



## شبکه‌های کامپیوتری

تمرین سوم

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

نیم سال دوم ۹۹-۰۰

---

استاد:

جناب آقای دکتر جعفری

نام و نام خانوادگی:

امیرمهدی نامجو - ۹۷۱۰۷۲۱۲



## ۱ سوال اول

روش UDP Hole Punching روشی است که به کمک آن می‌توان ارتباط بین دو کلاینت که یک یا هر دوی آن‌ها پشت NAT قرار دارند را برقرار کرد. در اصل این روش به نوعی یک حفره در دیواره NAT ایجاد می‌کند و برای همین Hole Punching نام دارد. نحوه کار این روش بدین صورت است:

فرض کنید می‌خواهیم ارتباط بین A و B را برقرار کنیم. در روش Hole Punching نیاز به داشتن یک واسطه مانند C است که هر دوی A و B آدرس IP آن را بدانند.

در مرحله اول A و B هر دو پکت‌های UDP را به C می‌فرستند. با عبور پکت‌های آنان از NAT شان، این NAT، آدرس IP مبدا این پکت‌ها را بازنویسی می‌کند تا مشخص باشد که پاسخ آن باید به کجا ارسال شود.

در مرحله دوم، C متوجه IP آدرس و همچنین پورت درخواست‌هایی که از سمت A و B آمده‌اند می‌شود. (مثلاً فرض کنید پورت A برابر X و پورت B برابر Y باشد) با توجه به ساختار عمومی NAT، در حال حاضر C می‌تواند به راحتی از این طریق با A و B ارتباط برقرار کند و با ارسال پیام به NAT هر کدام از آن‌ها، از آن جایی که NAT می‌داند که شروع درخواست از سمت قسمت‌های درونی خود بوده است و اطلاعات را دارد، بسته را به درستی به مقصد می‌رساند.

در مرحله بعد، C به A پیامی می‌دهد که می‌گوید برای ارتباط برقرار کردن با B، برای آدرس IP مربوط به NAT آن و پورت Y پیام ارسال کن. از طرفی به B هم می‌گوید برای ارتباط برقرار کردن با A به آدرس IP مربوط به NAT آن و پورت X پیام ارسال کن.

در مرحله بعد، ابتدا اولین پکت‌های ارسالی از سمت A و B به درستی به مقصد نمی‌رسد و توسط NAT‌های مربوطه Reject می‌شود. اما با ارسال اولین پیام از سمت A به B و عبور آن از NAT مربوط به A، این متوجه IrNAT می‌شود که A قصد ارتباط برقرار کردن با IP آدرسی که مربوط به NAT هاست B است و پورت Y را دارد و از این رو پیام‌های دریافتی بعدی از این آدرس را برای A می‌فرستد. همین اتفاق از سمت NAT دیگر هم می‌افتد. از این به بعد این دو NAT می‌دانند درخواست‌هایی که از سمت مقابل می‌آید را باید به کدام یک از Host های سمت خود تحویل بدهند. به نوعی یک حفره در NAT ایجاد شده که درخواست‌هایی که از آدرس خاصی می‌آیند را به درستی به یکدیگر تحویل می‌دهد. بدین ترتیب ارتباط P2P بین A و B برقرار می‌شود.

در اصل اگر بخواهیم به صورت دقیق تر توضیح بدهیم به شکل زیر می‌شود (مطابق توضیحات ویکیپدیا):

۱. هر کدام از A و B ارتباط UDP را با C شروع می‌کنند. NAT های هر کدام یعنی NA و NB دو پورت خارجی موقت EPA و EPB را به این کار اختصاص می‌دهند.

۲. C بسته‌های دریافتی را بررسی می‌کند تا آدرس IP هر کدام از NAT ها و همچنین EPA و EPB را بیابد.

۳. C پیامی حاوی EIPA:EPA را به B و پیامی حاوی EIPB:EPB را به A می‌فرستد.

۴. A یک بسته به EIPB:EPB می‌فرستد.

۵. NAT هاست A بررسی بسته را بررسی می‌کند و در جدول ترجمه اش قرار می‌دهد:

(Source-IP-A , EPA, EIPB , EPB)

که بداند پیام‌های دریافتی از B را باید به کجا بفرستد.



۶. B یک بسته به EPA:EIPA می فرستد.
۷. NAT هاست B بررسی بسته را بررسی می کند و در جدول ترجمه اش قرار می دهد:  
(EPA , EIPA , EPB , Source-IP-B)  
که بداند پیام های دریافتی از A را باید به کجا بفرستد.
۸. بسته به وضعیت NAT هاست A که در هنگام دریافت بسته B، داده مربوط به آن را در جدول ترجمه اش نوشته باشد یا نه، بسته اول دریافتی از B یا رد می شود و یا دریافت می شود.
۹. بسته به وضعیت NAT هاست B که در هنگام دریافت بسته A، داده مربوط به آن را در جدول ترجمه اش نوشته باشد یا نه، بسته اول دریافتی از A یا رد می شود و یا دریافت می شود.
۱۰. در بدترین حالت، هر دو دومین بسته ای که از سمت دیگری می آید را دریافت کرده و ارتباط برقرار می شود.
- به طور کلی این روش امروزه هم در ارتباطات P2P و همچنین VoIP استفاده می شود. با این حال باید چند نکته را در نظر داشت. بعضی از NAT ها در هر بار ارتباط آدرس پورت را عوض می کنند. یعنی حتی اگر پورت و IP مبدا هم یکی باشد، با متفاوت شدن مقصد ها پورت های متفاوتی روی NAT به آن ها اختصاص می یابد. این موضوع در Symmetric NAT ها وجود دارد و باعث ایجاد مشکل در این تکنیک می شود.
- مشکل دیگری که ممکن است پیش بیاید، این است که یک سیستم پشت چند سطح مختلف از NAT ها باشد. در این موارد ممکن است یکی از NAT های لایه بالاتر که به تعداد خیلی زیادی سرویس خدمت رسانی می کند، Port ها را تغییر بدهد و بدون نیاز دائمی به واسط C نتوان ارتباط درست P2P برقرار کرد.
- همچنین یک چالش دیگر زمانی پیش می آید که هر دو سیستم پشت یک NAT باشند. در این مواقع بعضی از NAT ها درخواستی که برای خودشان آمده باشد و Loopback به درون سیستم باشد را به درستی جواب نمی دهند. برای رفع این مشکل، معمولاً به این شکل عمل می شود که اطلاعات IP و Port خصوصی که در ابتدا Host های اولیه قرار داده اند هم از طریق واسط به دیگری فرستاده می شود تا سعی کند از طریق شبکه داخلی هم ارتباط برقرار کند و اگر هر دو متوجه شدند پشت یک NAT هستند صرفاً از طریق شبکه داخلی ارتباط را برقرار کنند و دیگر به آدرس IP و Port عمومی NAT که از بیرون قابل مشاهده بوده چیزی ارسال نکنند.
- روش کار Skype بدین صورت است که از طریق پروتکل هایی نظیر STUN یا ICE متوجه وضعیت NAT هر کدام از کاربرها می شود. تعدادی از کاربران هستند که پشت NAT نیستند و IP عمومی قابل دسترس دارند. این کاربران به نوعی در نقش واسط هایی عمل می کنند که در روش Hole Punching ارتباط بین دو Host دیگر را برقرار می کردند. با توجه به این موضوع خود Skype نیاز دارد که بداند چه کاربرانی پشت NAT هستند و چه کاربرانی نیستند.
- در صورتی که به دلیل ساختار خاص NAT یکی از کاربران نظیر عوض کردن پورت در سیستم Symmetric به هیچ وجه امکان UDP Hole Punching مهیا نباشد، عملاً آن کاربر واسط، تبدیل به نوعی سرور میانی می شود که ارتباط میان دو کاربری که قصد ارتباط برقرار کردن را داشتند را بین آن ها جا به جا می کند. یعنی در این حالت دیگر این واسط، فقط برای ارتباط برقرار کردن اولیه استفاده نمی شود، بلکه همه پیام ها باید یک بار به دست آن واسط رسیده و بعد برای مقصد اصلی ارسال بشوند.



البته در کنار همه این ها باید توجه کرد که یکسری Login Server هم وجود دارد که مقوله آن ها مستقل از برقراری ارتباط است و برای احراز هویت اولیه است. بدیهتا این سرورها به صورت متمرکز در دیتاسنترهای مایکروسافت قرار دارد و اطلاعات احراز هویت دست کاربران مختلف نیست.

همچنین باید به یک نکته دیگر هم توجه کرد و آن هم این که هر چند در پروتکل اولیه استفاده شده توسط Skype، هر کاربری که پشت NAT نبود می توانست در نقش سرورهای برقرارکننده ارتباط قرار بگیرد، اما این موضوع باعث نارضایتی برخی کاربران شده بود که از آن ها به دلیل محدودیت کمتر اینترنتشان و قرار نداشتن پشت NAT به عنوان واسط برقراری ارتباط میان دو کاربر دیگر استفاده می شود. به همین علت بعد از خریداری شدن Skype توسط مایکروسافت، تقریباً همه این Supernode ها که وظیفه ارتباط برقرار کردن بین کاربران را دارند، متشکل از سرورهای اختصاصی قرار گرفته در دیتاسنترهای مایکروسافت هستند و از کاربران برای برقراری UDP Hole Punching استفاده نشده و این وظیفه برعهده سرورهای اختصاصی مایکروسافت است.

در مورد این که آیا نیاز به اطلاع از وجود NAT داریم یا نه، می توان هم جواب بله داد و هم خیر. به طور کلی خود عملیات UDP Hole Punching مستقل از وجود یا عدم وجود NAT در دو سیستمی که قصد ارتباط برقرار کردن دارند، قابل اجراست و در روند انجام فرایند تغییری ایجاد نمی شود. با این حال، به هر حال مسئله پیدا کردن شخص ثالث واسط وجود دارد. در سیستمی نظیر Skype، تا قبل از متمرکز شدن سرورها در دیتاسنترهای مایکروسافت، شخص ثالث واسط هم از کاربران بود و برای پیدا کردن چنین فردی، نیاز به پروتکل های پیمایش NAT بود تا مطمئن بشویم که این فرد پشت NAT نیست و به طور مستقیم IP عمومی دارد. همچنین در مواقعی که چندین لایه NAT وجود دارد، ممکن است به دلیل نحوه تنظیم NAT ها عملاً به طور کلی UDP Hole Punching امکان پذیر نباشد و نیاز باشد که سرور واسط، تمامی پیام ها را بین دو endpoint جا به جا کند. در چنین حالتی، نیاز به دانستن ساختار شبکه و این که NAT وجود دارد یا نه و به چه صورتی تنظیم شده است وجود دارد. در نتیجه هر چند خود نحوه انجام عملیات برای سیستمی که پشت NAT هست یا نه تفاوت چشمگیری ندارد، اما در یک نرم افزار نظیر Skype که قصد پیدا کردن واسطه ها را دارد و همین طور می خواهد در سختگیرانه ترین تنظیمات NAT هم بتواند سرویس دهی کند، نیاز به دانستن این موضوع دارد.

در مورد محدودیت های این تکنیک، ابتدا باید به این نکته اشاره کرد که همان طور که گفته شد، اگر NAT به صورت Symmetric باشد، عملاً امکان انجام این کار وجود ندارد. به علاوه این شیوه متکی بر وجود یک سیستم ثالث است که ارتباط دهی اولیه را برقرار کند و در نتیجه نیاز به نوعی پروتکل یا سرور مرکزی هست که این واسطه ها را پیدا کند. مسئله دیگر در این است که به هر حال در این شیوه روی سیستم های واسطه فشار وارد می شود و عملاً خود آن ها از این لود وارد شده، منفعت و سودی نمی برند. همین موضوع باعث اعتراض برخی کاربران Skype هم شده بود و در نهایت بعد از خریداری آن توسط مایکروسافت، این سرورهای واسطه هم به دیتاسنترهای مایکروسافت منتقل شدند. علاوه بر این طول عمر اتصالات UDP معمولاً خیلی طولانی نیست و در نتیجه باید پکت های Keep-Alive ارسال بشود که از بسته نشدن اتصال اطمینان حاصل بشود. چون در صورت بسته شدن اتصال عملاً ممکن است جدول ترجمه NAT هم دچار تغییر باشد و در نتیجه دوباره نیاز به انجام فرایند UDP Hole Punching باشد.

منابع استفاده شده برای پاسخ این سوال:

ویکی پدیا، ویکی پدیا، Infosec و [bford.info](http://bford.info)



## ۲ سوال دوم

## ۱.۲ الگوریتم ساده

## Link State ۱.۱.۲

جدول آن به صورت زیر می شود. در ستون های مربوط به هر کدام از Node ها عدد اول مربوط به کمترین هزینه تا آن Node و عدد دوم مربوط به Parent آن Node است که با استفاده از آن این کمترین هزینه حاصل می شود.

Step	N'	R2	R3	R4	R5
0	R1	6,R1	2,R1	1,R1	$\infty$
1	R1,R4	3,R4	2,R1	-	4,R4
2	R1,R4,R3	3,R4	-	-	4,R4
3	R1,R4,R3,R2	-	-	-	4,R4
4	R1,R4,R3,R2,R5	-	-	-	-

مقادیر نهایی به صورت زیر می شود:

$$D_{R1}(R2), P_{R1}(R2) = 3, R4$$

$$D_{R1}(R3), P_{R1}(R3) = 2, R1$$

$$D_{R1}(R4), P_{R1}(R4) = 1, R1$$

$$D_{R1}(R5), P_{R1}(R5) = 4, R4$$

## Dsistance Vector ۲.۱.۲

برای این سوال فرض را بر این می گذاریم که در ابتدای کار و تا مرحله استیبل شدن مسیریاب ۵ به درستی کار کرده باشد و بعد غیرفعال شده باشد. چون اگر از ابتدای کار کلا غیرفعال باشد که مانند این است که اصلا حذف شده باشد و از آن جایی که همچنان بقیه متصل هستند، مشکل خاصی پیش نمی آید.

اگر به توپولوژی داده شده دقت کنیم، می بینیم که ۵ فقط با ۲ و ۴ در ارتباط است. بقیه هیچ کدام مستقیماً با ۵ در ارتباط نیستند. از طرف دیگر، ارتباط داشتن ۵ با ۲ و ۴ باعث نمی شود که



تغییری در فاصله کمینه ای که این دو از هم در نظر دارند اتفاق بیفتد. زیرا فاصله این دو از هم ۲ است ولی اگر بخواهند از ۵ استفاده کنند فاصله ۴ می شود. در نتیجه ۵ اثری روی فاصله این دو هم نخواهد گذاشت. در نتیجه در تمامی مراحل بردار فاصله که اطلاعات بین مسیر یاب ها رد و بدل می شود، اطلاعات ۵ اثری روی کمینه فاصله ای که بقیه نسبت به هم دارند نخواهد گذاشت. یعنی ۵ به جز مسیرهایی که به خودش منتهی می شود، در هیچ مرحله ای در هیچ مسیر کمینه ای نیست. در نتیجه در هر مرحله ای که این مسیر یاب به مشکل بخورد، به جز این که بدیهتا مسیر یابی به خود آن دچار مشکل می شود، برای بقیه مسیرها هیچ مشکلی پیش نمی آید.

پس در این مورد خاص مشکلی پیش نیامد. اما فرض کنیم که واقعا خرابی ۵ باعث مشکل در مسیر می شد. باید راهکار را برای این حالت بررسی کنیم. با خرابی ۵ در ابتدا ممکن است بقیه فکر کنند که چون تغییری در بردارهای مربوط به آن ایجاد نشده پیامی نمی آید. اما به هر حال در یک مرحله ای یکی از مسیر یاب های متصل به ۵ (مثلا با نام A) یا سعی می کند بردار آپدیت شده خود را به ۵ بفرستد یا این که سعی می کند که داده ای که به مقصد ۵ است را به آن بفرستد. در این زمان متوجه می شود که ۵ پاسخگو نیست. در این حالت می تواند وزن مربوط به ۵ را در بردار خود بی نهایت قرار بدهد. البته باید توجه کرد که اگر چند گره دیگر هم به ۵ مسیر داشته باشند (مانند همین توپولوژی سوال)، ممکن است گره ما فرض کند شاید فقط لینک خودش خراب شده است و سعی کند از طریق بقیه انتقال را انجام دهد. در این حالت نیز Poisoned Reverse چاره کار خواهد بود. یعنی می تواند سعی کند از طریق گره های دیگری که به ۵ متصل هستند این کار را انجام دهد ولی حداقل به همان گره که از طریق آن مسیر یابی را انجام می دهد، اعلام می کند که فاصله خودش تا ۵ بی نهایت است که گره دوم سعی نکند از طریق خود A مسیر یابی به ۵ انجام بدهد. بدین ترتیب به تدریج گره های دیگری که به ۵ متصل بودند هم متوجه می شوند که امکان ارتباط با ۵ را ندارند.

یک راه بهتر برای رفع مشکل هم این است که صرفا منتظر نباشیم که ببینیم چه زمانی پیامی به مقصد ۵ می آید و آن زمان وضعیت ۵ را چک کنیم. بلکه همه گره ها مستقل از پیام معمول آپدیت بردارها و پیام هایی که بنا بر وظیفه جا به جا می کنند، در بازه های زمانی مشخص به گره های همسایه خود از طریق لینک مربوطه، یکسری پیام Hello به یکدیگر بفرستند تا مطمئن شوند که لینک و گره دیگر سالم است و در صورتی که دیدند پاسخی دریافت نمی شود، سعی کنند مسیر یابی را از طریق یکی دیگر از همسایه های خود به آن لینک انجام دهند و به آن همسایه هم اعلام کنند که مسیرشان تا گره احتمالا شکست خورده بی نهایت است که دور پیش نیاید. بدین ترتیب با ضریب اطمینان بیش تری مطمئن خواهیم بود که بعد از مدت زمان تعیین شده و کوتاهی، همگی متوجه خرابی آن گره خواهند شد. در حالت قبلی تا زمانی که نیازی به ارسال پیام به آن گره احساس نمی شد، متوجه خرابی آن نمی شدیم ولی در این روش که به صورت دوره ای همه گره ها و لینک ها چک می شوند، در صورت خرابی یک گره به سرعت همگی متوجه خواهند شد.

در کل بهترین روش ترکیبی از Reverse Poisoned و خبر گرفتن هر گره از همسایه هایش در بازه های زمانی مشخص است که از فیل شدن لینک (و احتمالا خود گره) مطمئن شویم.

در صفحه بعد مراحل اجرای الگوریتم (بدون در نظر گرفتن Poisoned Reverse) قرار دارند. مرحله بعدی درست مشابه مرحله آخر خواهد بود و بدین دلیل دوباره نوشته نشده است. (مراحل با فرض فیل نشدن ۵ رسم شده اند تا نشان داده شود که ۵ تاثیری در مسیرهای پیدا شده ندارد)



R1					
	R1	R2	R3	R4	R5
R1	0	6,R1	2,R1	1,R1	-
R2	-	-	-	-	-
R3	-	-	-	-	-
R4	-	-	-	-	-
R5	-	-	-	-	-

R2					
	R1	R2	R3	R4	R5
R1	-	-	-	-	-
R2	6,R2	0,R2	-	2,R2	1,R2
R3	-	-	-	-	-
R4	-	-	-	-	-
R5	-	-	-	-	-

R3					
	R1	a	R3	R4	R5
R1	-	-	-	-	-
R2	-	-	-	-	-
R3	2,R3	-	0,R3	4,R3	-
R4	-	-	-	-	-
R5	-	-	-	-	-

R4					
	R1	R2	R3	R4	R5
R1	-	-	-	-	-
R2	-	-	-	-	-
R3	-	-	-	-	-
R4	1,R4	2,R4	4,R4	0,R4	3,R4
R5	-	-	-	-	-

R5					
	R1	R2	R3	R4	R5
R1	-	-	-	-	-
R2	-	-	-	-	-
R3	-	-	-	-	-
R4	-	-	-	-	-
R5	-	1,R5	-	3,R5	0,R5

TimeStep 0					
------------	--	--	--	--	--



R1					
	R1	R2	R3	R4	R5
R1	0	3,R4	2,R1	1,R1	4,R4
R2	6,R2	0,R2	-	2,R2	1,R2
R3	2,R3	-	0,R3	4,R3	-
R4	1,R4	2,R4	4,R4	0,R4	3,R4
R5	-	-	-	-	-

R2					
	R1	R2	R3	R4	R5
R1	0	6,R1	2,R1	1,R1	-
R2	3,R4	0,R2	6,R4	2,R4	1,R5
R3	-	-	-	-	-
R4	1,R4	2,R4	4,R4	0,R4	3,R4
R5	-	1,R5	-	3,R5	0,R5

R3					
	R1	R2	R3	R4	R5
R1	0	6,R1	2,R1	1,R1	-
R2	-	-	-	-	-
R3	2,R3	6,R4	0,R3	3,R1	7,R4
R4	1,R4	2,R4	4,R4	0,R4	3,R4
R5	-	-	-	-	-

R4					
	R1	R2	R3	R4	R5
R1	0	6,R1	2,R1	1,R1	-
R2	6,R2	0,R2	-	2,R2	1,R2
R3	2,R3	-	0,R3	4,R3	-
R4	1,R4	2,R4	3,R3	0,R4	3,R4
R5	-	1,R5	-	3,R5	0,R5

R5					
	R1	R2	R3	R4	R5
R1	-	-	-	-	-
R2	6,R2	0,R2	-	2,R2	1,R2
R3	-	-	-	-	-
R4	1,R4	2,R4	4,R4	0,R4	3,R4
R5	4,R4	1,R5	7,R4	3,R5	0,R5

TimeStep 1					
------------	--	--	--	--	--





R1					
	R1	R2	R3	R4	R5
R1	0,R1	3,R4	2,R1	1,R4	4,R4
R2	3,R4	0,R2	6,R4	2,R4	1,R5
R3	2,R3	6,R4	0,R3	3,R1	7,R4
R4	1,R4	2,R4	3,R3	0,R4	3,R4
R5	-	-	-	-	-

R2					
	R1	R2	R3	R4	R5
R1	0	6,R1	2,R1	1,R1	-
R2	3,R4	0,R2	5,R4	2,R2	1,R5
R3	-	-	-	-	-
R4	1,R4	2,R2	3,R3	0,R4	3,R4
R5	4,R4	1,R2	7,R4	3,R4	0,R5

R3					
	R1	R2	R3	R4	R5
R1	0	3,R4	2,R1	1,R1	4,R4
R2	-	-	-	-	-
R3	2,R3	5,R1	0,R3	3,R1	6,R1
R4	1,R4	2,R4	3,R3	0,R4	3,R4
R5	-	-	-	-	-

R4					
	R1	R2	R3	R4	R5
R1	0	6,R1	2,R1	1,R1	-
R2	3,R4	0,R2	6,R4	2,R4	1,R5
R3	2,R3	6,R4	0,R3	3,R1	7,R4
R4	1,R4	2,R4	3,R1	0,R4	3,R4
R5	4,R4	1,R2	7,R4	3,R4	0,R5

R5					
	R1	R2	R3	R4	R5
R1	-	-	-	-	-
R2	3,R4	0,R2	6,R4	2,R4	1,R5
R3	-	-	-	-	-
R4	1,R4	2,R4	4,R4	0,R4	3,R4
R5	4,R4	1,R5	7,R4	3,R5	0,R5

TimeStep 2					
------------	--	--	--	--	--



R1					
	R1	R2	R3	R4	R5
R1	0,R1	3,R4	2,R1	1,R4	4,R4
R2	3,R4	0,R2	5,R4	2,R2	1,R5
R3	2,R3	5,R1	0,R3	3,R1	6,R1
R4	1,R4	2,R4	3,R1	0,R4	3,R4
R5	-	-	-	-	-

R2					
	R1	R2	R3	R4	R5
R1	0,R1	3,R4	2,R1	1,R4	4,R4
R2	3,R4	0,R2	5,R4	2,R2	1,R5
R3	-	-	-	-	-
R4	1,R4	2,R4	3,R1	0,R4	3,R4
R5	4,R4	1,R2	7,R4	3,R4	0,R5

R3					
	R1	R2	R3	R4	R5
R1	0,R1	3,R4	2,R1	1,R4	4,R4
R2	-	-	-	-	-
R3	2,R3	5,R1	0,R3	3,R1	6,R1
R4	1,R4	2,R4	3,R1	0,R4	3,R4
R5	-	-	-	-	-

R4					
	R1	R2	R3	R4	R5
R1	0,R1	3,R4	2,R1	1,R4	4,R4
R2	3,R4	0,R2	5,R4	2,R2	1,R5
R3	2,R3	6,R4	0,R3	3,R1	7,R4
R4	1,R4	2,R4	3,R1	0,R4	3,R4
R5	4,R4	1,R2	7,R4	3,R4	0,R5

R5					
	R1	R2	R3	R4	R5
R1	-	-	-	-	-
R2	3,R4	0,R2	5,R4	2,R2	1,R5
R3	-	-	-	-	-
R4	1,R4	2,R4	3,R1	0,R4	3,R4
R5	4,R4	1,R5	6,R4	3,R5	0,R5

TimeStep 3					
------------	--	--	--	--	--



RIP ۲.۲

(آ)

$$A \rightarrow D \rightarrow E$$

(ب)

$$C \rightarrow E \rightarrow D$$

(ج) گراف شبکه به صورت زیر خواهد بود:

