به نام خدا

درس: شبکه مخابرات داده‌ها

استاد: دکتر محمدرضا پاکروان

گزارش پروژه پایانی درس

Simple Chat Application

سیّدمحمّدامین منصوری طهرانی

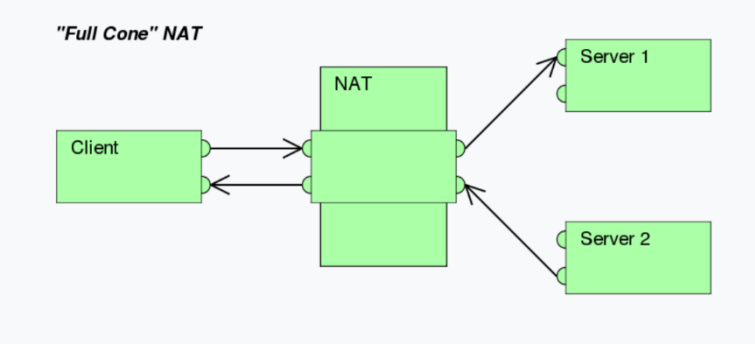
۹۴۱۰۵۱۷۴

NAT Traversal[1]

در تمام NAT های زیر، همان‌طور که می‌دانیم پس از map شدن socket ای که پشت NAT است، تمام بسته‌های فرستاده شده از یک internal socket با IP Address و Port No. جدیدی که NAT به آن‌ها اختصاص داده توسط شبکه شناخته می‌شوند. تفاوت آن‌ها در نحوه ارتباط host های خارجی با هر یک از آن‌هاست.

هم‌چنین در مدل‌های نامتقارن(۳ مورد اول) در ترجمه‌ای که توسط NAT صورت می‌گیرد، source port no. حفظ می‌شود. اما در مورد آخر یا symmetric NAT در ازای هر کانکشنی که برقرار می‌شود شماره پورت مبدأ به عددی رندم نگاشته می‌شود.

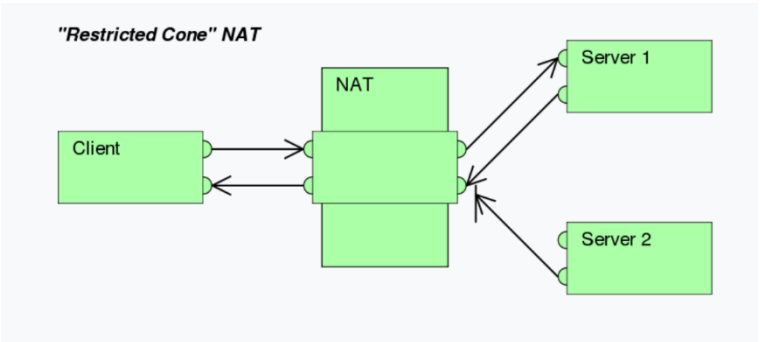
۱. در مدل Full Cone، تمامی hostهای خارج NAT می‌توانند با ارسال بسته‌های خود به socket address مربوط به NAT با internal socket ارتباط برقرار کنند. بدیهی است برای ارتباط با iAdd,iPort باید بسته‌ها را به eAdd,ePort ای که NAT ترجمه کرده بفرستند. در واقع تنها لزومی که برای برقراری ارتباط لازم است این است که پورت‌های internal host و NAT IP را بدانیم و بسته‌ها را به آن‌ها برسانیم تا به دست client در شکل فوق برسد.



شمایی که که نحوه تغییر عددها را نشان بدهد در زیر آورده شده‌است.

```
(LAN_IP, LAN_PORT) <= [(WAN_IP, LAN_PORT) <- (*, *)]
```

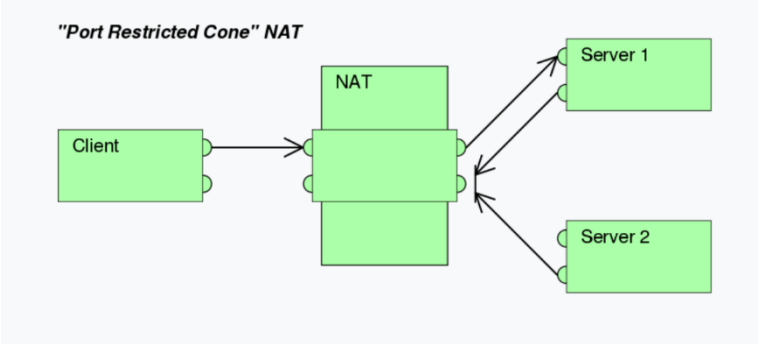
۲. در مدل Restricted Cone، یک external host فقط در صورتی می‌تواند از طریق eAdd,ePort(ترجمه شده iAdd,iPort) با سورس پشت NAT ارتباط برقرار کند که قبلاً بسته‌ای از این سورس به یکی از پورت‌های این external host address (شماره پورت آن مهم نیست) فرستاده شده باشد. سرورهای دیگر که بسته‌ای از این client و NAT نگرفته‌اند، حتی اگر



NAT IP و شماره پورت مربوط به client را بدانند و به آن بسته ارسال کنند، بسته آن‌ها توسط NAT، drop می‌شود.

```
1. (LAN_IP, LAN_PORT) => [(WAN_IP, LAN_PORT) -> (REM_IP, REM_PORT)]
2. (LAN_IP, LAN_PORT) <= [(WAN_IP, LAN_PORT) <- (REM_IP, *)]
```

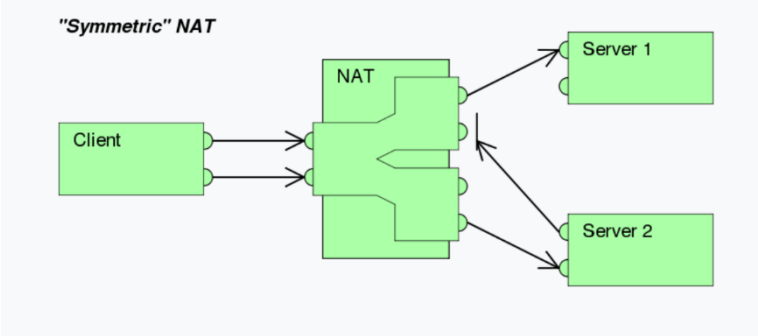
۳. در مدل Port Restricted Cone علاوه بر به ارث بردن ویژگی‌های Restricted Cone ویژگی اضافه‌تری هم دارد: یک external host فقط در صورتی می‌تواند از یک پورت خاص خودش به eAdd,ePort(ترجمه شده iAdd,iPort) بسته بفرستد و با سورس پشت NAT ارتباط برقرار کند که قبلاً بسته‌ای از این سورس دقیقاً به همان پورت خاص این external host address فرستاده شده باشد. در شکل زیر سرور ۲ چون هیچ بسته‌ای از NAT نگرفته‌است، نمی‌تواند با آن ارتباط برقرار کند.(ویژگی Restricted Cone) سرور ۱ نیز از پورت دوم خود نمی‌تواند به NAT بسته‌ای بفرستد چون قبلاً به این پورت از طرف NAT چیزی نرسیده است. جواب‌ها باید دقیقاً از همان پورتی که درخواست شده reply شوند. در غیر این‌صورت drop می‌شوند.(۳ عدد باید یکسان باشند:NAT IP, source port number, destination port number)




```
1. (LAN_IP, LAN_PORT) => [(WAN_IP, LAN_PORT) -> (REM_IP, REM_PORT)]
2. (LAN_IP, LAN_PORT) <= [(WAN_IP, LAN_PORT) <- (REM_IP, REM_PORT)]
```

۴. در مدل Symmetric همان‌طور که در بالا اشاره شد به ازای هر کانکشن از طرف مبدأ نگاشت به این صورت انجام می‌شود که internal IP به NAT public IP ترجمه شده و internal port به یک wan port رندم نگاشته می‌شود. جواب‌ها نیز باید دقیقاً به همین دو عدد فرستاده شود.(اگر سروری بسته‌ای از مبدأ دریافت نکرده باشد نیز طبیعتاً مثل قبل نمی‌تواند چیزی به این مبدأ بفرستد.)

```
1. (LAN_IP, LAN_PORT) => [(WAN_IP, WAN_PORT) -> (REM_IP, REM_PORT)]
2. (LAN_IP, LAN_PORT) <= [(WAN_IP, WAN_PORT) <- (REM_IP, REM_PORT)]
```



UDP Hole Punching

۱. اگر پشت یک NAT باشند شروع می‌کنند پیام دادن به private end همدیگر و چون پشت یک NAT و در یک private network هستند ارتباط برقرار می‌شود.

۲. هم به private end point و هم به public end point پیام ارسال می‌کنند و آن پیامی که اولی می‌دهند یا به مقصد نمی‌رسد یا به مقصد غلطی می‌رسد. پیامی که به دومی می‌رسد از NAT رد می‌شود و اگر فرض کنیم رفتار NAT مناسب باشد و تمام ترافیک ورودی از یک نود را به یک نود و یک IP Address بنگارد در این صورت وقتی یک طرف به NAT دیگری پیام دهد، NAT خود این نود متوجه می‌شود مسیر معتبر رو به بیرون است. وقتی طرف دیگر به public end point دیگری پیام می‌دهد نیز همین اتفاق برای NAT دیگر می‌افتد.

۳. باز به public end point و private end point پیام ارسال می‌کنند که طبیعتاً دومی بازهم نتیجه نمی‌دهد. اگر هر دو در پشت دو لایه NAT که لایه بالایی یکسان و لایه پایینی متفاوت است باشند، در صورتی که hair pin translation پشتیبانی شود می‌توانند ارتباط برقرار کنند. در واقع NAT لایه بالا باید بتواند بین دو NAT زیرین خود ارتباط برقرار کند.

STUN, ICE, TURN

۱ و ۲ و ۳:

یکی از موارد بسیار پرکاربرد برای VoIP پروتکل SIP یا Session Initiation Protocol می‌باشد که در عین حال با چالش‌های بزرگی از جمله NAT ها و firewall ها روبروست. VoIP یک مثال بارز از peer to peer connection است. اگر بخواهیم مسأله را به طور خلاصه توضیح دهیم، مشکل ارتباط peer to peer به خاطر NAT ها به وجود می‌آید چون دو طرفی که پشت NAT هستند، از این موضوع اطلاع ندارند و socket هایی که برای هم ارسال می‌کنند حاوی پورت و IP ای است که خودشان فکر می‌کنند نه آن چیزی که از طرف شبکه دیده می‌شود. بنابراین ارتباط دو طرف غیرممکن می‌شود چون نمی‌توانند بسته‌ها را به مسیر درست بفرستند. برای حل این مشکل راه‌حل‌های مختلفی ارائه شد:

ALG(Application Layer Gateway): در این حالت این پروتکل بسته‌ها را در حین عبور از NAT بررسی کرده و آن‌هایی که حاوی آدرس IP باشند را به آدرسی که به NAT مربوط است و می‌داند تغییر می‌دهد. بدیهی است که با این کار در واقع باید از امنیت به مقدار قابل ملاحظه‌ای صرفنظر کنیم(!) زیرا نمی‌توانیم بسته‌ها را رمز کنیم.(ALG باید بتواند به ازای هر بسته سریعاً این عملیات را انجام دهد.) پس این روش مناسب نیست. هم‌چنین یک عیب دیگر آن نیاز به اطلاع کامل از SIP است و چون بخشی از router است، فلسفه IP که جدایی اپلیکیشن از شبکه بود را زیر سوال می‌برد. پس قابلیت extension برای SIP نمی‌گذارد.

انجمن IETF در اولین روش simple traversal of UDP through NAT را پیشنهاد داد.(STUN) در این روش هر peer با ارتباط برقرار کردن با STUN server آدرس و پورتی که شبکه از او می‌بیند را به دست می‌آورد و در تمام بسته‌هایی که به آدرس نیاز دارند (مثل کاربرد VoIP) این آدرس جدید را قرار می‌دهد و در این حالت امنیت آن هم به خطر نمی‌افتد. البته برای بعضی NAT ها مثل متقارن‌ها کار نمی‌کند زیرا آدرس هر بار تصادفاً به عددی جدید نگاشته می‌شود. برای رفع این مشکل traversal using relay NAT یا همان TURN پیشنهاد شد که از یک relay استفاده می‌شود و پیام‌ها برای آن از دو peer فرستاده می‌شوند. دیگر مستقیماً نیازی به آدرس دو طرف نیست. در این حالت از مزایای STUN برای بعضی کاربرها نمی‌توان استفاده کرد و ICE برای بهره‌گیری از همه مزایا پیشنهاد شد. در این روش تمامی راه‌های ارتباطی ممکن پیدا شده(هر دو با آدرسی که خودشان فکر می‌کنند درست است-هر دو با استفاده از Relay- هر دو با STUN server- یکی آدرس خودش و دیگری STUN server و ...) به هر کدام یک اولویت نسبت داده می‌شود و پس از محاسبه بهترین مسیر، نوع ارتباط بین این دو مشخص می‌شود و از مزایای STUN و TURN با هم استفاده می‌شود.[2]

References

[1] Wikipedia-NAT, Think Like a Computer, Think Like a Computer, Kurento

[2] IETF Journal ICE, IETF STUN, Wikipedia STUN, Wikipedia ICE, IETF TURN, IETF ICE, SIP Wikipedia