به نام خداوند بخشنده مهربان

دانشگاه صنعتی شریف
مدرس: مهدی خرازی

تمرین برنامه نویسی دوم *
شبکه‌های رایانه‌ای

دانشکده‌ی مهندسی کامپیوتر
بهار ۱۳۹۸

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

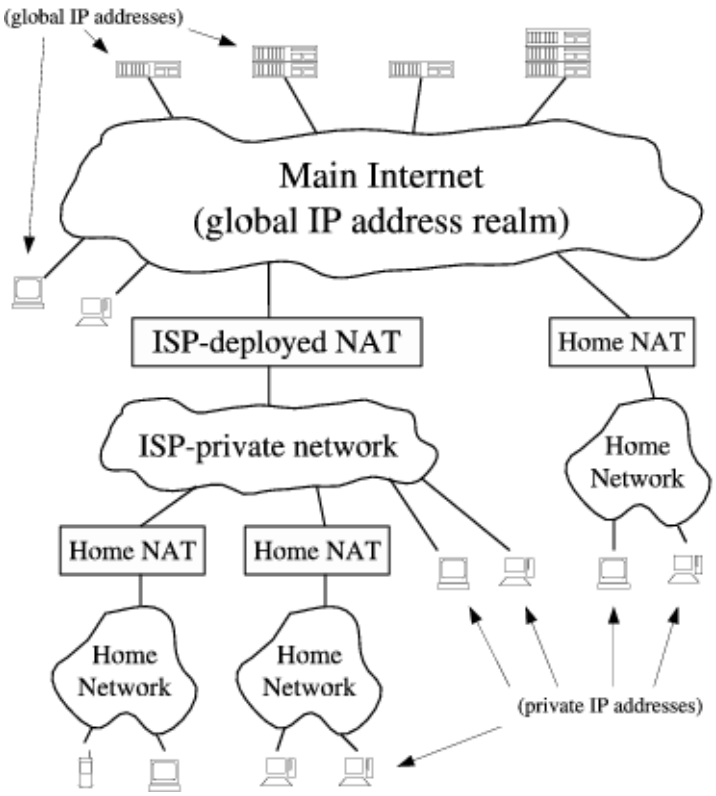
همانطور که می‌دانید اعضای شبکه مستقیما با هم در ارتباط نیستند. از این رو هر عضو شبکه برای برقراری ارتباط با عضوی دیگر مجبور است بسته‌اش را از مسیری بگذراند که شامل اعضای میانی است. کار برخی از این اعضای میانی، صرفا مسیریابی است و بسته را بدون تغییر به گره‌ی بعدی انتقال می‌دهند. اما اعضایی هم وجود دارند که بسته را بازرسی و فیلتر کرده و در برخی مواقع آن را تغییر می‌دهند. به طور کلی به این اعضا Middle Box می‌گوییم.

حالت خاصی از Middle Boxها، 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 شبکه 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 [۱] بخواهند یک ارتباط همتا به همتا [۲] ایجاد کنند و به یکدیگر بسته بفرستند، چگونه آدرس همدیگر را پیدا کنند و به نوعی محدودیت‌های NAT را دور بزنند. روش‌های مختلفی برای اینکار وجود دارد و به طور کلی به این مساله NAT Traversal می‌گویند.

ساختار شبکه را در حالت کلی می‌توان به شکل زیر در نظر گرفت:

* با سپاس از تیم دستیاران آموزشی

[1] Peer
[2] P2P



شکل ۱: ساختار کلی شبکه ـ منبع شکل

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 داده و یا ترجمه کند و سپس بسته را به گره بعدی بدهد. NATها با توجه به نوع عملکردشان به چند دسته تقسیم می‌شوند که رایج‌ترین آن‌ها Outbound NAT است. این نوع از NATها در حالت عادی برای بسته‌های وارد شده آدرس محلی را به آدرس عمومی ترجمه می‌کنند. اما آدرس بسته‌هایی که از خارج وارد می‌شوند را تنها در صورتی به آدرس محلی تبدیل می‌کنند که از قبل نشستی [۳] 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 این امکان را می‌دهند تا همزمان چند همتا از یک آدرس عمومی روی درگاه‌های مختلف استفاده کنند .

متاسفانه اکثر حالت‌های NAT باعث می‌شود اگر هر دو همتا پشت NATهای جدا باشند، هیچ کدام نتوانند با دیگری نشستی ایجاد کنند، زیرا NAT همتای مقابل بسته وارد شده از خارج را دور می‌ریزد و در نتیجه تنها حالت

[۳] یک نشست یا Session در پروتکل UDP و TCP از یک چهارتایی (Local IP, Local Port, Remote IP, Remote Port) تشکیل می‌شود.

ممکن برای برقراری ارتباط این است که یک همتا شروع کننده نشست باشد و همتای دیگر پشت NAT نباشد. برای برقراری ارتباط بین دو همتای پشت NAT باید از روش‌هایی مانند 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 لازم است هر دو همتا با کارگزار یک نشست درست کنند. در این زمان، 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 همتا را برای دیگری 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 NATاند و کارگزاری مانند S وجود دارد که پشت NAT نیست و هر دو A و B از قبل با S یک نشست UDP ایجاد کرده‌اند. 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.

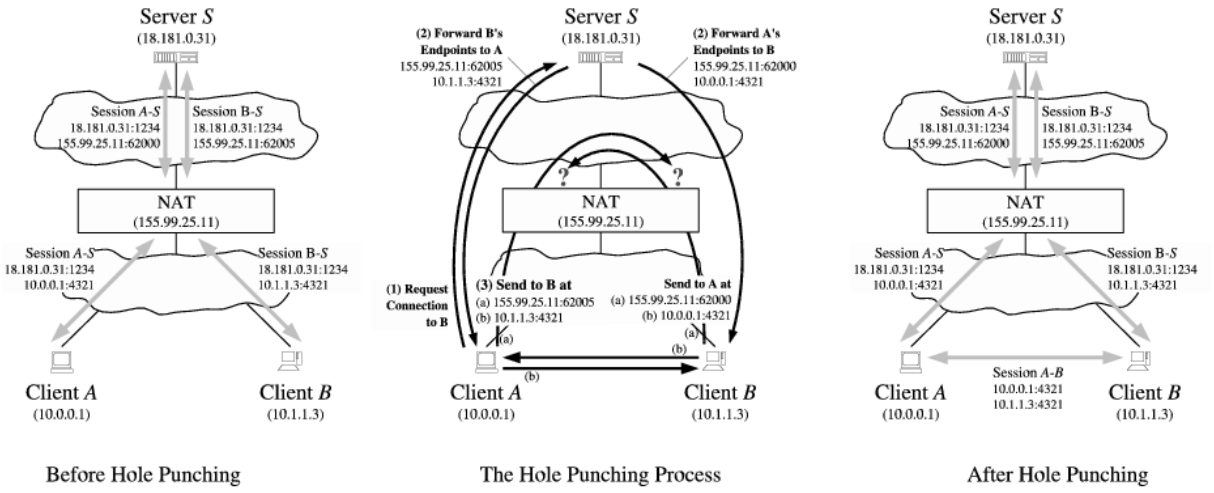
۲. S بسته ای به A شامل آدرس و درگاه محلی همچنین آدرس و درگاه عمومی B می‌فرستد. به طور همزمان اطلاعات A را نیز به B می‌فرستد تا هر دو همدیگر را بشناسند.

۳. هنگامی که A اطلاعات را از S دریافت کرد، به هردو 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

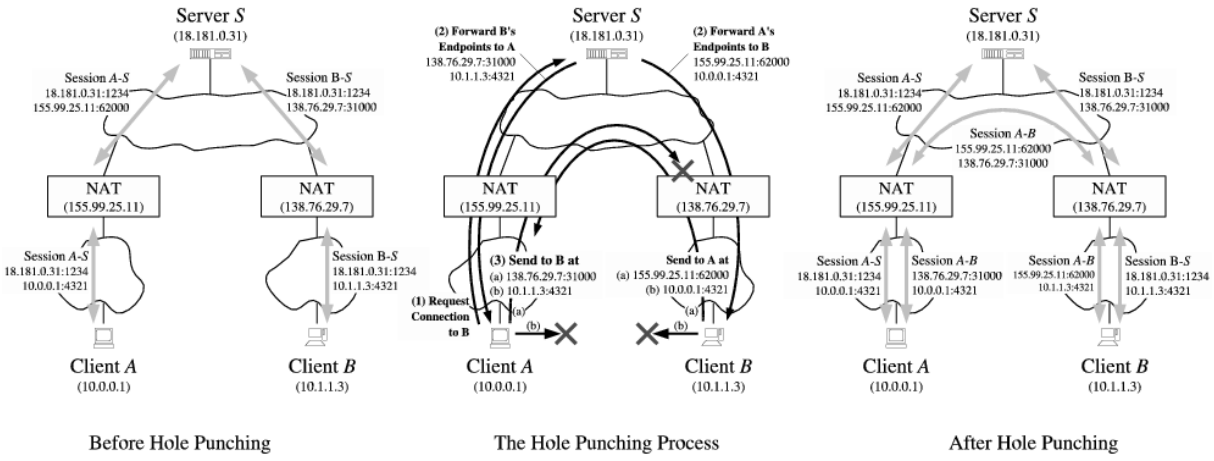
در این حالت هر دو همتا پشت یک NAT مشترک هستند. همانطور که در شکل می‌بینید، دو همتا پس از دریافت اطلاعات از S، از طریق آدرس عمومی و محلی یکدیگر تلاش به برقراری ارتباط می‌کنند. در اینجا چون هر دو در یک شبکه محلی هستند، مسیر محلی زودتر شکل می‌گیرد و در نتیجه نشست از طریق ارتباط محلی برقرار می‌شود.



شکل ۲: برقراری ارتباط در حالت NAT مشترک ـ منبع شکل

۲.۳. NAT متفاوت

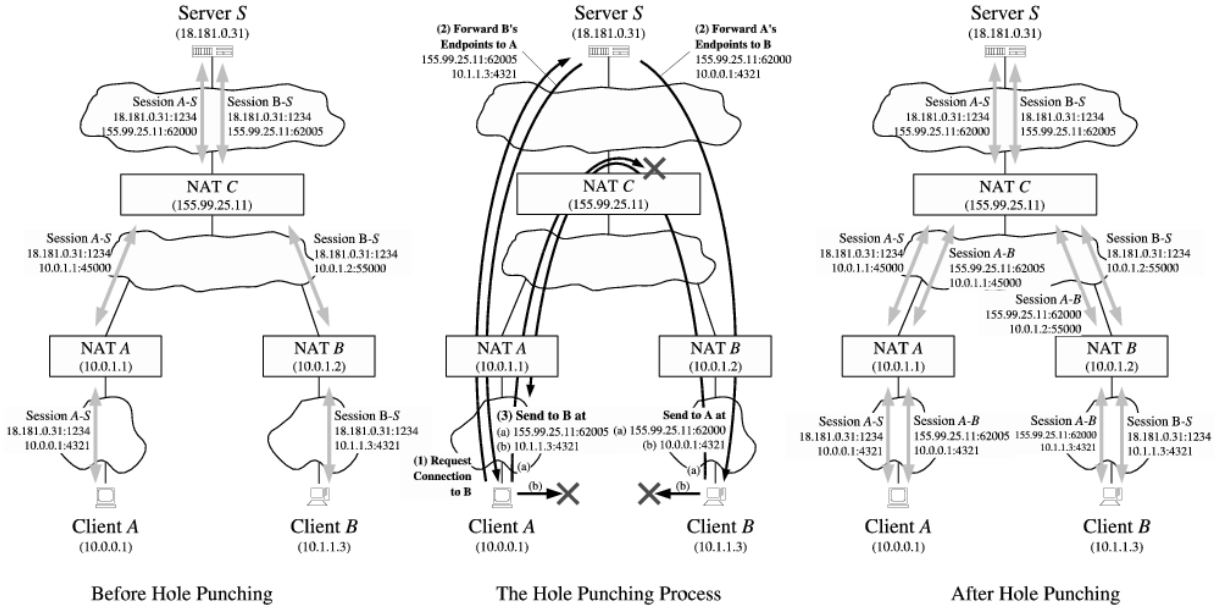
در این حالت دو همتا پشت NATهای متفاوت هستند و نمی‌توانند از طریق اتصالات محلی با هم ارتباط برقرار کنند. اما هنگامی که A درخواست خود را به آدرس و درگاه عمومی B ارسال می‌کند، یک خانه جدید در NAT مربوط به A ایجاد می‌شود که شامل آدرس و درگاه محلی A و آدرس و درگاه عمومی B است. اگر B از قبل مشابه همین درخواست را برای آدرس و درگاه عمومی A ارسال کرده باشد، جدول NAT مربوط به B نیز یک خانه شامل اطلاعات عمومی A و اطلاعات محلی B دارد و بسته به مقصد می‌رسد. در غیر این صورت این بسته دور ریخته می‌شود ولی هنگامی که B بسته‌اش را ارسال می‌کند، از هر دو NAT عبور می‌کند. مشاهده می‌شود که در هر صورت ارتباط دو طرفه ممکن خواهد بود.



شکل ۳: برقراری ارتباط در حالت NAT متفاوت ـ منبع شکل

۳.۳. NAT چندلایه

در این حالت دو همتا پشت چند لایه NAT متفاوت هستند و در این حالت نیز ارتباط محلی پاسخگو نخواهد بود. در این حالت مسیر بسته‌ها تا بالاترین گره‌ای که شبکه های دو همتا را به هم متصل می‌کند ادامه می‌یابد. در این گره آدرس‌ها ترجمه می‌شوند و درنتیجه بسته‌ها به داخل شبکه بر می‌گردند و به مقصد می‌رسند. چرا که تمام NAT های مسیر توسط بسته‌ی ارسالی همتای دیگر باز شده اند.



شکل ۴: برقراری ارتباط در حالت NAT چند لایه ـ منبع شکل

۴.۳. توپولوژی شبکه

توپولوژی شبکه در 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، آن را 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.

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.

شبکه های محلی، می‌توانند IP های مشترکی داشته باشند، زیرا از هم جدا هستند. از این رو، در تست‌ها هیچ

نشست یکسانی تشکیل نمی‌شود؛ یعنی دو IP یکسان، حتما روی درگاه‌های متفاوتی گوش [۴] می‌کنند.

ساختار شبکه به گونه‌ایست که، حداکثر ۱۵ عضو در شبکه حضور دارند. در این میان، ممکن است یک کارخواه پشت چند لایه NAT باشد بنابراین ممکن است یک بسته برای رسیدن به مقصد چندین بار دست‌خوش 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 (سرویس‌دهنده NAT ، کارگزار [۵] ، کارخواه [۶]) دارد. شما در نقش کارخواه باید دو کارخواهی که پشت NAT هستند و از توپولوژی شبکه خبر ندارند را بهم وصل کنید. در صورتی که بسته‌ای به کارخواهی وارد شد، باید آن را از طریق IP مسیریابی کرده و به گره بعدی برسانید. همچنین در نقش کارگزار باید بسته‌های ورودی را دریافت و ذخیره کنید و سپس آن‌ها 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 (البته این کد فقط وظایف اولیه NAT را انجام میدهد تا با ساختار بسته ها آشنا شوید). اما برای قسمت امتیازی نیاز است این بخش را نیز خودتان پیاده سازی کنید.

۱.۴. انواع بسته‌ها

تمامی بسته‌هایی که در این تمرین تولید و بین گره‌ها جابه‌جا می‌شوند، ساختار زیر را دارند و شما موظفید تمام این قسمت‌ها را پر کنید:

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

جدول ۱ : ساختار بسته‌ها

۱.۱.۴. Ethernet:

آدرس مبدا را برابر با آدرس Interfaceای که بسته از آن ارسال می‌شود بگذارید. آدرس مقصد را Broadcast (`FF:FF:FF:FF:FF:FF`) و Type را برابر IP (`0x0800`) قرار دهید. [۷]

[4] Listen
[5] Server
[6] Client
[7] 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. [۸]
- دقت کنید که هر گره ممکن است چند `Interface` داشته باشد که هر کدام از آن‌ها آدرس مخصوص به خود را دارند. قسمت مربوط به `Source IP address` تمام بسته‌ها هنگام ساخت، باید برابر IP در `Interface` شماره ۰ این گره باشد.
- `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 این قسمت را 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 گره بعدی انتقال دهید.

نکته: کم کردن `TTL` باعث بی‌اعتباری `Checksum` می‌شود، پس باید این قسمت را نیز هربار بروز کنید.

۳.۱.۴. UDP Header:

- مطابق با درگاه‌های گفته شده و طول پیام ارسالی جزئیات این قسمت را پر کنید. [۹]
- `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.

۲.۴. کارخواه

هر کارخواه چند وظیفه بر عهده دارد.

- مسیریابی و رد و بدل کردن تمام بسته‌هایی که از گره‌های همسایه می‌گیرد. هر کارخواه باید روی تمام واسط‌ها [۱۱] گوش کند و بسته‌هایی که وارد می‌شوند را پردازش کند. در صورتی که آدرس مقصد بسته مربوط به این

[8] https://en.wikipedia.org/wiki/IPv4#Header
[9] https://en.wikipedia.org/wiki/User_Datagram_Protocol#Packet_structure
[۱۰] طول این پیام ممکن است تا ۵۰ بایت باشد
[11] Interfaces

Type	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 Response	Server	Client	111

جدول ۲: انواع بسته‌ها

کارخواه نباشد (یعنی برابر با آدرس واسط شماره ۰ این کارخواه نباشد)، باید بسته را مسیریابی کرده و آن را روی واسط مناسب ارسال نماید، یعنی واسطی که آدرس IP و Mask اش با آدرس مقصد این بسته مطابقت داشته باشد. به طور دقیق تر واسطی که:

Interface IP & Interface Mask = Destination IP & Interface Mask

باشد. در صورتی که واسط مناسبی برای این بسته یافت نشد، بسته باید از واسط شماره ۰ به بیرون فرستاده شود.
دقت کنید که در این تمرین، در هنگام forward کردن این بسته ها احتیاجی به تغییر دادن آدرس های لایه‌ی Ethernet نیست.

- 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.

Type	ID	Data			
	5 bit	4 Byte	2 Byte	4 Byte	2 Byte
Req. assigning ID	0	Local IP	Local Port	-	-
Resp. assigning ID	New ID	-	-	-	-
Drop	0	drop	-	-	-
Req. getting IP	Dest ID	-	-	-	-
Resp. getting IP	Dest ID	Local IP	Local Port	Public IP	Public Port
Req. local session	Src ID	ping	-	-	-
Resp. local session	Src ID	pong	-	-	-
Req. public session	Src ID	ping	-	-	-
Resp. public session	Src ID	pong	-	-	-
Message	Src ID	Message[10]			
NAT updated	0	-	-	-	-
Req. updating info	Curr. ID	Local IP	Local Port	-	-
Status	0	Local IP	Local Port	-	-
Status Response	FLAG	0	0	-	-

جدول ۳: جزئیات بسته‌ها

نکته: دقت کنید از این‌جا 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.

درون بسته، Local Port (یعنی `X`) و IP واسط شماره ۰ خودش را قرار می‌دهد. آن گاه بسته را روی واسط مناسب به کارگزار شبکه ارسال می‌کند. با این کار کارگزار یک `ID` به این گره اختصاص داده و آن را در قالب پیام Response Assigning ID برای این گره خواهد فرستاد.

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 خواهد شد.

- دریافت بسته Drop : بر مبنای درگاه مبدا در این بسته دو حالت ممکن است پیش بیاید:

— 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 چندین‌بار تکرار شود و شما مجبور باشید هربار 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 پکت ارسالی توسط NAT از ادامه مسیریابی باز مانده است. در صورتی که از قبل به این `ID` متصل بودید، پیام زیر را چاپ کنید.

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

دقت کنید که ID مربوط به آخرین همتایی است که به آن بسته‌ای ارسال شده است. در صورتی که اتصال کامل نبود لازم نیست پیغامی چاپ شود.

- دریافت بسته Response assigning ID شامل `ID` اختصاص داده شده به این گره توسط 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 $(1, 31)$ است. هنگامی که این عبارت وارد شد، باید بسته Request getting IP را با `ID` وارد شده تولید شود. پس از آن از طریق واسط مربوطه به تنها کارگزار شبکه ارسال شود. کارگزار در پاسخ، بسته‌ی Response getting IP را برای این کارخواه خواهد فرستاد.

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

کارخواه 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`

چاپ شده و بسته‌ای ارسال نخواهد شد.

- 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 و از قبل به این ID متصل نشده باشد. سپس باید یک بسته نوع ۳ مانند بسته دریافتی درست کند و در جواب بفرستد، با این تفاوت که این بار 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` دیگری دریافت نشده بود. پس همانطور که می‌بینید اتصال دو طرفه است و وقتی یک نشست برقرار می‌شود هر دو طرف بهم متصل می‌شوند.

- پردازش دستور

`send msg to` ID:msg

که ID شماره همتای مقصد و msg پیامیست که باید ارسال شود.

نکته: هر پیامی شامل حروف الفبا، اعداد و فاصله با طول کمتر از ۵۰ حرف باید قابل ارسال باشد.

- دریافت بسته Message : باید پیام موجود در این بسته به شکل زیر نمایش داده شود:

`received msg from` ID:msg

که ID شماره همتای مبدا و msg پیامیست که دریافت شده است.

- دریافت بسته NAT updated : باید بسته‌ای از نوع Request updating info به همان نحوه‌ی پر شدن بسته Request assigning ID پر شود به سمت کارگزار فرستاده شود، تنها با این تفاوت که در قسمت ID باید ID فعلی این کارخواه نوشته شود.

- پردازش دستور

`status`

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

- دریافت بسته Status Response: این بسته از کارگزار در پاسخ به بسته‌ی `Status` دریافت می‌شود. این بسته، حاوی `FLAG` است که نشان دهندهٔ نوع ارتباط با کارگزار است، در صورتی که `FLAG` برابر با ۱ بود باید عبارت

`direct`

و در صورتی که ۰ بود باید عبارت

`indirect`

چاپ شود.

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 برابر با این آدرس و درگاه باشد. وظایف کارگزار عبارت اند از:

- مسیریابی بسته‌ها: بسته‌هایی که مقصدشان برابر با آدرس بالا نبود باید مسیریابی شده و به واسط مناسب ارسال گردند.

- 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 (IP باید با فرمت x.x.x.x نمایش داده شود). در انتها بسته Response assigning ID ای شامل ID تولید کرده و به کارخواه فرستنده، باز می‌گرداند.
نکته: ID های تولید شده به ترتیب از ۱ شروع می‌شوند و هربار 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 دریافت می‌کند، ابتدا پیام زیر را چاپ می‌کند:

ID_A `wants info of node` ID_B

که ID_A و ID_B به ترتیب ID طالب(کسی که بسته از آن آمده) و مطلوب(کسی که قرار است مبدا به آن متصل شود) است. سپس یک بسته Response getting IP شامل اطلاعات عمومی و محلی مطلوب(B) تولید می‌کند و به آدرس عمومی گره طالب(A) می‌فرستد. دقت کنید قسمت ID را نیز با ID_B پر کنید .

- دریافت بسته Request updating info از کارخواهان و سپس بروزرسانی اطلاعات 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 ارسال‌کننده، بسته‌ی `Status Respond` را می‌فرستد. در صورتی که در بین راهِ کارخواه و کارگزار NAT قرار گرفته بود `FLAG` برابر با ۰، و در غیر این صورت `FLAG` برابر با ۱ خواهد بود.

۴.۴. کارگزار NAT (امتیازی)

هر کارگزار NAT در این شبکه به صورتی قرار گرفته است که تمام واسط‌های خود بجز واسط ۰ را زیر شبکه‌ی خود می‌بیند و از طریق واسط 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 می‌دهد و این مقادیر را جایگزین می‌کند(ترجمه می‌کند).

در صورتی که از قبل این آدرس و درگاه ترجمه شده بودند، این بار نیز از همان ترجمه قبلی استفاده می‌کند.

- تغییر مقصد بسته‌هایی که از بیرون وارد می‌شوند:

  اگر ترجمه‌ای از قبل برای مقصد 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 (هدف اصلی این تمرین، دور زدن این بخش با ایجاد ترجمه‌ای در NAT است.)

  الگوریتم اختصاص آدرس و درگاه عمومی به این صورت است:

  اولین درگاه اختصاص یافته برابر با ۲۰۰۰ و اولین آدرس اختصاص یافته برابر با اولین آدرس بعد از آدرس واسط ۰ این NAT است. برای تولید آدرس‌های بعدی هر بار عدد درگاه ۱۰۰ واحد افزایش داده می‌شود و این کار ۲ بار انجام می‌شود. برای تولید آدرس عمومی بعدی، آدرس را یک واحد افزایش داده و درگاه دوباره روی ۲۰۰۰ تنظیم می‌شود. برای واضح‌تر شدن قضیه به مثال زیر توجه کنید:

  فرض کنید آدرس واسط ۰ این NAT برابر با `2.3.4.5` است. بنابراین آدرس‌های عمومی‌ای که تولید می‌شود به ترتیب برابر با:

  `2.3.4.6:2000`

  `2.3.4.6:2100`

  `2.3.4.6:2200`

  `2.3.4.7:2000`

  `2.3.4.7:2100`

  ...

  خواهد بود.

  نکته: دقت کنید که برای راحتی کار، بسته‌هایی که از زیرشبکه ارسال می‌شوند، ابتدا آدرس مبدأشان عوض شده، سپس مسیریابی شده و روی واسط مناسب 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 تعیین می‌کند درگاه مبدا بسته‌هایی که از زیرشبکه‌ی این NAT می‌آیند، در چه بازه‌ای می‌تواند باشد. با هربار دریافت این دستور، شما باید بازه‌ای شامل خود اعداد گفته شده را مسدود کنید و از این پس بسته‌های

این بازه را دور بریزید. (ابتدا و انتهای بازه‌های وارد شده نیز باید مسدود شوند.)
نکته: ممکن است چندین بار این دستور با محدوده های متفاوت وارد شود. در این صورت باید اجتماع تمام این بازه‌ها مسدود شوند.

- 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 باید برای تمامی گره های داخل شبکه که تا آن لحظه در NAT صاحب Session هستند، بسته ی NAT Updated ارسال شود. برای این بسته درگاه مبدا را برابر با ۱۲۳۴ و آدرس مبدا را برابر با آدرس واسط شماره ۰ این NAT قرار دهید.

- ارسال بسته Drop در صورتی که بسته‌ای از زیرشبکه دریافت شد، اما درگاه بسته در محدوده درگاه‌های مسدود شده قرار داشت. در این حالت باید درگاه مبدأ برابر با `1234` و آدرس مبدأ برابر با آدرس واسط شماره ۰ این NAT قرار داده شود. آدرس و درگاه مقصد را نیز برابر آدرس و درگاه مبدأ بسته اصلی قرار دهید.
- ارسال بسته Drop در هنگام دریافت بسته از شبکه بیرونی، در صورتیکه گیرنده‌ی بسته در زیرشبکه قبلا بسته‌ای برای فرستنده این بسته نفرستاده باشد. در این حالت باید درگاه مبدأ برابر با `4321` و آدرس مبدأ برابر با آدرس واسط شماره ۰ این NAT قرار داده شود. آدرس و درگاه مقصد را نیز برابر آدرس و درگاه مبدأ بسته اصلی قرار دهید. همچنین باید عبارت:

```
outer packet dropped
```

در خروجی چاپ شود.

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 بیافتد، 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 کافیست 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 خود درخواست‌کننده از قبل ID ای نگرفته بود، کارگزار باید بسته را دور بریزد و عبارت:

```
id not exist, dropped
```

را چاپ کند.

- 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 نیست، 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 فایل‌های دیگر خود را نیز تنها در این بخش قرار دهید.
- سه فایل `ServerMachine.java` ، `NATMachine.java` و `ClientMachine.java` به صورت پیش فرض پر شده‌اند. شما باید این سه فایل را برای هر یک از حالت‌هایی که گره شما در نقش کارگزار، NAT ، و یا 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 (Clion، eclipse، codeblocks و غیره) است، چرا که ممکن است نیاز به استفاده از کتابخانه‌هایی داشته باشید که تا به حال به آن‌ها برنخورده‌اید. با امکانات این نرم‌افزارها می‌توانید کار خود را راحتتر 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 می‌خواهید چند فایل دیگر نیز اضافه کنید، آن‌ها را در پوشه user قرار دهید و مطمن شوید که کد شما با روش گفته شده کامپایل می‌شود.

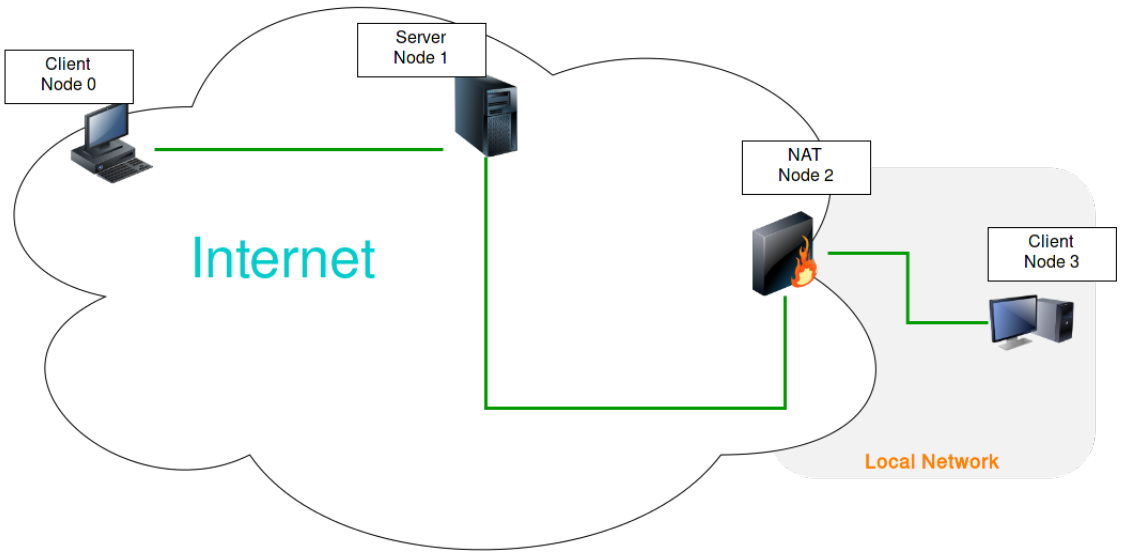
۴.۶ کد nat

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

برای بخش امتیازی، در صورتی که می‌خواهید کد NAT را بزنید، مطابق گفته شده، فایل های مربوط به nat machine را پر کنید. تا در تست‌ها 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 برنامه‌ها 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 ۵: Hole Punching Simple

تحویل دادنی‌ها

- سامانه‌ی داوری برخط برای هر تمرین یک هفته پس از شروع تمرین، 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.
- شما باید گزارشی با فرمت md از روند طراحی و پیاده‌سازی تمرین به همراه کد نهایی خود ارسال کنید.
- 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 ce443-972-[stdid]/PA2
git status
git add PA2.zip report.md
git commit -m "Finished PA2"
git push origin master
```

- در نهایت مخزن شما باید ساختار زیر را داشته باشد:

```
--README.md
--PA2/
        --report.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 یک هفته فعال می‌شود.
- به دلیل مشکلات اینترنتی بهتر است داوری را هنگامی که به اینترنت دانشگاه متصل هستید، انجام دهید.
- در صورتی‌که 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 زیپی 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 .
- نسخهٔ نهایی تمرین را در مخزنِ خود در وب سایت طرشت بارگذاری نمایید