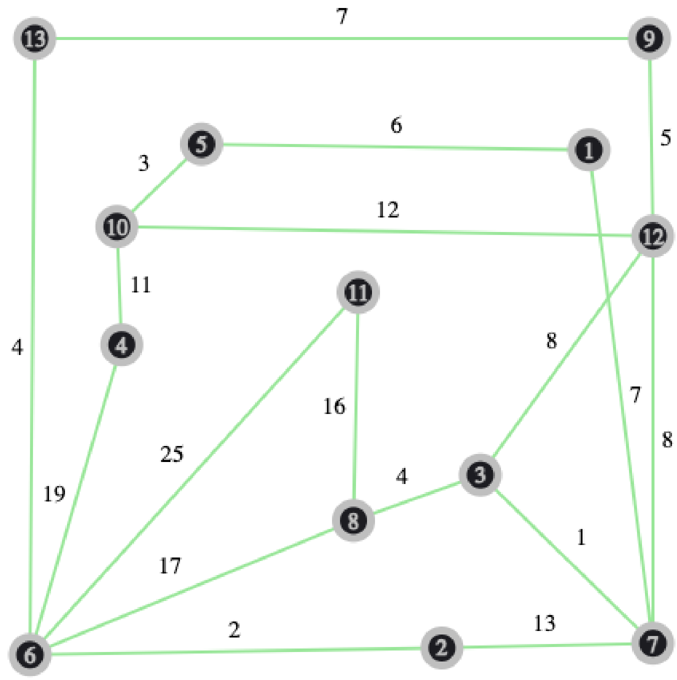
این دستور شماره نود مبدا را گرفته و جدول مسیریابی را برای آن رسم می‌کند. در صورتی که این دستور بدون آرگومان وارد شود، نتایج را برای تمامی گره‌های شبکه چاپ کنید.

ورودی
`dvrp <source>`
خروجی
`It shows route table.`
ورودی نمونه
`dvrp 1`
خروجی نمونه

`Dest | Next Hop | Dist | Shortest-Path`
`-------------------------------------`
`1 | 1 | 0 | [1]`
`2 | 2 | 19 | [1->2]`
`3 | 3 | 9 | [1->3]`
`4 | 2 | 22 | [1->2->4]`

۸- مقایسه نتایج

عملیات مسیریابی را بر روی توپولوژی زیر برای تمامی مقصدها انجام دهید و نتایج هر دو روش را با یکدیگر مقایسه کنید. همچنین زمان همگرایی هر دو الگوریتم مسیریابی را در هر مورد اندازه بگیرید. سپس اقدام به حذف لینک بین دو گره ۱۰ و ۴ کرده و نتایج مسیریابی را به همراه زمان همگرایی پس از حذف لینک مجددا برای هر دو روش محاسبه کنید.


7
13
9
6
5
1
5
3
12
10
12
11
11
4
4
8
7
16
25
8
3
4
19
8
1
17
13
2
6
2
7

۹- جمع بندی و نکات پایانی

- مهلت تحویل: **۱۲ خردادماه**
- 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_1 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 زیر رو به Repo خودتون به عنوان Maintainer به پروژه اضافه کنید.

Github ID: UT-CNs02

- برای پیاده سازی این تمرین از C یا C++ استفاده کنید.
- سیستم عامل ترجیحا Linux باشد.
- 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 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.
- سؤالات خود را تا حد ممکن در فروم 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.

موفق باشید