Data Networks, Final Project

Instructor: Dr. MohammadReza Pakravan

Simple Chat Application

Fall 2018

گزارش پروژه

Ali Fathi 94109205

تمرین اول:

نمونه یک مکالمه بین امیر و علی در زیر آمده است(P2PClient.py):

```
Login as? (Your Name): Amir
Your Port to Listen? : 2222
Your Command (Show/End/New): New
Enter Your Peer Name: Ali
Enter Your Peer IP: 127.0.0.1
Enter Your Peer Port: 1111
*** Chatting ***
(ESC to go out)You: Hi Ali!
(ESC to go out)You: How Are You?
Hi Amir!
Tnx.
And U?
(ESC to go out)You: Me Too
(ESC to go out)You: Bye
(ESC to go out)You: ESC
Your Command (Show/End/New): End
Have a Nice Day :))
```

```
Login as? (Your Name): Ali
Your Port to Listen? : 1111
You Have New Message From Amir
You Have New Message From Amir
Your Command (Show/End/New): Show
Whom to Show (Amir): Amir
*** Chatting ***
Hi Ali!
How Are You?
(ESC to go out)You: Hi Amir!
(ESC to go out)You: Tnx.
(ESC to go out)You: And U?
Me Too
Bye
Your Peer Left The Chat
Your Command (Show/End/New):End
Have a Nice Day :))
```

Describe the UDP hole punching mechanism in the following scenarios:

1. Peers are behind a common NAT.

در این حالت، آدرس‌های Public و Private ای که سرور (Rendezvous Server) از Peer های A و B می‌داند، با هم متفاوت‌اند زیرا هر دو پشت NAT هستند. پس از صحبت اولیه، هر دو از آدرس‌های Public و Private هم مطلع می‌شوند. سپس تلاش

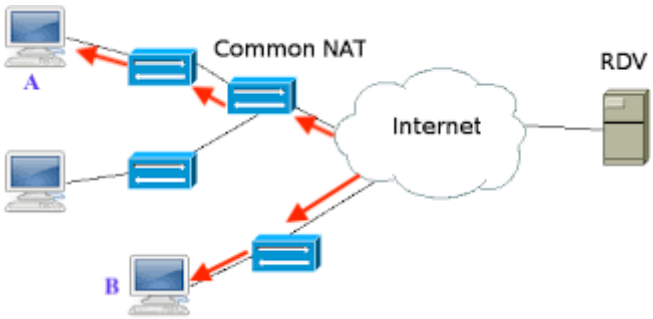
می‌کنند روی هر دو آدرس، به هم اطلاعات بفرستند اما چون پشت NAT یکسانی هستند، آدرس Private همدیگر را می‌بینند و پاسخ در مکالمه با آدرس خصوصی، بسیار سریع‌تر از مکالمه با آدرس عمومی دریافت می‌شود (چون لازم نیست داده ابتدا به NAT برود و سپس به طرف مقابل ارسال شود) پس هر دو، آدرس Private هم را برای ارتباط ذخیره می‌کنند و روی آن داده می‌فرستند.

2. Peers are behind different NATs.

مشابهاً، سرور آدرس‌های Public و Private متفاوتی از هر دو می‌بیند. اما پس از آنکه هر دو Peer آدرس‌های یکدیگر را دانستند و اقدام به پیدا کردن آدرس مناسب برای ارتباط گشتند، جوابی از آدرس‌های Private نمی‌گیرند زیرا پشت NAT های مختلفی هستند. در این حالت هر دو آدرس Public یکدیگر را ذخیره کرده و برای ارتباط از آن استفاده خواهند کرد.

3. Peers are behind multiple levels of NATs.

کلیت این حالت شبیه سوال قبل است؛ یعنی Peer ها برای ارتباط، آدرس Public هم را ذخیره می‌کنند که در واقع آدرس IP آن آدرس آخرین لایه NAT است. وقتی A ، به آدرس عمومی B پیام درخواست ارتباط می‌فرستد، این پیام به NAT آخرین لایه می‌رسد. از دید این NAT ، B یکی از NAT های متصل به آن است (لایه قبل) و این بسته را به آن تحویل می‌دهد و همین روند ادامه می‌یابد تا پیام به دست B واقعی برسد. در فرآیند برعکس هم همین اتفاق در سمت A می‌افتد و NAT ها پیام‌ها را روی Port های مربوط به A ، به سمت عقب پس می‌دهند.

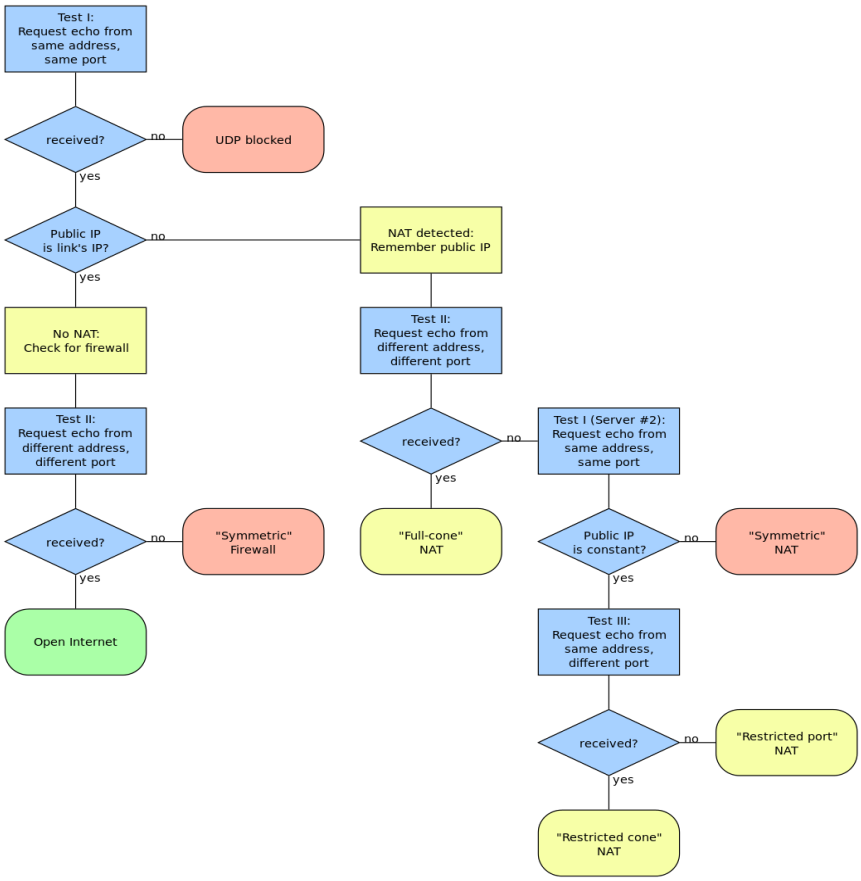


لازم به ذکر است اگر هر دو Peer در پشت NAT لایه بالا باشند (برای مثال در شکل بالا B کامپیوتر میانی باشد) همچنان آدرس Private یکدیگر را نمی‌بینند و در هنگام ارتباط با NAT لایه‌ی بالای خود صحبت می‌کنند (یعنی پیام، تا بالاترین NAT که به سرور Rendezvous متصل است بالا می‌رود حتی اگر در لایه‌های قبل تری هر دو Peer پشت یک NAT بوده اند).

Briefly describe the role of the following protocols in initiating a peer to peer communication,

1. RFC 5389: Session Traversal Utilities for NAT (STUN),

به طور کلی، در حضور NAT ها، تنها ارتباط Peer to Server قابل انجام است (مانند خواندن وبسایت) و ارتباطاتPeer to Peer مانند برقرار ارتباط ویدئویی دو نفر ممکن نیست، زیرا در حالت اول NAT مبدا می‌فهمد جواب سرور را به کدام سمت بفرستد؛ اما در حالت دوم NAT گیرنده چون هنوز Session ای باز نکرده، هنوز مشخص نیست چه آدرس عمومی‌ای دارد و در واقع نمی‌دانیم Peer مقصد پشت چه NAT ای قرار دارد. در روش STUN ، آدرس عمومی NAT ها را شناسایی می‌کنیم و در ۹۵ درصد حالات با موفقیت همراه است. مراحل کلی آن به صورت زیر است:



که دسترسی Client به Peer خود در حالات مشخص شده با رنگ قرمز از راه STUN وجود ندارد و باید به سراغ روش‌های معرفی شده در ادامه برویم.

2. RFC 5766: Traversal Using Relays around NAT (TURN),

این الگوریتم تنها برای ارتباط Peer to Peer استفاده می‌شود و ارتباطات یک به چند را مقدور نمی‌سازد. در این روش، Client به سرور TURN یک پیام allocate می‌فرستد و از او می‌خواهد مقداری از منابع خود را برای مدتی در اختیار او بگذارد تا او بتواند با Peer خود ارتباط برقرار کند. در پاسخ به این درخواست، یک پیام تایید از طرف سرور TURN فرستاده می‌شود که در این پیام، اطلاعات allocated relayed transport address موجود است. سپس کلاینت یک پیام CreatePermissions به این سرور می‌فرستد تا به او بگوید اگر Peer مدنظرش وارد ارتباط با سرور شد، آن را به عنوان یک ارتباط مجاز در نظر بگیرد. سپس، کلاینت پیام خود را (چه باشد UDP چه TCP) به سرور TURN می‌دهد و آن را به صورت UDP به سمت Peer می‌فرستد که شامل آدرس Relay است و Peer نیز روی همین آدرس با UDP پاسخ می‌دهد، و TURN نیز اگر ارتباط را مجاز تشخیص دهد آن را برای Client می‌فرستد. این کار شکست STUN در ارتباط با Symmetric NAT را ندارد زیرا دارای آدرس مشخص Relay می‌باشد. در کل، این روش چون پهنای باند اشغال شده‌ای بیش از STUN دارد، تنها وقتی استفاده می‌شود که Symmetric NAT داشته باشیم (اینکه کدام یک از اینها استفاده شود را ICE مشخص می‌کند که در بخش بعد توضیح داده می‌شود).

3. RFC 5245: Interactive Connectivity Establishment (ICE).

این روش مشخص می‌کند چه آدرس‌های با چه آدرس‌هایی ارتباط برقرار کنند و بالای دوروش مذکور می‌نشیند.ICE در واقع تعیین می‌کند کدام agent ها در شبکه کنترل‌کننده باشند و کدام agent ها کنترل‌شونده.

تمرین دوم:

نمونه یک مکالمه بین امیر و علی در زیر آمده است (chat.py):

```
Enter Your ID (1/2): 1
Login as? (Your Name): Ali
Registration Request Sent to Server: 127.0.0.1 On Port: 1356
Registered
Your Command (Show/End/New): Non Auth Seq received 0
You Have New Message From Amir
You Have New Message From Amir
Your Command (Show/End/New): Show
Whom to Show (Amir): Amir
*** Chatting ***
Hi Ali!
OK?
(ESC to go out)You: Nice
(ESC to go out)You: Bye
(ESC to go out)You: ESC
Your Command (Show/End/New): End
Have a Nice Day :))
```

```
Enter Your ID (1/2): 2
Login as? (Your Name): Amir
Registration Request Sent to Server: 127.0.0.1 On Port: 1356
Registered
Your Command (Show/End/New): New
Enter Your Peer Name: Ali
*** Chatting ***
(ESC to go out)You: Hi Ali!
(ESC to go out)You: OK?
Nice
Bye
Your Peer Left The Chat
Your Command (Show/End/New):End
Have a Nice Day :))
```