بسمه تعالی

گزارش تمرین 4
شبکه‌های کامپیوتری

محسن کربلائی امینی، 98242128

تیر 1403

تنظیم مقادیر اولیه و آماده‌سازی کدهای مورد نیاز:

برای آماده‌سازی کد در ابتدا نیاز هست تا کدهای مربوط به گره‌های جدید ساخته شود. کد اصلی node0.h و node0.c را برای هر کدام کپی کرده و سپس با بهره گیری از فایل‌های changes و script.sh هر فایل آماده کامپایل می‌کنیم.

```
┌──(root㉿mohsen-pc)-[/home/…/University/Network/HW4/CN_CA2_DistanceVector]
└─# vi changes

┌──(root㉿mohsen-pc)-[/home/…/University/Network/HW4/CN_CA2_DistanceVector]
└─# ./script.sh node1.h 1
sed: -e expression #1, char 0: no previous regular expression

┌──(root㉿mohsen-pc)-[/home/…/University/Network/HW4/CN_CA2_DistanceVector]
└─# ./script.sh node2.h 2
sed: -e expression #1, char 0: no previous regular expression

┌──(root㉿mohsen-pc)-[/home/…/University/Network/HW4/CN_CA2_DistanceVector]
└─# ./script.sh node3.h 3
sed: -e expression #1, char 0: no previous regular expression
```

محتویات این دو فایل کمکی:

```
┌──(root㉿mohsen-pc)-[/home/…/University/Network/HW4/CN_CA2_DistanceVector]
└─# cat script.sh
#!/bin/bash
file=$1
number=$2

while read line; do
 changedLine=`echo $line | sed "s/0/$number/g"`
 sed -i "s/$line/$changedLine/g" $file || echo $line
done < changes


┌──(root㉿mohsen-pc)-[/home/…/University/Network/HW4/CN_CA2_DistanceVector]
└─# cat changes

node0.h
NODE_ID 0
dt0
connectcosts0
mincosts0
findmincosts0
sendcosts0
rtinit0
printdt0
rtupdate0
linkhandler0
D0
```

و در آخر تنظیم مقادیر اولیه مربوط به هزینه به گره‌های متصل:

```
┌──(root㉿mohsen-pc)-[/home/…/University/Network/HW4/CN_CA2_DistanceVector]
└─# cat node*.c | grep 'int connectcosts'
int connectcosts0[4] = { 0, 1, 999, 7 };
int connectcosts1[4] = { 1, 0, 54, 32 };
int connectcosts2[4] = { 999, 54, 0, 7 };
int connectcosts3[4] = { 7, 32, 7, 0 };

┌──(root㉿mohsen-pc)-[/home/…/University/Network/HW4/CN_CA2_DistanceVector]
```

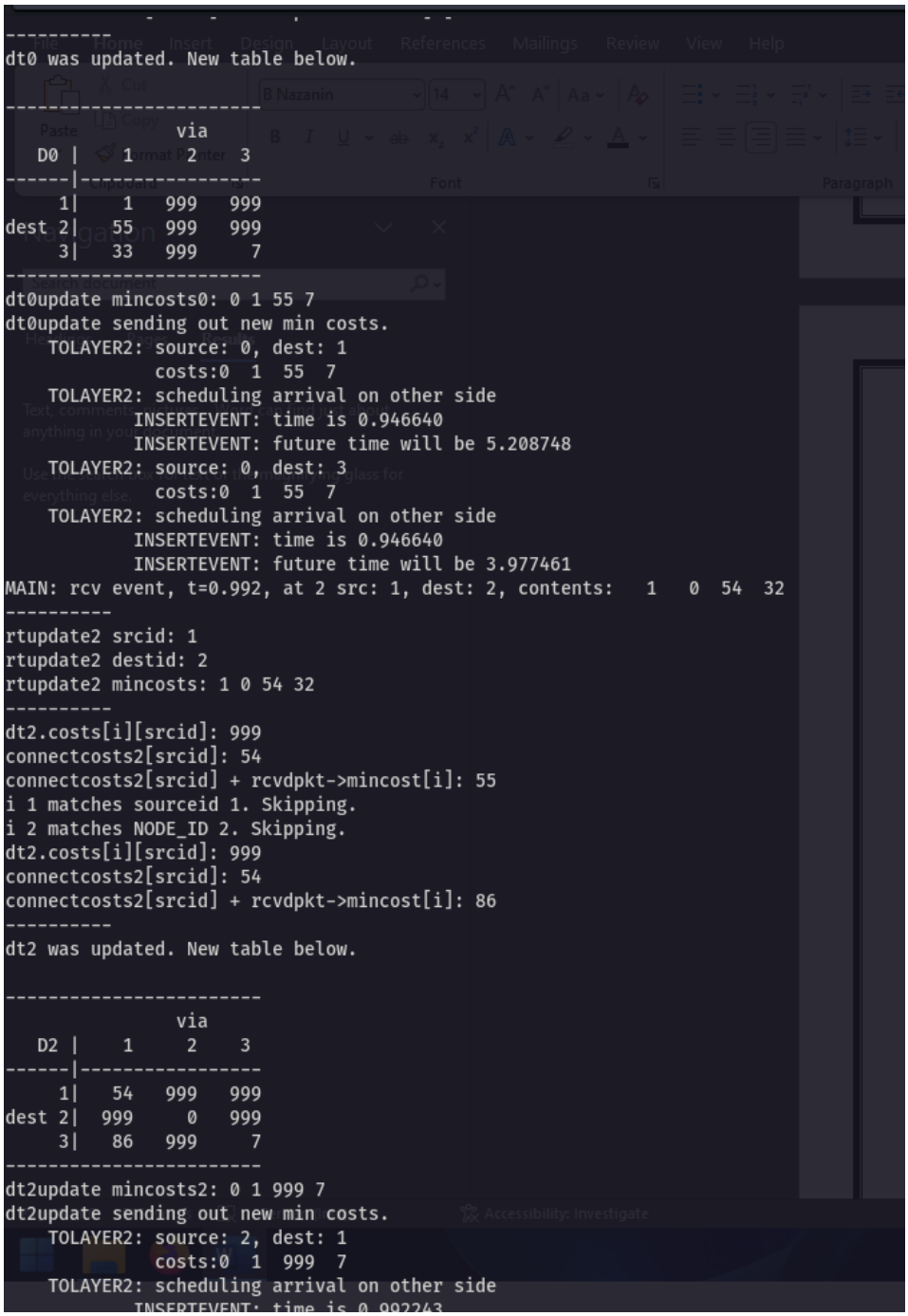
توضیح ارتباطات:

مرحله اول: شناخت خود
در این مرحله جداول اولیه هر گره تولید می‌شود. مثلا برای گره شماره 0 مشخص می‌شود که به هر گره دیگر از چه طریقی می‌تواند عبور کند و اگر یک گره دیگر را برای رسیدن به آن گره واسطه قرار دهد، چه هزینه‌ای را به همراه دارد. برای مثال اگر گره 0 در این مرحله بخواهد با کمک گرفتن گره 1 به مقصد 3 برسد، این کار امکان‌پذیر نیست چرا که هنوز از مسیرهای گره شماره 1 خبر ندارد.
بنابراین فقط از مسیرهای مستقیمی که خودش لینک مستقیم دارد می‌تواند استفاده کند. برای باقی هم به همین شکل:

```
└─# gcc *.c -o run ; ./run | tee ../results.txt
4
Enter TRACE:     TOLAYER2: source: 0, dest: 1
              costs:0  1  999  7
    TOLAYER2: scheduling arrival on other side
            INSERTEVENT: time is 0.000000
            INSERTEVENT: future time will be 1.870574
    TOLAYER2: source: 0, dest: 3
              costs:0  1  999  7
    TOLAYER2: scheduling arrival on other side
            INSERTEVENT: time is 0.000000
            INSERTEVENT: future time will be 1.641910
----------------------
               via
   D0 |    1     2    3
------|----------------
     1|    1   999   999
dest 2|  999   999   999
     3|  999   999     7
----------------------
    TOLAYER2: source: 1, dest: 0
              costs:1  0  54  32
    TOLAYER2: scheduling arrival on other side
            INSERTEVENT: time is 0.000000
            INSERTEVENT: future time will be 0.946640
    TOLAYER2: source: 1, dest: 2
              costs:1  0  54  32
    TOLAYER2: scheduling arrival on other side
            INSERTEVENT: time is 0.000000
            INSERTEVENT: future time will be 0.992243
    TOLAYER2: source: 1, dest: 3
              costs:1  0  54  32
    TOLAYER2: scheduling arrival on other side
            INSERTEVENT: time is 0.000000
            INSERTEVENT: future time will be 2.165707
----------------------
               via
   D1 |    1     2    3
------|----------------
     1|    0   999   999
dest 2|  999    54   999
     3|  999   999    32
----------------------
    TOLAYER2: source: 2, dest: 1
              costs:0  1  999  7
    TOLAYER2: scheduling arrival on other side
            INSERTEVENT: time is 0.000000
            INSERTEVENT: future time will be 3.285052
    TOLAYER2: source: 2, dest: 3
              costs:0  1  999  7
    TOLAYER2: scheduling arrival on other side
            INSERTEVENT: time is 0.000000
            INSERTEVENT: future time will be 3.106066
----------------------
               via
   D2 |    1     2    3
------|----------------
     1|   54   999   999
dest 2|  999     0   999
     3|  999   999     7
----------------------
    TOLAYER2: source: 3, dest: 0
              costs:7  32  7  0
    TOLAYER2: scheduling arrival on other side
            INSERTEVENT: time is 0.000000
```

مراحل میانی: شناخت دیگران و ارتباطاتشان

در این مراحل به مرور زمان گره‌ها در هر مرحله اطلاعات مسیریابی را از دیگران گرفته و تلاش می‌کنند تا جداول خود را بهینه کنند. برای مثال گره 0 که در مرحله اول نمی‌توانست از طریق گره 1 به گره 3 برسد بعد از گذشت مرحله شناخت، با ارتباط گرفتن با گره 1 متوجه می‌شود که این گره لینکی با هزینه 32 با گره 3 دارد. بنابراین 32+1 که هزینه رسیدن از 0 به 1 است، مقدار هزینه 33 را برای این مسیر تعیین می‌کند:



```
----------
dt0 was updated. New table below.

-----------------------
            via
   D0 |   1    2   3
------|----------------
     1|    1   999   999
dest 2|   55   999   999
     3|   33   999     7
-----------------------
dt0update mincosts0: 0 1 55 7
dt0update sending out new min costs.
    TOLAYER2: source: 0, dest: 1
              costs:0  1  55  7
    TOLAYER2: scheduling arrival on other side
            INSERTEVENT: time is 0.946640
            INSERTEVENT: future time will be 5.208748
    TOLAYER2: source: 0, dest: 3
              costs:0  1  55  7
    TOLAYER2: scheduling arrival on other side
            INSERTEVENT: time is 0.946640
            INSERTEVENT: future time will be 3.977461
MAIN: rcv event, t=0.992, at 2 src: 1, dest: 2, contents:   1   0  54  32
----------
rtupdate2 srcid: 1
rtupdate2 destid: 2
rtupdate2 mincosts: 1 0 54 32
----------
dt2.costs[i][srcid]: 999
connectcosts2[srcid]: 54
connectcosts2[srcid] + rcvdpkt->mincost[i]: 55
i 1 matches sourceid 1. Skipping.
i 2 matches NODE_ID 2. Skipping.
dt2.costs[i][srcid]: 999
connectcosts2[srcid]: 54
connectcosts2[srcid] + rcvdpkt->mincost[i]: 86
----------
dt2 was updated. New table below.

-----------------------
               via
   D2 |    1    2    3
------|----------------
     1|   54   999   999
dest 2|  999     0   999
     3|   86   999     7
-----------------------
dt2update mincosts2: 0 1 999 7
dt2update sending out new min costs.
    TOLAYER2: source: 2, dest: 1
              costs:0  1  999  7
    TOLAYER2: scheduling arrival on other side
            INSERTEVENT: time is 0.992243
```

مرحله اشباع:
در این مرحله تمام بهینه‌سازی‌ها در جداول مسافت هر گره اتفاق افتاده. البته در دنیای واقعی این شرایط همیشه پایدار نخواهد بود و به محض آپدیت هزینه یا اضافه شدن مسیری به هر یک از این گره‌ها، دوباره ارتباط‌ها برای بهینه‌سازی جداول مسافت از سر میگیرد.
اما در این سناریو در نهایت به جداول زیر خواهیم رسید:

```
------------------------
                via
   D0 |    1     2    3
------|-----------------
     1|    1   999   15
dest 2|   16   999   14
     3|    9   999    7
------------------------
------------------------
                via
   D1 |    1     2    3
------|-----------------
     1|    0   999  999
dest 2|  999    54   39
     3|  999    61   32
------------------------
------------------------
                via
   D2 |    1     2    3
------|-----------------
     1|   54   999   15
dest 2|  999     0  999
     3|   62   999    7
------------------------
------------------------
                via
   D3 |    1     2    3
------|-----------------
     1|   32     8  999
dest 2|   47     7  999
     3|  999   999    0
------------------------

Simulator terminated at t=17.638222, no packets in medium
```

توضیح توابع کد:

void findmincosts0():

این تابع وظیفه به‌روزرسانی مقادیر موجود در آرایه mincosts0 را بر عهده دارد.
ابتدا این تابع یک حلقه برای هر یک از چهار گره انجام می‌دهد (به جز گره خودش).
سپس برای هر یک از چهار گره، یک حلقه دیگر برای چهار گره انجام می‌دهد (به جز گره خودش).
در این حلقه‌ها، اگر هزینه مسیر موجود در جدول مسافت (dt0.costs[i][j]) کمتر از مقدار فعلی mincosts0[i] باشد، مقدار mincosts0[i] به هزینه جدید بروزرسانی می‌شود.

void sendcosts0:()

این تابع مسئول ارسال هزینه‌های کمینه به همسایگان مستقیم گره است.
ابتدا یک ساختار rtpkt به نام cost_pkt ایجاد می‌کند که حاوی مقادیر sourceid، destid و mincost است. سپس یک حلقه برای هر یک از چهار گره (به جز خودش و گره‌هایی که به طور مستقیم به آن‌ها متصل نیستند) انجام می‌دهد. در این حلقه‌ها، مقدار destid در ساختار cost_pkt تنظیم شده و توابع tolayer2() برای ارسال بسته فراخوانی می‌شود.

void rtinit0:()

این تابع مسئول فراهم‌سازی اولیه جدول مسافت است.
ابتدا یک حلقه برای هر یک از چهار گره انجام می‌دهد. سپس برای هر گره، یک حلقه دیگر برای چهار گره اجرا می‌شود. اگر گره مبدأ و مقصد یکی باشند، هزینه مربوطه از connectcosts0 گرفته می‌شود. در غیر این صورت، هزینه به مقدار 999 تنظیم می‌شود.
پس از تنظیم جدول مسافت، تابع sendcosts0() فراخوانی می‌شود تا هزینه‌های کمینه به همسایگان ارسال شود.
در نهایت، تابع printdt0() فراخوانی می‌شود تا جدول مسافت چاپ شود.

rtupdate0(struct rtpkt *rcvdpkt):

این تابع زمانی فراخوانی می‌شود که یک بسته مسیریابی دریافت شود.
ابتدا اطلاعات بسته مسیریابی دریافتی (مانند شناسه منبع و مقصد و هزینه‌های مسیر) را چاپ می‌کند.
سپس بروزرسانی جدول مسیریابی را انجام می‌دهد. اگر هزینه مسیر جدید کم‌تر از هزینه مسیر قبلی باشد، جدول مسیریابی بروزرسانی می‌شود.
در صورت بروزرسانی جدول مسیریابی، هزینه‌های کمینه مسیر به‌روز شده و به همسایه‌ها ارسال می‌شود.

printdt0:()

این تابع جدول مسیریابی dt0 را چاپ می‌کند.
جدول شامل هزینه مسیرها از هر گره به بقیه گره‌ها است.

linkhandler0(int linkid, int newcost):

این تابع زمانی فراخوانی می‌شود که هزینه یک پیوند تغییر کند. ابتدا هزینه‌های جدید را چاپ می‌کند و سپس جدول مسیریابی را به این شکل بروزرسانی می‌کند:

- هزینه مستقیم پیوند را بروز می‌کند.
- هزینه پیوندهای غیرمستقیم از طریق آن پیوند را به صفر تنظیم می‌کند.
- هزینه‌های کمینه مسیر را بروز و محاسبه می‌کند.

در نهایت جدول مسیریابی بروزرسانی شده را به همسایه‌ها ارسال می‌کند.