



## اهداف تمرین

- آشنایی با ارتباطات P2P
- آشنایی با لایه سوم و چهارم شبکه، مسیریابی و پروتکل UDP
- آشنایی با NAT ، NAT Traversal و روش Hole Punching

### ۱. مقدمه

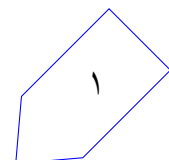
همانطور که می‌دانید اعضای شبکه مستقیماً باهم در ارتباط نیستند. از این رو هر عضو شبکه برای ارتباط برقرار کردن با عضوی دیگر مجبور است بسته‌اش را از مسیری که شامل اعضای میانی است بگذراند. کار برخی از اعضای مسیر، مسیریابی است و صرفاً بسته را بدون تغییر به گرهی بعدی انتقال می‌دهند. اما اعضای هستند که بسته را بازرسی و فیلتر می‌کنند و در برخی مواقع بسته را تغییر می‌دهند. به طور کلی به این اعضا Middle Box می‌گوییم.

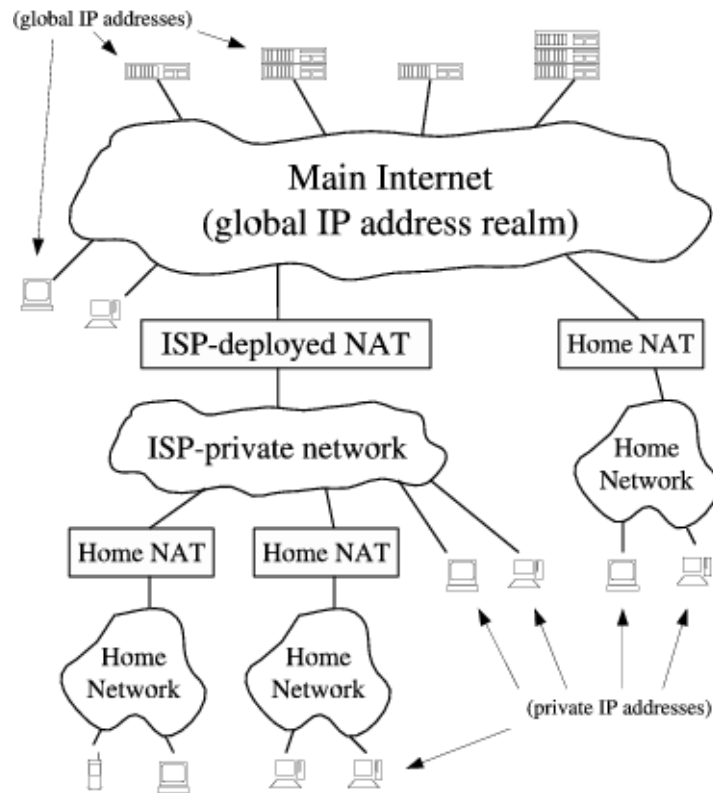
حالت خاصی از Middle Box ها، Network address translation (NAT) ها هستند. این اعضاء آدرس مبدا و یا مقصد را تغییر می‌دهند، بنابراین گره‌هایی که پشت NAT قرار دارند، از دید اعضای دیگر NAT آدرس متفاوتی دارند و بعضاً ممکن است دو گره شبکه یک آدرس داشته باشند. مشکلی که اینجا پیش می‌آید این است که اگر دو همتا<sup>۱</sup> بخواهند به یکدیگر بسته بفرستند و یک ارتباط همتا به همتا<sup>۲</sup> را ایجاد کنند، چگونه آدرس همدیگر را پیدا کنند و به نوعی محدودیت‌های NAT را دور بزنند. روش‌های مختلفی برای اینکار وجود دارد که به طور کلی به این مساله NAT Traversal می‌گویند. در حالت کلی شبکه به شکل زیر است:

\* با سپاس از مهدی بهروزی‌خواه، نیما کریمی‌پور و سولماز سلیمی

<sup>1</sup>Peer

<sup>2</sup>P2P





شکل ۱: NAT with network Sample

۳

## ۲. مقدمه‌ای بر NAT

امروزه استفاده از NATها بسیار رایج شده است و اکثر همتاها پشت چند لایه از NAT قرار دارند. کار مهم NAT این است که آدرس و یا درگاه بسته‌ی وارد شده را تغییر می‌دهد و یا ترجمه کند و سپس بسته را به گره بعدی بدهد. NAT با توجه به کاری که می‌کند به چند دسته مختلف تقسیم می‌شوند که رایج‌ترین آن‌ها Outbound NAT است. این دسته در حالت عادی برای بسته‌های وارد شده آدرس محلی را به آدرس عمومی ترجمه می‌کند. اما بسته‌هایی که از خارج وارد می‌شوند را تنها در صورتی به آدرس محلی تبدیل می‌کند که از قبل نشستی<sup>۴</sup> با این آدرس از داخل شبکه محلی ایجاد شده باشد. Outbound NAT خود به دو زیر دسته تقسیم می‌شود:

Basic NAT: تنها آدرس را ترجمه می‌کنند

Network/Port Translation (NAPT): که هم آدرس و هم درگاه را ترجمه می‌کنند و حالت کلی‌تری هستند و این امکان را می‌دهند تا همزمان چند همتا از یک آدرس عمومی روی درگاه‌های مختلف استفاده کنند. متأسفانه اکثر حالت‌های NAT باعث می‌شود اگر هر دو همتا پشت یک NAT جدا باشند، هیچ کدام نتوانند با

<sup>3</sup> <http://www.justsysadmin.net/es/theoretical/nat/>

<sup>۴</sup> یک نشست یا Session در پروتکل UDP و TCP تشکیل شده از یک چهارتایی (Local IP, Local Port, Remote IP, Remote Port) است.

دیگری نشستی ایجاد کنند، زیرا NAT همتای مقابل بسته وارد شده از خارج را دور می‌ریزد. بدین ترتیب تنها راه ارتباطی حالتی است که یک همتا شروع کننده نشست باشد و همتای دیگر پشت NAT نباشد. اما با روش‌هایی می‌توان بین دو همتای پشت NAT نیز ارتباط برقرار کرد. یکی از این روش‌ها روش UDP Hole Punching است.

### ۳. مقدمه‌ای بر UDP Hole Punching

روش Hole Punching یکی از روش‌هاییست که برای عبور از NAT استفاده می‌شود. ایده کلی این روش کمک گرفتن از یک کارگزار شناخته شده برای دو همتا است. ابتدا کفایت هر دو با کارگزار یک نشست درست کنند. سپس کارگزار به ازای هر همتا دو آدرس و دو درگاه نگهداری می‌کند. یکی آدرس و درگاهی است که خود همتا بسته را با آن فرستاده بود که به آن اطلاعات محلی می‌گوییم. دیگری آدرس و درگاهی است که کارگزار هنگام دریافت بسته در سرآیندهای بسته می‌بیند که به آن اطلاعات عمومی می‌گوییم. اطلاعات محلی را باید خود همتاها بفرستند زیرا پس از عبور از NAT دیگر قابل شناسایی نیست. اما اطلاعات عمومی از سرآیندها معلوم است. پس از این کارگزار تمام اطلاعات هر دو همتا را برایشان می‌فرستد. هر کدام از همتاها با هر دو اطلاعات محلی و عمومی تلاش به برقراری یک نشست جدید می‌کنند. هر نشستی که زودتر انجام شد، همان را ادامه می‌دهند.

برای مثال فرض کنید دو همتای A و B داریم که هر دو پشت NAT اند. و کارگزاری مانند S وجود دارد که پشت NAT نیست و هر دو A و B از قبل با S یک نشست UDP ایجاد کرده اند. حال A در تلاش است تا با B یک نشست UDP ایجاد کند. پس مراحل زیر را انجام می‌دهند:

۱. A آدرس B را نمیداند، پس از S می‌پرسد.

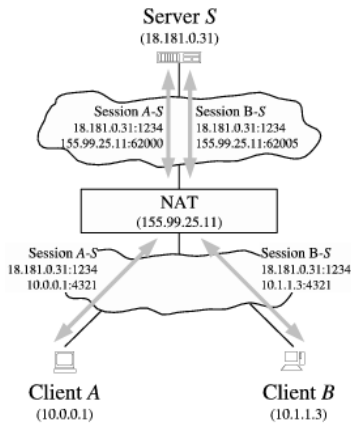
۲. S بسته ای به A شامل آدرس و درگاه محلی همچنین آدرس و درگاه عمومی B می‌فرستد. به طور همزمان اطلاعات A را نیز به B می‌فرستد تا هر دو همدیگر را بشناسند.

۳. هنگامی که A اطلاعات را از S دریافت کرد، به هر دو آدرس محلی و عمومی B بسته اش را می‌فرستد و منتظر میشود تا یکی از این دو آدرس جوابش را بدهند و با آن ارتباط را ادامه دهد. به طور موازی B نیز هنگامی که بسته را دریافت کرد، به هر دو آدرس محلی و عمومی A بسته را می‌فرستد و منتظر میشود تا از یکی پاسخ بگیرد.

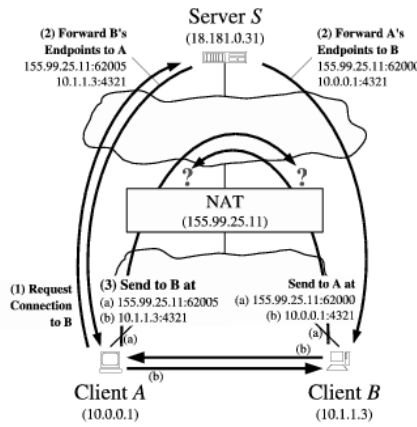
با توجه به سناریو بالا، سه حالت مختلف برای A و B پیش می‌آید که در ادامه به بررسی آنها می‌پردازیم.

#### ۱.۳ NAT مشترک

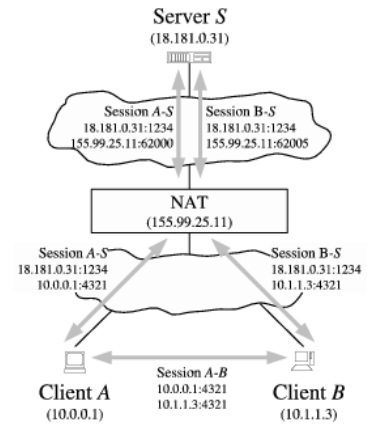
در این حالت هر دو همتا پشت یک NAT مشترک هستند. همانطور که در شکل می‌بینید، دو همتا پس از دریافت اطلاعات از S، از طریق آدرس عمومی و محلی یکدیگر تلاش برای برقراری ارتباط می‌کنند. در اینجا چون هر دو در یک شبکه محلی هستند، مسیر محلی زودتر شکل می‌گیرد و در نتیجه نشست از طریق ارتباط محلی برقرار می‌شود.



Before Hole Punching



The Hole Punching Process

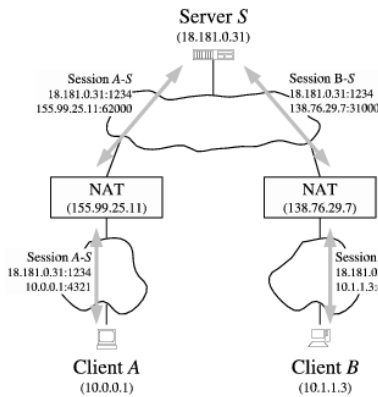


After Hole Punching

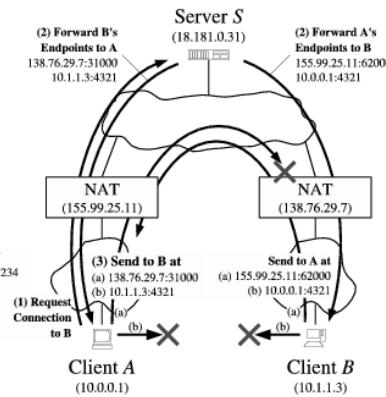
۵

### ۲.۳ NAT متفاوت

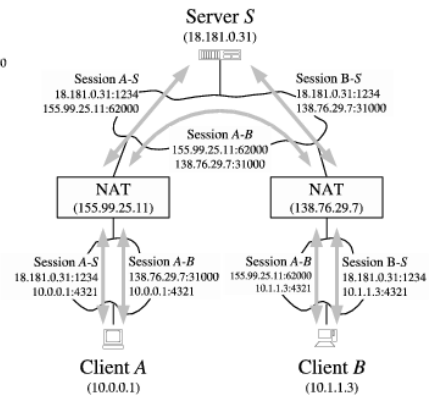
در این حالت دو همتا پشت یک لایه NAT متفاوت هستند. شکل زیر را مشاهده کنید.



Before Hole Punching



The Hole Punching Process



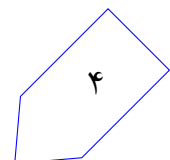
After Hole Punching

۶

از این رو از طریق اتصالات محلی نمیتوانند ارتباط برقرار کنند. پس هنگامی که A بسته درخواست را از طریق NAT به B می فرستد، در جدول NAT ای که A پشت آن است یک خانه جدید با اطلاعات این نشست (شامل آدرس و درگاه A و آدرس و درگاه B) ایجاد می کند. اما هنگامی که بسته به NAT ای که B پشت آن است می رسد، ممکن است عبور نکند. اگر بسته ای که B به سمت A فرستاده، هنوز به NAT B نرسیده باشد، جدول NAT B بروز نشده است، در نتیجه بسته A عبور نمی کند، اما اگر بسته B زودتر رسیده باشد، بسته A نیز عبور می کند. اما در هر صورت، اگر هر دو NAT به صورت ترتیبی بسته ها را پردازش کنند، یکی از دو بسته به مقصد می رسد و ارتباط برقرار می شود.

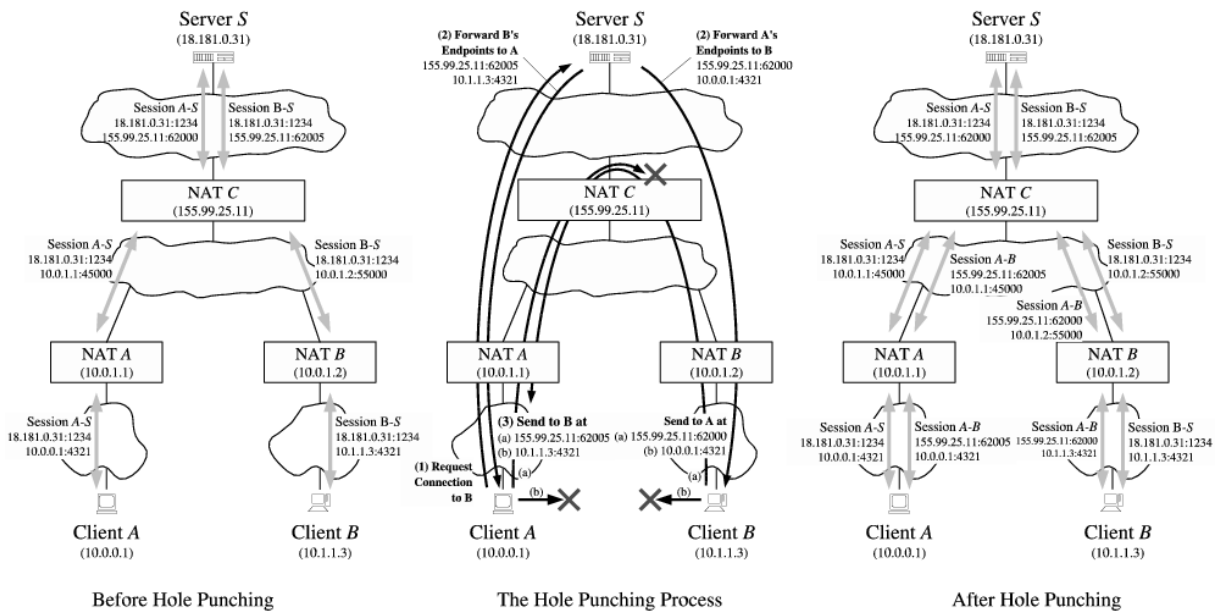
<sup>5</sup> <http://www.brynosaurus.com/pub/net/p2pnat/>

<sup>6</sup> <http://www.brynosaurus.com/pub/net/p2pnat/>



### ۳.۳ NAT چندلایه

در این حالت دو همتا پشت چندلایه NAT متفاوت هستند. شکل زیر را مشاهده کنید.



v

در این حالت نیز ارتباط محلی پاسخگو نیست. این حالت مانند حالت قبلی است، با این تفاوت که مسیر بسته‌ها تا بالاترین گره‌ای است که شبکه‌های دو همتا را بهم مرتبط کند و آدرس‌هایشان معنی دار می‌شود. در گره پایانی، عملیات ترجمه انجام می‌شود و دوباره بسته به شبکه داخلی باز می‌گردد.

### ۴.۳ توپولوژی شبکه

توپولوژی این سوال به صورت درختی است. هر عضو شبکه (بجز کارگزار) از طریق واسط شماره ۰ خود به شبکه عمومی‌تری متصل است. در نتیجه، هر عضو شبکه اگر مقصد بسته‌ای را نیافت، از واسط شماره ۰ خود به بیرون می‌فرستد بدون اینکه بررسی کند آیا mask شبکه واسط شماره ۰ برای این بسته است درست است یا خیر. برای گره‌های NAT، دنیای بیرونی و آدرس‌های عمومی همه در واسط شماره ۰ آن قرار دارند و واسط‌های دیگر آن هر کدام ممکن است مربوط به یک شبکه مجزا باشند.

در ریشه این درخت، کارگزار قرار دارد. تضمین می‌شود که کارگزار در دنیای اینترنت قرار دارد و پشت هیچ NAT ای نیست. همچنین کارگزار همیشه در آدرس و درگاه 1.1.1.1:1234 قرار دارد و شما موظفید تنها به/از این آدرس و درگاه بسته بفرستید، حال آنکه واسط‌های دیگری با آدرس‌های دیگری نیز داشته باشد. ممکن است ساختار شبکه به گونه‌ای باشد که این درخت، در ظاهر درخت نباشد، یعنی اتصالات آن شامل دور باشد، اما با توجه به الگوریتم مسیریابی و ساختار mask، برای هر بسته تنها یک واسط خروجی وجود خواهد داشت. تضمین می‌شود در صورت مسیریابی درست، هرگز دوری ایجاد نخواهد شد.

<sup>7</sup> <http://www.brynosaurus.com/pub/net/p2pnat/>

شبکه های محلی، می توانند IP های مشترکی داشته باشند، زیرا از هم جدا هستند، از این رو، در تست ها برای راحتی شما، سعی شده هیچ نشست یکسانی تشکیل نشود، یعنی دو IP یکسان، حتما روی درگاه ها متفاوتی گوش<sup>۸</sup> می کنند.

ساختار شبکه به گونه ای است که، حداکثر ۱۵ عضو در شبکه حضور دارند. در این میان، ممکن است کاربر شما پشت چند لایه NAT باشد بنابراین ممکن است یک بسته برای رسیدن به مقصد چندین بار دست خوش تغییر قرار بگیرد.

## ۴. توضیح تمرین

هدف این تمرین پیاده سازی روش UDP Hole Punching است که در مقدمه به طور کلی توضیح داده شد. بنابراین شما در یک شبکه قرار دارید که سه نوع گره (سرویس دهنده NAT، کارگزار<sup>۹</sup>، کارخواه<sup>۱۰</sup>) دارد. پس شما در نقش کارخواه باید دو کارخواهی که پشت NAT هستند و از توپولوژی شبکه خبر ندارند را بهم وصل کنید. در صورتی که بسته ای به کارخواهی وارد شد، باید آن را از طریق IP مسیریابی کند و به گره بعدی برساند. همچنین در نقش کارگزار باید بسته هایی را دریافت و ذخیره کنید و سپس بسته ها را مسیریابی کرده و برای کارخواهان دیگر بفرستید. دقت کنید که برای کمتر شدن حجم کار، کد سرویس دهنده NAT بصورت اجرایی در اختیار شما قرار خواهد گرفت. اما برای قسمت امتیازی نیاز است این بخش را نیز خودتان پیاده سازی کنید.

### ۱.۴. انواع بسته ها

تمامی بسته هایی که در این تمرین تولید و بین گره ها جابه جا می شوند، ساختار زیر را دارند و شما موظفید تمام این قسمت ها را پر کنید:

Ethernet	IP Header	UDP Header	Data Type + ID	Data
14 Byte	20 Byte	8 Byte	1 Byte(3+5 bit)	Not Fixed

جدول ۱: ساختار بسته ها

#### ۱.۱.۴. Ethernet:

آدرس مبدأ را آدرسی که در Interface شما نوشته شده است بگذارید، آدرس مقصد را Broadcast و Type را برابر IP (۰x۰۸۰۰) قرار دهید.<sup>۱۱</sup>

<sup>8</sup>Listen

<sup>9</sup>Server

<sup>10</sup>Client

<sup>11</sup>[https://en.wikipedia.org/wiki/Ethernet\\_frame#Ethernet\\_II](https://en.wikipedia.org/wiki/Ethernet_frame#Ethernet_II)

#### ۲.۱.۴ .IP Header

- تک تک قسمت‌ها را باید مطابق با استاندارد IPv4 پر کنید.<sup>۱۲</sup>
- دقت کنید که هر گره ممکن است چند Interface داشته باشد که در هر کدام از آن‌ها یک آدرس مخصوص به خود را دارد. قسمت مربوط به Source IP address تمام بسته‌ها هنگام ساخت، باید برابر IP این گره در Interface شماره ۰ این گره باشد. سپس گره‌های میانی دیگر کاری با این قسمت ندارند (مگر سرویس دهنده‌های NAT).

- DSCP ، ECN ، Identification ، Flags و Fragment Offset را با 0 پر کنید.
- Checksum پر کردن این قسمت و درست بودن آن برای تمام بسته‌ها الزامیست.
- Protocol پروتکل لایه بالایی را باید در این قسمت قرار دهید. چون تمام بسته‌ها UDP هستند، پس با 17 پر کنید.
- TTL را با ۶۴ شروع کنید و پس از دریافت هر بسته، یک واحد از آن کم کرده و سپس به بعدی انتقال دهید. نکته: کم کردن TTL باعث بی‌اعتباری Checksum می‌شود، پس باید این قسمت را نیز هر بار بروز کنید.

#### ۳.۱.۴ .UDP Header

- مطابق با درگاه‌های گفته شده و طول پیام ارسالی جزئیات این قسمت را پر کنید.<sup>۱۳</sup>
- Checksum را همیشه برابر با ۰ قرار دهید.

#### ۴.۱.۴ .Data Type, Data

با توجه به نوع بسته، محتویات بسته متفاوت خواهد بود. در ادامه جزییات این قسمت به صورت دو جدول آمده است. دقت کنید که طول Data شما همیشه ثابت نیست و با توجه به جدول می‌توانید طول هر بسته را حساب کنید، کافیت ببینید تا چند بایت اول برای هر بسته مورد نیاز است.

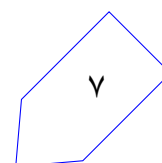
#### ۲.۴ .کارخواه

هر کارخواه چند وظیفه بر عهده دارد.

<sup>12</sup><https://en.wikipedia.org/wiki/IPv4#Header>

<sup>13</sup>[https://en.wikipedia.org/wiki/User\\_Datagram\\_Protocol#Packet\\_structure](https://en.wikipedia.org/wiki/User_Datagram_Protocol#Packet_structure)

<sup>14</sup> طول این پیام ممکن است تا ۵۰ بایت باشد



Name	Sender	Receiver	Data Type
Request assigning ID	Client	Server	000
Response assigning ID	Server	Client	000
Drop	NAT	Client	000
Request getting IP	Client	Server	001
Response getting IP	Server	Client	001
Request local session	Client	Client	010
Response local session	Client	Client	010
Request public session	Client	Client	010
Response public session	Client	Client	010
Message	Client	Client	011
NAT Updated	NAT	Client	100
Request updating info	Client	Server	101
Status	Client	Server	110
Status Respond	Server	Client	111

جدول ۲: انواع بسته‌ها

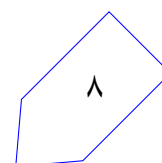
- مسیریابی و رد و بدل کردن تمام بسته‌هایی که از گره‌های همسایه می‌گیرد. هر کارخواه باید روی تمام واسط‌ها ۱۵ گوش کند و بسته‌هایی که وارد می‌شوند را پردازش کند. در صورتی که آدرس مقصد بسته مربوط به این کارخواه نباشد، باید بسته را با توجه به Mask هر واسط مسیریابی کند و روی آن واسط بفرستد. در صورتی که بسته مسیریابی نمی‌شود، بسته را از واسط شماره ۰ خود به بیرون می‌فرستد.

- پردازش دستور

make a connection to server on port X

که X یک عدد صحیح در بازه (1000, 5000) است. هنگامی که دستور بالا وارد شد، باید یک بسته از نوع Request assigning ID روی درگاه X فرستاده شود. نکته: دقت کنید ازین جا به بعد همه بسته‌های خروجی این گره باید روی درگاه X فرستاده شوند (مگر اینکه بسته drop دریافت شود) تا بتواند NAT را دور بزند.

<sup>15</sup>Interfaces





Data Type	Data				
3 bit(MSB)	5 bit	4 Byte	2 Byte	4 Byte	2 Byte
000	0	Local IP	Local Port	-	-
000	New ID	-	-	-	-
000	0	drop	-	-	-
001	Dest ID	-	-	-	-
001	Dest ID	Local IP	Local Port	Public IP	Public Port
010	Src ID	ping	-	-	-
010	Src ID	pong	-	-	-
010	Src ID	ping	-	-	-
010	Src ID	pong	-	-	-
011	Src ID	Message <sup>14</sup>			
100	-	-	-	-	-
101	0	Local IP	Local Port	-	-
110	0	Local IP	Local Port	-	-
111	FLAG	0	0	-	-

جدول ۳: جزئیات بسته‌ها

درون بسته، Local IP و Local Port واسط شماره ۰ خودش را قرار می‌دهد. پس از این بسته را روی واسط مربوطه به کارگزار شبکه ارسال می‌کند. با این کار کارگزار یک ID به این گره اختصاص می‌دهد و آن را برای این گره می‌فرستد.

ممکن است کارگزار NAT این پورت مبدأ را مسدود کرده باشد و بسته شما به مقصد نرسد. در صورتی که این اتفاق افتاده باشد یک بسته drop از NAT مربوطه دریافت می‌کنید.

- دریافت بسته Drop: در صورتی که این بسته با درگاه 1234 دریافت شود، یعنی درخواست برای گرفتن ID نافرجام مانده و باید درگاه دیگری را برای فرستادن بسته انتخاب کنید. برای تلاش مجدد، عدد درگاه قبلی را با ۱۰۰ جمع کرده و سپس دوباره همان بسته حالت قبل را ارسال کنید و در خروجی پیام زیر را چاپ کنید.

connection to server failed, retrying on port X

که X برابر با Last Source Port + 100 است.

نکته: ممکن است این فرآیند چندین بار تکرار شود و شما مجبور باشید هر بار درگاه را افزایش دهید. تضمین

می‌شود که در تمام تست‌ها حداکثر بعد از چند بار تلاش بسته شما از تمام NAT های میانی عبور خواهد کرد و به کارگزار خواهد رسید.

در صورتی که این بسته با درگاه 4321 دریافت شود، یعنی یعنی پکت ارسالی توسط NAT از ادامه مسیر یابی باز مانده است. در صورتی که با این ID قبلا در ارتباط بودید پیام زیر را چاپ کنید.

```
connection lost, perhaps ID's info has changed, ask server for updates
```

- دریافت بسته Response assigning ID شامل ID خود از کارگزار و چاپ عبارت

```
Now My ID is X
```

که X برابر عددی است که کارگزار فرستاده است.

- پردازش دستور

```
get info of ID
```

که ID یک عدد صحیح در بازه (0, 31) است. هنگامی که این عبارت وارد شد، باید بسته Request getting IP با ID وارد شده تولید شود. پس از آن از طریق واسط مربوطه به تنها کارگزار شبکه ارسال شود. نکته: همانطور که گفته شد، تضمین می‌شود که در تست‌ها هر کارخواه تنها از واسط شماره ۰ به کارگزار مسیر دارد و ساختار شبکه مانند درختی است که همه از طریق واسط ۰ به پدرشان که شبکه عمومی تری است وصل شده اند.

- دریافت بسته Response getting IP و چاپ عبارت

```
packet with (ID, LocalIP, LocalPort, PublicIP, PublicPort) received
```

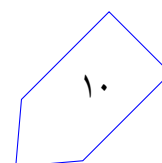
عبارات داخل پرانتز را باید از بسته بخوانید و در قسمت مربوطه قرار دهید.

- دریافت دستور

```
make a local session to ID
```

یا

```
make a public session to ID
```



یک بسته Request local/public session تولید کرده، و ID خود و پیام ping را درون آن قرار می‌دهد و می‌فرستد. اطلاعات مقصد را با آدرس و درگاه محلی یا عمومی (با توجه به دستور وارد شده) گره مورد نظر پر می‌کند.

نکته: در صورتی که اطلاعات ID مورد نظر هنوز دریافت نشده است، عبارت

```
info of node ID was not received
```

را چاپ کنید و بسته‌ای ارسال نکنید.

- دریافت بسته Request local/public session با پیام ping و چاپ عبارت

```
Connected to ID
```

تنها در صورتی که مقصد بسته متعلق به این گره باشد. پس همانطور که می‌بینید اتصال دو طرفه است و وقتی یک نشست برقرار می‌شود هر دو طرف بهم متصل می‌شوند. سپس باید یک بسته نوع ۳ مانند بسته دریافتی درست کند و در جواب بفرستد، با این تفاوت که اینبار پیام را برابر pong قرار می‌دهد و جای فرستنده و گیرنده را عوض می‌کند (ارسال بسته Response local/public session)

- دریافت بسته Response local/public session با پیام pong و چاپ عبارت

```
Connected to ID
```

اگر و تنها اگر بسته ping ای از قبل برای این ID فرستاده شده بود و pong ای دریافت نشده بود.

- پردازش دستور

```
send msg to ID:msg
```

که ID مقصد و پیامی که می‌خواهید را وارد می‌کنید.

نکته: شما باید بتوانید هر پیامی با طول کمتر از ۵۰ حرف شامل "حروف الفبا، اعداد و فاصله" را ارسال کنید.

- دریافت بسته NAT update باید بسته info update به همان نحوه ی پر شدن بسته request assigning id پر شود به سمت کارگزار فرستاده شود

- پردازش دستور

```
status
```

با دریافت این دستور باید بسته Status به کارگزر فرستاده شود.

- دریافت بسته Status Respond این بسته از کارگزر پس از اجرای دستور، Status. دریافت میشود این بسته، حاوی FLAG است که نشان دهنده نوع ارتباط با کارگزار است، در صورتی که FLAG برابر با ۱ بود باید عبارت

direct

ور در صورتی که ۰ بود باید عبارت

indirect

چاپ بشود

### ۳.۴ کارگزار

در این تمرین تنها یک کارگزار داریم که در آدرس عمومی 1.1.1.1 قرار دارد و به درگاه 1234 گوش می‌کند. شما به عنوان کارخواه باید بسته‌های خود را به این آدرس و درگاه بفرستید. همچنین مبدا تمام بسته‌هایی که از طریق کارگزار ایجاد و ارسال می‌شوند، باید با این آدرس و درگاه پر شوند. کارهای کارگزار به ترتیب در زیر آمده‌اند:

- دریافت بسته Request assigning ID از کارخواهان و سپس اختصاص ID جدید به آن‌ها و چاپ پیام:

new id ID assigned to IP:Port

که IP:Port مشخصات عمومی کارخواه فرستنده پیام است. در انتها بسته Response assigning ID ای شامل این ID تولید کرده و به کارخواه فرستنده، باز می‌گرداند.

نکته: ID های تولید شده به ترتیب از ۱ شروع می‌شوند و هر بار یک واحد افزایش می‌یابند. بنابراین شماره اولین درخواست کننده عدد ۱، دومین درخواست کننده عدد ۲ و الی آخر است.

- هنگامی که کارگزار بسته Request getting IP را از همسایه‌ها می‌گیرد، ابتدا پیام زیر را چاپ می‌کند.

ID<sub>A</sub> wants info of node ID<sub>B</sub>

که ID<sub>A</sub> و ID<sub>B</sub> به ترتیب ID طالب (کسی که بسته از آن آمده) و مطلوب (کسی که قرار است مبدا به آن متصل شود) است.

سپس باید یک بسته Response getting IP شامل اطلاعات مطلوب (B) تولید کند و به آدرس عمومی گره طالب (A) می‌فرستد. دقت کنید قسمت ID را نیز با مطلوب پر کنید.

- دریافت بسته Request updating info از کارخواهان و سپس بروزرسانی اطلاعات آنها ID جدید به آنها و چاپ پیام:

`id ID infos updated to IP:Port`

که `IP:Port` مشخصات عمومی کارخواه فرستنده پیام است.

- دریافت بسته Status، با دریافت این بسته، به ارسال کننده ی بسته بسته ی `Status Respond` را میفرستد. در صورتی که در بین راه کار خواه و کارگزار NAT قرار گرفته بود `FLAG` برابر با ۰، در غیر این صورت `FLAG` برابر با ۱ است.

#### ۴.۴ کارگزار NAT (امتیازی)

وظیفه کارگزار NAT در این شبکه به این شکل است که تمام واسطه‌های خود بجز واسطه ۰ را زیر شبکه‌ی خود می‌بیند و از طریق واسطه ۰ به دنیای بیرون متصل است. در نتیجه آدرس و درگاه بسته‌های ورودی و خروجی به زیر شبکه‌هایش را عوض می‌کند. وظایفی که برعهده دارد به شرح زیر است:

- تغییر مبدا بسته‌هایی که از زیر شبکه وارد می‌شوند با الگوریتم زیر:
- اگر تا بحال بسته‌ای با این آدرس و درگاه مبدا وارد نشده بود، یک آدرس و درگاه عمومی جدید به این اختصاص بده و این مقادیر را جایگزین کن (ترجمه کن).
- در صورتی که از قبل این آدرس و درگاه ترجمه شده بودند، این بار نیز از همان ترجمه قبلی استفاده کن.
- تغییر مقصد بسته‌هایی که از بیرون وارد می‌شوند:

اگر ترجمه‌ای از قبل برای مقصد وارد شده وجود نداشت بسته را دور می‌ریزد.

اگر ترجمه‌ای برای مقصد وجود داشت، تنها در صورتی ترجمه و مسیریابی می‌شود که از قبل بسته‌ای در جهت عکس برای مبدأ بسته فرستاده شده باشد. به بیان دیگر گره‌های بیرونی نمی‌توانند هیچ بسته‌ای به داخل زیر شبکه NAT بفرستند مگر اینکه از قبل بسته‌ای برای آنها فرستاده شده باشد (قسمت اصلی این مسئله، دور زدن این بخش است با ایجاد ترجمه‌ای در NAT.)

الگوریتم اختصاص آدرس و درگاه عمومی به این صورت است: با شروع از درگاه ۲۰۰۰ و اولین آدرس بعد از آدرس واسطه شماره ۰، هر بار درگاه را ۱۰۰ واحد اضافه می‌کنید، تا به عدد ۲۳۰۰ برسید، در این صورت باید آدرستان را یک واحد افزایش دهید و درگاه را دوباره از ۲۰۰۰ شروع کنید. برای واضح تر شدن قضیه به مثال زیر توجه کنید:

فرض کنید آدرس واسطه ۰ شما برابر با `1.2.3.4` است. بنابراین آدرس‌های عمومی‌ای که تولید می‌کنید به ترتیب برابر با:

1.2.3.5:2000

1.2.3.5:2100

1.2.3.5:2200

1.2.3.6:2000

1.2.3.6:2100

... می باشد. نکته: دقت کنید که برای راحتی کار، بسته‌هایی که از زیرشبکه ارسال می‌شوند، ابتدا آدرس مبدأ آن‌ها عوض می‌شود، سپس مسیریابی می‌شوند و بسته فرستاده می‌شود. ممکن است آدرس مقصد نیز نیاز به ترجمه داشته باشد، اما این ترجمه صورت نمی‌گیرد و پس از ارسال بسته دوباره از شبکه بسته به NAT برمیگردد و این بار چون بسته از بیرون آمده ترجمه صورت می‌گیرد و به زیرشبکه باز می‌گردد.

- دریافت دستور

```
block port range  $Port_{min}$   $Port_{max}$ 
```

که تنظیمات NAT را انجام می‌دهد. این تنظیمات تعیین می‌کند که محدوده درگاه مبدأ بسته‌هایی که از زیر شبکه این NAT می‌آیند، در چه بازه‌ای باشد. با هر بار دریافت این دستور، شما باید بازه‌ای شامل خود اعداد گفته شده را مسدود کنید و بسته‌های این بازه را دور بریزید. سپس بسته‌ی Drop assigning ID ای تولید کرده و در جواب برای فرستنده ارسال کنید. نکته: ممکن است چندین بار این دستور با محدوده‌های مشترک و غیر مشترک وارد شود، شما باید اجتماع تمام این بازه‌ها را مسدود کنید.

- دریافت دستور

```
reset network settings
```

با دریافت این دستور، NAT تمامی تنظیمات را به حالت اول برمیگرداند و تمامی Session های ایجاد شده را میبندد. به بیانی دیگر NAT بطور کامل RESET میشود و به حالت اولیه در زمان شروع اجرا باز میگردد. سپس با چاپ عبارت زیر:

```
please enter the base start number for port.
```

عدد شروع برای تخصیص درگاه برای گره‌های شبکه داخلی انتخاب میشود. در ادامه NAT باید برای تمامی گره‌های داخل شبکه که تا آن لحظه در NAT صاحب Session هستند، بسته‌ی NAT Updated ارسال شود.

- ارسال بسته Drop در صورتی که بسته‌ای از زیر شبکه خود دریافت کردید، اما بسته در محدوده block قرار داشت. در این حالت باید درگاه مبدأ را 1234 بگذارید و آدرس مبدأ بسته را، آدرس واسط شماره ۰ خود قرار دهید. آدرس و درگاه مقصد را نیز برابر آدرس و درگاه مبدأ بسته اصلی قرار دهید.
- ارسال بسته Drop در صورتیکه بسته‌ی دریافت شده از بیرون شبکه متعلق به هیچکدام از Session های ایجاد شده نباشد، در این حالت باید درگاه مبدأ را 4321 بگذارید و آدرس مبدأ بسته را، آدرس واسط شماره ۰ خود قرار دهید. آدرس و درگاه مقصد را نیز برابر آدرس و درگاه مبدأ بسته اصلی قرار دهید.

## ۵. موارد خاص

در این قسمت حالت‌های خاصی از مسئله که ممکن است پیش بیاید و در حالت‌های مسئله گفته نشده بود را آورده‌ایم تا راحت‌تر بررسی کنید. نکته بسیار مهم این است که کد شما به هیچ وجه نباید در طول تست از کار بیفتد، زیرا ممکن است نمره برخی قسمت‌ها را به کلی نگیرید. حالت‌های خاص بیشتری برای مسئله وجود دارد مانند اینکه یک گره به خودش پیامی بفرستد یا گره شما بسته‌هایی دریافت کند که از نظر منطقی غلط هستند ولی از نظر پروتکل تعریف شده درست هستند. در زیر حالت‌های مهم که در تست‌ها باید رعایت شوند آمده است و کفایت همین حالت‌ها را بررسی کنید:

- دور ریختن و چاپ عبارت

`invalid packet, dropped`

برای بسته‌هایی که:

`Checksum` لایه `IP` آن‌ها غلط است.

عدد پروتکل لایه `IP` و یا عدد پروتکل `UDP` که در لایه‌های زیرین خود نوشته می‌شوند، غلط باشد.

- برای گره‌های کارخواه و NAT در صورتی که دستور وارد شده غلط باشد، باید عبارت:

`invalid command`

را چاپ کنید. و منتظر دستورات بعدی باشید.

- در صورتی که درخواست اطلاعات ID گره‌ای از کارگزار شد که وجود نداشت یا خودش از قبل id ای نگرفته بود. کارگزار باید بسته را دور بریزد و عبارت:

`id not exist, dropped`

را چاپ کند.

- در صورتی که گره A از شبکه خارجی NAT بسته‌ای به گره B در زیر شبکه NAT بفرستد، در صورتی که B قبلاً بسته‌ای برای A نفرستاده باشد، باید عبارت:

```
outer packet dropped
```

را چاپ کنید و بسته را دور بریزید.

- در صورتی که کارخواهی از کارگزار id گرفته باشد، دیگر نمیتواند اینکار را بکند و در صورتی که دستور مربوطه وارد شد، عبارت:

```
you already have an id, ignored
```

را چاپ کنید.

- در صورتی که کارخواهی دستور ارسال پیام به id ای کرد که به آن id متصل نبودید، باید عبارت

```
please make a session first
```

را چاپ کنید.

## ۶. پیاده سازی

برای پیاده سازی این تمرین، شما حق استفاده از دو زبان ++c و java را دارید. پیشنهاد ما استفاده از زبان java است، چرا که مشکلات کار با اشاره‌گرها را نخواهید داشت. در تمام این تمرین، برای شبیه‌سازی شبکه و ارسال پیام بین گره‌ها، شما نیاز به استفاده از سیستم پرتو دارید.

### ۱.۶. مشترک

- برای کار با سیستم پرتو، نام کاربری و رمز خود را در پرونده `info.sh` قرار دهید.
- برای کامپایل شدن کد خود، از دستور `make` استفاده کنید. دقت کنید که کد ارسالی شما باید از این طریق کامپایل شود وگرنه شما نمره‌ای نخواهید گرفت.
- پس از کامپایل، ابتدا به اینترنت متصل شوید. سپس جهت اجرا شدن کد، باید فایل `free.sh` را اجرا کنید تا اطلاعات نقشه قبلی از پرتو شما حذف شود. سپس، با اجرای `new.sh` یک نقشه جدید ایجاد کنید. پس از این می‌توانید کد کامپایل شده خود را با اجرای `run.sh X` اجرا کنید. که  $X$  شماره گره‌ای از شبکه است که کد شما قرار است جای آن بنشیند.



## ۲.۶ . برنامه نویسی java

- در صورتی که زبان java را برای پیاده سازی انتخاب کردید، پیشنهاد ما استفاده از Eclipse IDE است، تا کارتان راحت تر شود. شما تنها حق تغییر فایل های پکیج `ir.sharif.ce.partov.machine` را دارید و فایل های دیگر خود را نیز تنها در این بخش قرار دهید.
- سه فایل `ServerMachine.java` ، `NATMachine.java` و `ClientMachine.java` به صورت پیش فرض پر شده اند. شما باید این سه فایل را برای هر یک از حالت هایی که گره شما در نقش کارگزار، NAT، کارخواه باشد، پر کنید و منطق خود را پیاده سازی کنید.

## ۳.۶ . برنامه نویسی C++

در صورتی که زبان c++ را انتخاب کردید، پیشنهاد ما استفاده از یکی از IDE های رایج مانند ( `intelij`، `eclipse`، `codeblocks` و غیره) است، چرا که ممکن است نیاز به استفاده از کتابخانه هایی داشته باشید که تا به حال به آن ها برنخورده اید. با امکانات این نرم افزارها می توانید کار خود را راحت تر انجام دهید.

شما باید کد اصلی خود را در این سه فایل `server_machine.cpp` ، `nat_machine.cpp` و `client_machine.cpp` قرار دهید تا هرگاه کد شما به عنوان یکی از این اعضاء اجرا شد، منطق گفته شده کار کند.

در صورتیکه می خواهید چند فایل دیگر نیز اضافه کنید، آن ها را در پوشه `user` قرار دهید و مطمئن شوید که کد شما با روش گفته شده کامپایل می شود.

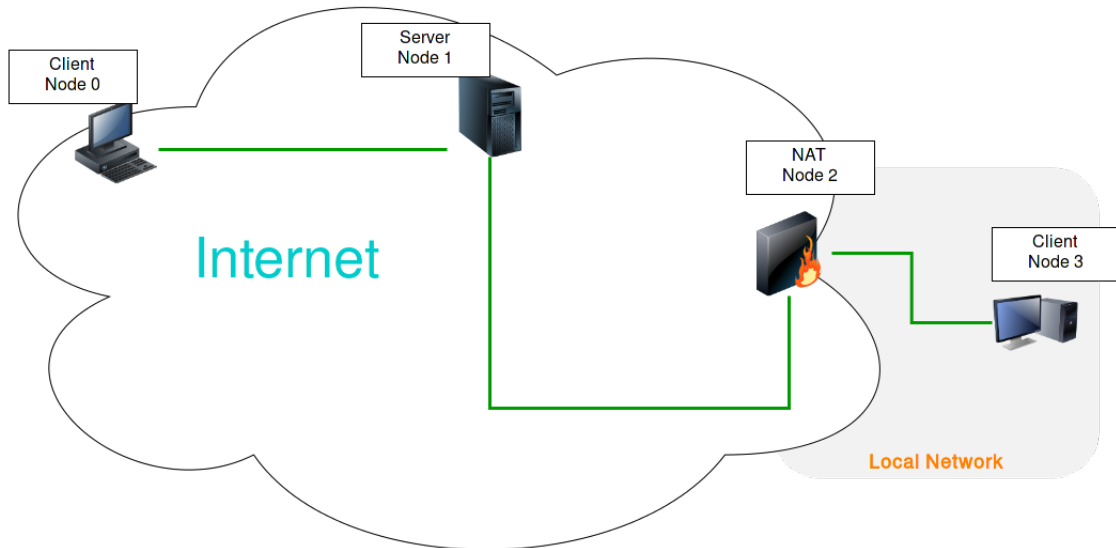
## ۴.۶ . کد nat

همانطور که گفته شد، فایل اجرایی NAT به صورت `jar`. در اختیار شما قرار داده شده است. شما می توانید پس از درست کردن نقشه روی پرتو، این کد را روی گره هایی که NAT هستند، اجرا کنید. برای اینکار کافیسیت فایل `run.sh X` را اجرا کنید که `X` شماره گره مورد نظر است. دقت کنید که برای اجرا این کد در رایانیتان نیاز به استفاده از جاوا نسخه ۸ را دارید، همچنین اگر این فایل را روی گره ای بجز NAT اجرا کنید، با پیغام خطا مواجه خواهید شد.

برای بخش امتیازی، در صورتی که می خواهید کد NAT را بزنید، مطابق گفته شده، فایل های مربوط به `nat machine` را پر کنید. تا در تست ها نمره امتیازی این بخش را بگیرید. در غیر اینصورت، نیازی به پر کردن این فایل در c++ یا java ندارید.

## ۵.۶. نقشه نمونه

برای راحتی کار شما، نقشه ساده‌ای جهت تست برنامه‌ی‌تان وجود دارد با نام `Hole_Punching_Simple` که به شکل زیر است. دقت کنید که نقشه مورد آزمون در داوری نمرات با این نقشه فرق می‌کند.



شکل ۲: Hole Punching Simple

## تحويل دادنی‌ها

- شما باید در موعد مقرر، مستند طراحی را تغییر دهید و در مخزن خود قرار دهید. نام این پرونده `desgin_documnet_pa2.md` می‌باشد و شما نباید نام این پرونده را تغییر دهید، همچنین توجه داشته باشید که عناوین موجود در این مستند را حذف نکنید و فقط پاسخ‌های خود را اضافه نمایید.
- سامانه‌ی داوری برخط برای هر تمرین پس از فرا رسیدن موعد تحويل مستند طراحی، اجرا می‌شود. این سامانه به صورت تصادفی حداکثر هر دو ساعت، به تمارین شما نمره می‌دهد. بنابراین نیاز است همواره یک نسخه‌ی به‌روز از کد خود را به سامانه‌ی گیت ارسال نمایید. به کمک دستور `make archive` فایل زیپی شامل تمام فایل‌هایی که برای اجرا شدن کد شما نیاز است بسازید. (این دستور فایل `info.sh` شما را درون زیپ قرار نمی‌دهد زیرا نیازی به این فایل نیست!) در صورتی که از کلاس‌ها و فایل‌های اضافه شده خودتان استفاده می‌کنید، سعی کنید در پوشه گفته شده باشد. در هر صورت فایل آرشیو شما باید قابلیت کامپایل/اجرا شدن را به روش سیستمی داشته باشد، در غیر اینصورت نمره شما صفر خواهد شد.
- به ازای هر تمرین در مخزن شخصی خود یک پوشه با حروف بزرگ و با شماره‌ی تمرین بسازید، همه‌ی پرونده‌های لازم را با همان نامی که در مستند تمرین ذکر شده‌است جهت نمره‌دهی با دستورهای زیر ارسال کنید:

```
cd ce443-962-stdid/PA2
git status
git add PA2.Zip design-doc.md
git commit -m "Finished PA2"
git push origin master
```

- در صورتی که پس از تحويل کد، تغییری در مستند طراحی به وجود آمده بود، در موعد مقرر این تغییرات را در یک پرونده‌ی `report.md` بنویسید و آن را ارسال نمایید.
- در نهایت مخزن شما باید ساختار زیر را داشته باشد:

```
—README.md
—PA2/
  —report.md
  —desgin_documnet_pa2.md
  —PA2.zip
```

## نکات ضروری

- به علت اینکه نمره‌ی تمرین به صورت خودکار داده می‌شود، ساختار پیام‌های گفته شده باید دقیقاً به صورت گفته شده باشد.
- نقشه‌ای که برای ارزیابی استفاده می‌شود با نقشه تست که در اختیار شما قرار گرفته فرق می‌کند.
- داوری خودکار بصورت برخط پس از اتمام مهلت ارسال "مستند طراحی" فعال می‌شود.
- به دلیل مشکلات اینترنتی بهتر است داوری را هنگامی که به اینترنت دانشگاه متصل هستید انجام دهید.
- در صورتی که هر مشکل یا پرسشی داشتید که فکر می‌کنید پاسخ آن برای همه مفید خواهد بود، آن را به گروه اینترنتی درس ارسال کنید.
- از فرستادن جواب تمرین به گروه اینترنتی درس خودداری کنید.
- تمام برنامه‌ی شما باید توسط خود شما نوشته شده باشد. فرستادن کل یا قسمتی از برنامه‌تان برای افراد دیگر، یا استفاده از کل یا قسمتی از برنامه‌ی فرد دیگری، حتی با ذکر منبع، تقلب محسوب می‌شود.
- پس از اتمام کارتان لازم است با اجرای دستور `make archive` فایل زیبایی شامل تمام فایل‌هایی که برای اجرا شدن کد شما نیاز است بسازید. در صورتی که از کلاس‌ها و فایل‌های اضافه شده خودتان استفاده می‌کنید، سعی کنید در پوشه گفته شده باشد. در هر صورت فایل آرشیو شما باید قابلیت کامپایل/اجرا شدن را به روش سیستمی داشته باشد، در غیر اینصورت نمره شما صفر خواهد شد.
- در این تمرین Judge فقط ابزار کمکی است و هر گونه خرابی در این سیستم تاثیری بر روی زمان تحویل ندارد.
- نسخه نهایی تمرین را در مخزن خود در [وب سایت طرشت](#) بارگذاری نمایید