به نام خداوند بخشنده‌ی مهربان

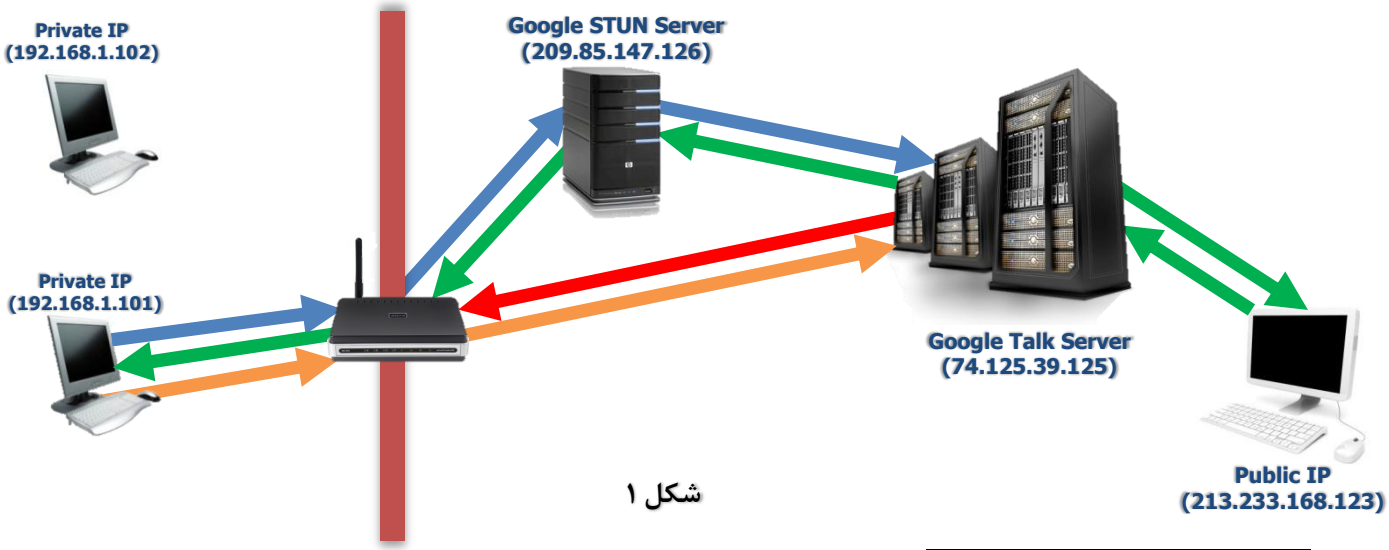
تمرین برنامه‌نویسی سوم[1]
شبکه‌های کامپیوتری

مدرس: مهدی خرازی **پائیز ۱۳۹۰**

در این تمرین شما ضمن آشنایی با چگونگی حل مشکل شبکه‌های پشت NAT، با پیاده‌سازی یک پروتکل، برنامه‌ای متنی برای انجام چت گروهی بدون نیاز به یک سرور مرکزی خواهید نوشت.

مقدمه

نرم‌افزارهایی که امکان چت را به کاربران می‌دهند، برای کاربرانی که پشت NAT هستند باید راهکاری ارائه دهند تا بتوان به این کاربران دسترسی پیدا کرد. راه‌های زیادی برای رفع این مشکل وجود دارد. از جمله این راهکارها STUN[2] و SOCKS هستند. این راهکارها به این صورت عمل می‌کنند که کاربر به یک سرور مرکزی با کمک یک سرور کمکی وصل می‌شود. با این روش تمامی ارتباط هایی که باید با کاربر پشت NAT برقرار شوند با سرور کمکی برقرار می‌شوند.(شکل ۱)



شکل ۱

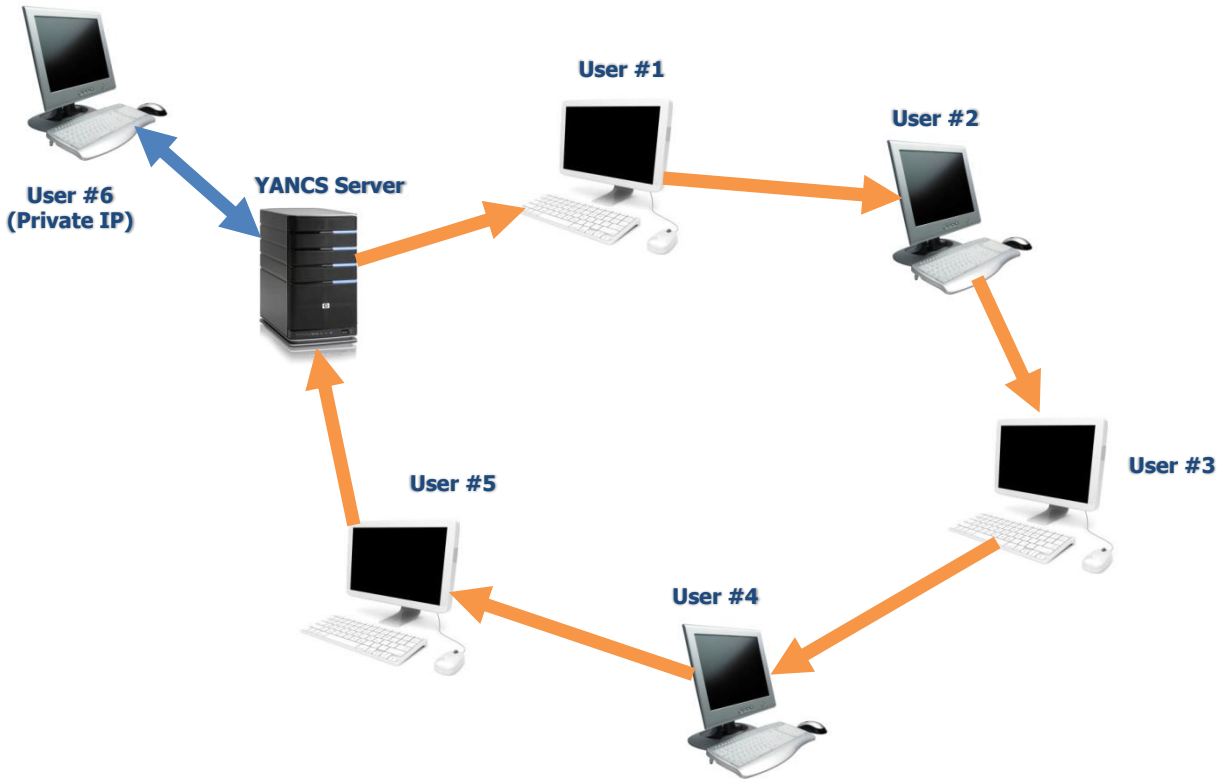
[1] با تشکر از بهنام مؤمنی، امیر شیخها، مهدی احمدی‌نژاد، کامیار اللهوردی، سجاد فولادی و مهرداد مرادی

[2] Session Traversal Utilities for NAT

با طراحی یک DHT[۳] ساده می‌خواهیم امکان چت گروهی را بدون نیاز به یک سرور مرکزی فراهم کنیم؛ در عین حال می‌خواهیم امکان استفاده از برنامه خود را برای کاربران پشت NAT هم فراهم کنیم.

برای حل مشکل کاربران پشت NAT از پروتکل YANCS[۴] که بر پایه پروتکل SOCKS طراحی شده است، استفاده می‌کنیم. به این صورت که کاربر ابتدا به یک سرور YANCS وصل می‌شود و دیگر این سرور که مستقیماً داخل اینترنت قابل دسترس است به جای کاربر داخل چت گروهی قرار خواهد گرفت و تمام اطلاعات دریافتی را به کاربر پشت NAT منتقل خواهد کرد.

جدول توزیع شده بین کاربران را ساده‌ترین شکل ممکن در نظر می‌گیریم. به این صورت که هر گره داخل یک چت گروهی به کاربر بعدی دسترسی خواهد داشت و به این شکل برای هر چت گروهی حلقه‌ای از گره‌ها ساخته می‌شود.(شکل ۲)



شکل ۲

[۳] Distributed Hash Table

[۴] Yet Another NAT traversal protocol for Chat-based Services

برای پیاده‌سازی چت گروهی پروتکلی تعریف می‌کنیم که بر اساس آن کاربران می‌توانند وارد چت گروهی شوند، متن مورد نظر خود را به دیگر کاربران ارسال کنند و در نهایت چت گروهی را ترک کنند. تعریف این پروتکل به شکل زیر است:

پروتکل HDCP[۵]

این پروتکل در لایه Application عمل می‌کند و نحوه ارتباط گره‌ها را برای سه عمل پیوستن به چت گروهی، چت کردن و ترک چت گروهی مشخص می‌کند. فرمت عمومی اطلاعات ارسالی توسط این پروتکل به شکل زیر است:

```
<Command Name>
<Content>
```

توجه داشته باشید که در تمامی این دستورها برای ایجاد خط جدید باید از \r\n استفاده کنید. (به استثنای خط آخر)

پیوستن به چت گروهی

برای شروع کاربر به یکی از گره‌های قابل دسترس خود پیغام زیر را می‌فرستد:

```
init
```

گره‌ای که این پیغام را دریافت می‌کند، در صورتی که درون چت گروهی نباشد، باید ابتدا یک شناسه‌ی ***۸ بایتی*** از کاراکترهای قابل نمایش، منحصر به این چت گروهی تولید کند. در صورتی که درون چت گروهی بود، از شناسه‌ی موجود چت گروهی استفاده می‌کند. در هر صورت در پاسخ پیغام زیر را باید بفرستد:

```
accept
<unique-id>
<next-IP>
<next-MAC>
```

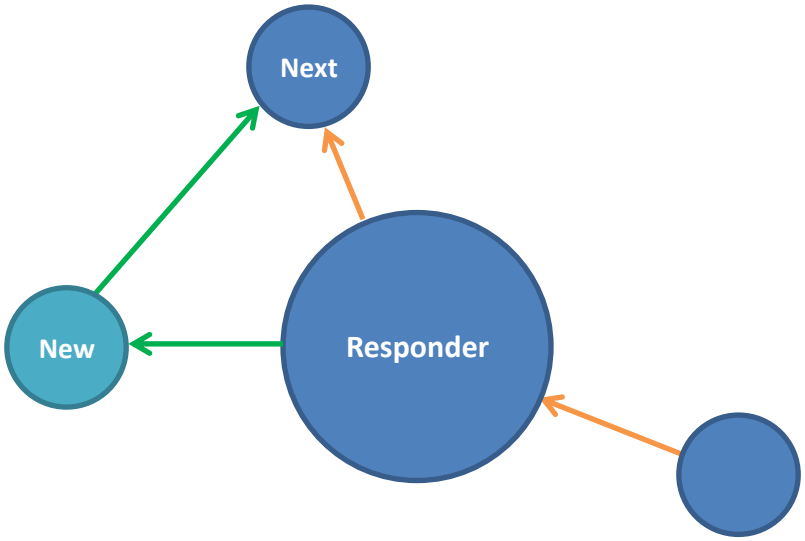
فقط در صورتی که چت گروهی از قبل وجود نداشته باشد، گره باید مقدار <next-IP> و <next-MAC> را برابر با اطلاعات خود قرار دهد، در غیراینصورت مقدار آن گره‌ی بعدی موجود در چت گروهی خواهد بود. همچنین این گره باید اطلاعات گره بعدی خود را نیز به گره‌ی درخواست دهنده تغییر دهد (شکل ۳). کاربر درخواست دهنده با دریافت این پیغام باید شناسه‌ی یکتا را برای چت کردن نگه‌داری کند و گره بعدی خود را برابر با IP و MAC موجود در پیغام دریافتی قرار دهد.
در صورتی که کاربر پشت NAT قرار داشته باشد، با فرستادن درخواست به تمام گره‌ها پاسخی دریافت نمی‌کند. در این صورت باید ابتدا به یک سرور YANCS، به شکلی که در بخش ضمیمه آمده است، وصل شود و پس از آن درخواست خود را تکرار کند.

در ادامه گره پاسخ دهنده، علاوه بر ارسال پیغام پذیرش به گره جدید، باید پیغام زیر را به گره بعدی خود ارسال کند:

```
info
<new-IP>
<new-MAC>
```

[۵] Hypertext Distributed Chat Protocol

گره بعدی با دریافت این پیغام در صورتی که توسط سرور YANCS در حلقه حضور داشته باشد، امکان دریافت اطلاعات را از این گره جدید باید فراهم کند (طریقه‌ی انجام این کار در بخش ضمیمه آورده شده است).



شکل ۳

چت کردن

هر کاربر برای چت کردن باید پیغام را در تمام حلقه چت گروهی پخش کند. برای جلوگیری از شنود پیغام توسط واسطهایی که در چت گروهی نیستند (از جمله سرور YANCS)، اطلاعات چت باید رمز شود[۶]. برای این منظور از الگوریتم DES[۷] استفاده می‌کنیم. فرمت چت ارسالی به شکل زیر خواهد بود:

```
chat
<unique-id>
<username>
<ip-address>
<chat-text>
```

بخش مشخص شده باید توسط الگوریتم رمزنگاری DES با کلید unique-id که همان شناسه‌ی یکتای چت گروهی است، رمز شود[۸]. پس از رمز کردن بخش مشخص شده، این پیغام برای گره بعدی فرستاده می‌شود. گره‌ای که این پیغام را دریافت می‌کند با کمک شناسه‌ی چت گروهی پیغام را رمزگشایی می‌کند و در صورتی که شناسه‌ی موجود در آن درست بود، پیغام را به گره‌ی بعدی ارسال می‌کند. این کار تا آنجایی ادامه پیدا می‌کند که گره بعدی IPای برابر با IP موجود در پیغام داشته باشد که در این صورت ارسال پیغام خاتمه پیدا می‌کند.

نمونه‌ای از پیغام قبل و بعد از رمزنگاری:

```
chat
password
user1
81.31.162.2
This is an encrypted message.
```

Encrypt

```
chat
&?O8???hA?  R???"?
    Z??%?z?:h?
        ???#?`RK?'?E]
```

دقت کنید که پس از رمزنگاری لزوماً کاراکترهای قابل نمایش ایجاد نخواهند شد، به همین دلیل این بخش از پیغام باینری خواهد بود و در نتیجه برخلاف پیغام‌های قبلی امکان ذخیره پیغام چت را در یک رشته نخواهید داشت.

تمامی گره‌ها با دریافت پیغام و رمزگشایی آن، در صورت عدم تطابق شناسه یکتا با شناسه‌ی خود، packet را drop خواهند کرد. همچنین پیغام malformed chat message را نمایش می‌دهند.

---

[۶] از آنجایی که پیغام پذیرش رمز نمی‌شود، امکان دزدیده شدن شناسه یکتا توسط واسطهای شبکه وجود دارد. این رمزگذاری مانع کسانی می‌شود که صرفاً اطلاعات رد و بدل شده در شبکه را به صورت متن می‌خوانند و هیچ دانشی نسبت به پروتکل ما ندارند.

[۷] Data Encryption Standard

[۸] برای رمز کردن نیازی به پیاده‌سازی الگوریتم نیست و می‌توانید از توابع آماده در کتابخانه OpenSSL استفاده کنید. یکی از این توابع DES_cfb64_encrypt است که امکان رمزنگاری و رمزگشایی را برای هر تعداد بایت از داده فراهم می‌کند. از سایت این کتابخانه می‌توانید توضیحات بیشتری پیرامون نحوه استفاده از این تابع پیدا کنید.

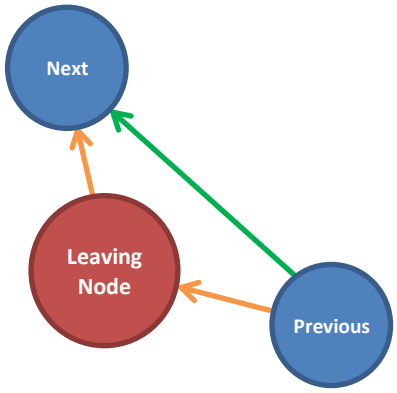
ترک چت گروهی

برای ترک کردن چت گروهی، گره مورد نظر پیغام زیر را به گره‌ی بعدی می‌فرستد. این پیغام حاوی شناسه چت گروهی و IP خود و IP و MAC گره بعدی است:

```
leave
<unique-id>
<leaving-node-ip>
<leaving-node-next-ip>
<leaving-node-next-MAC>
```

که بخش مشخص شده باید مانند قسمت قبل، رمزنگاری شود. مقدار <leaving-node-ip> آدرس IP خود گره است. مقادیر <leaving-node-next-ip> و <leaving-node-next-mac> اطلاعات گره‌ی بعدی است.

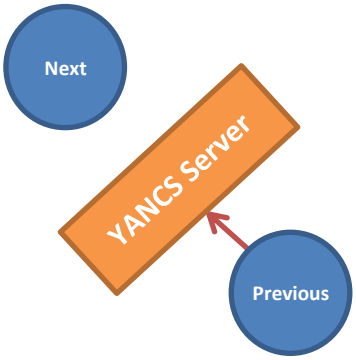
این پیغام توسط گره‌های بعدی دریافت می‌شود، در صورتی که شناسه‌ی یکتای آن درست بود به گره‌ی بعدی ارسال می‌شود. این کار تا آنجایی ادامه پیدا می‌کند که گره‌ی ماقبل گره‌ای که در حال ترک چت گروهی است این پیغام را دریافت کند که با توجه به IP گره که در پیغام وجود دارد، متوجه این امر می‌شود. در ادامه این گره اطلاعات گره‌ی بعدی خود را با توجه به اطلاعات موجود در پیغام دریافتی به‌روز می کند (شکل ۴).



شکل ۴

در اینجا هم تمامی گره‌ها با دریافت پیغام و رمزگشایی آن، در صورت عدم تطابق شناسه یکتا با شناسه‌ی خود، packet را drop خواهند کرد. همچنین پیغام malformed chat message را نمایش می‌دهند.

با توجه به اینکه امکان دارد گره‌ی Next توسط یک سرور YANCS به چت گروهی متصل باشد، گره‌ی Previous امکان اتصال به Next را نخواهد داشت، چون که Next به سرور YANCS اعلام نکرده است که پکت‌های ارسال شده از طرف Previous باید به این گره ارسال شوند.

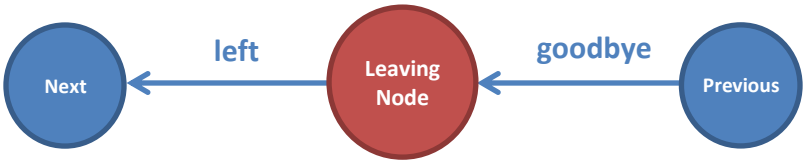


به همین منظور گره Previous ابتدا پیغام زیر را به گره‌ای که در حال ترک چت گروهی است ارسال می‌کند:

```
goodbye
<current-node-ip>
<current-node-mac>
```

گره با دریافت این پیغام، با کمک اطلاعات گره قبلی خود که اکنون به آن دسترسی دارد، به گره Next پیغام زیر را ارسال می‌کند، تا گره Next در صورتی که از طریق سرور YANCS متصل است بتواند این ارتباط را تعریف کند (توضیحات بیشتر در ضمیمه ۱).

```
left
<previous-node-ip>
<previous-node-mac>
```



محیط

در این تمرین نیز، مانند تمرین قبل، از محیط پرتو برای اجرای برنامه شما استفاده خواهد شد. برای اطلاعات بیشتر به مستند «راهنمای چارچوب کاربر» مراجعه فرمایید.

انتظارات

شما باید برنامهٔ مربوط به کاربر و کارگزار YANCS را بصورت جداگانه بنویسید.

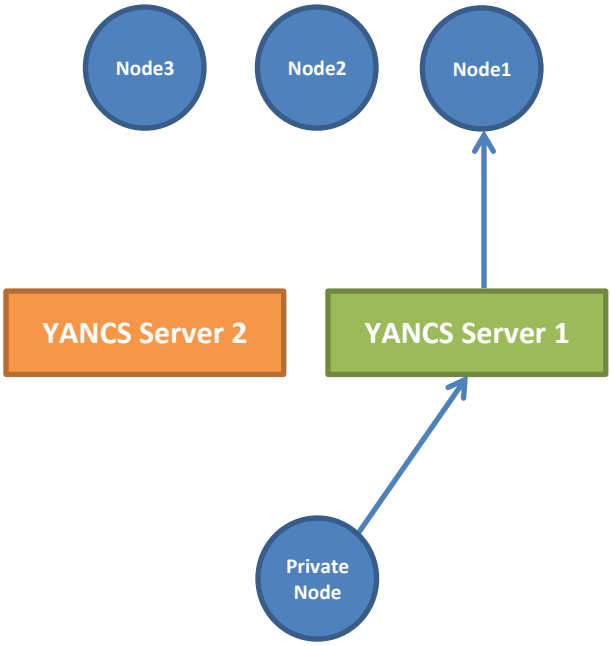
برنامه‌ها

برنامه‌ی مربوط به کاربر

برنامه‌ی کاربر باید پروتکل HDCP را پیاده سازی کند و دستورات زیر را پشتیبانی کند:

دستور	مثال	توضیحات
setusername <username>	setusername chatterbox	**تغییر نام کاربری(مقدار پیش‌فرض default است)**
init	init	**شروع یا اضافه شدن به چت گروهی**
chat <text>	chat this is my text message	**ارسال پیغام به کاربران در چت گروهی**
leave	leave	**ترک کردن چت گروهی**
print-next	print-next	**نمایش IP و MAC گره بعدی (به ترتیب)، در صورتی که گره در چت گروهی قرار ندارد، عبارت Not Connected باید نمایش داده شود**
print-uuid	print-uuid	**نمایش شناسه‌ی یکتای چت گروهی، مانند حالت قبل در صورت نبودن در چت گروهی، عبارت Not Connected باید نمایش داده شود**

هر گره با دریافت پیغام init اتصال به چت گروهی را آغاز می‌کند. این کار با ارسال درخواست برای اولین گره‌ی داده شده در بخش Custom Information انجام می‌شود. گره به مدت ۳ ثانیه منتظر پاسخ باید بماند. پس از آن درخواست را برای گره‌ی بعدی ارسال می‌کند. این کار را تا آخرین گره انجام می‌دهد. در صورتی که پاسخی دریافت نکرد، به این معنا است که این گره پشت NAT قرار دارد. با توجه به این امر ابتدا به اولین سرور YANCS موجود در Custom Information وصل می‌شود. در صورتی که پس از ۳ ثانیه پاسخی دریافت نکرد، درخواست را برای سرور YANCS بعدی ارسال می‌کند. این کار برای تمام سرورهای YANCS در دسترس انجام می‌شود. در صورتی که پاسخ دریافت شد گره ارتباط را به شکل قبل ادامه می‌دهد با این تفاوت که از این به بعد پکت‌ها را باید برای این سرور YANCS ارسال کند.



در صورتی که هیچکدام از سرورهای YANCS هم جواب ندادند، با نمایش Connection failed از تلاش دست می‌کشد.

با دریافت هر پیغام چت گروهی مرتبط با چت گروهی که در آن قرار دارد، باید پیغام آن را به شکل زیر در خروجی نمایش دهد:

```
<username>: <chat-message>
```

اطلاعات مربوط به گره‌های قابل دسترس و سرورهای YANCS، در Custom Information آن می‌آید.

```
<group-chat-port>
<number-of-public-ip-nodes>
<public-ip-1>
<public-mac-1>
…
<public-ip-n>
<public-mac-n>
<number-of-socks-servers>
<yancs-server-ip-1>
<yancs-server-port-1>
<yancs-server-mac-1>
…
```

برنامه‌ی مربوط به سرور YANCS

این برنامه باید پروتکل YANCS را که در بخش ضمیمه آمده است پیاده سازی کند. در بخش Custom Information آن portی که باید روی آن به درخواست‌ها پاسخگو باشد می‌آید:

```
<yancs-server-port>
```

این برنامه باید دستورات زیرا در خط فرمان پشتیبانی کند:

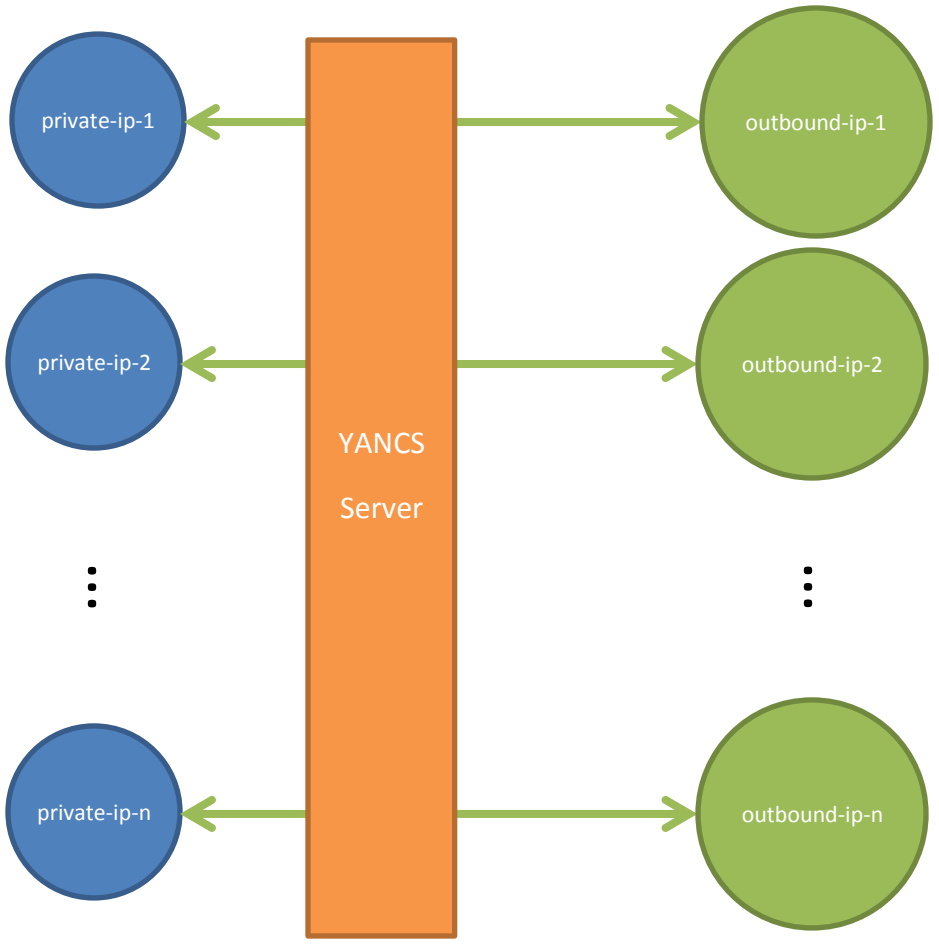
دستور	توضیحات
print-outbounds	**این دستور باید تمام ارتباطات دوطرفه موجود را نمایش دهد.**
print-inbounds	**این دستور باید تمام ارتباط یک طرفه که به سمت گره‌های داخلی هستند را نمایش دهد.**

فرمت خروجی این دستورها باید به شکل زیر باشد:

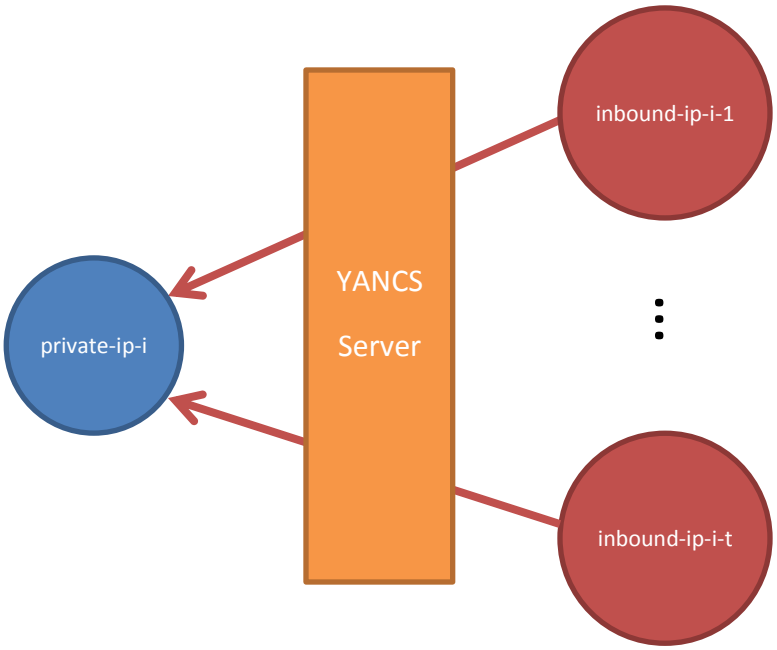
print-outbounds
[private-ip-1] Outbound:
[outbound-ip-1]

...
[private-ip-n] Outbound:
[outbound-ip-n]

در اینجا private-ip-1 آدرس گره‌ی اولی است که پشت سرور YANCS قرار دارد که در زیر آن آدرس گره‌ی بیرونی که با آن ارتباط دوطرفه دارد آمده است.



print-inbounds
[private-ip-1] Inbounds:
[inbound-ip-1-1]
…
[inbound-ip-1-m]
…
[private-ip-n] Inbounds:
[inbound-ip-n-1]
…
[inbound-ip-n-k]
در اینجا در زیر آدرس هر گره‌ی پشت سرور YANCS، آدرس تمام گره‌های بیرونی که امکان دسترسی به این گره را دارند آورده شده است.



نکات ضروری

- در صورتیکه هر مشکل یا پرسشی داشتید که فکر می‌کنید پاسخ آن برای همه مفید خواهد بود، لطفاً آن را به گروه پستی درس ارسال کنید.
- از فرستادن جواب تمرین به گروه پستی جداً خودداری کنید.
- فرستادن کل یا قسمتی از برنامه‌تان برای افراد دیگر، یا استفاده از کل یا قسمتی از برنامهٔ فردی دیگر به نام خود، تقلب محسوب می‌شود.
- پس از اتمام کارتان لازم است که پوشه user را به همراه Makefile فشرده کرده و بر روی سیستم خودکار داوری[9] upload کنید. دقت کنید که قبل از انجام اینکار حتماً پروژه را make clean کرده تا شامل فایل‌های دودویی نباشد.

موفق باشید

[9] http://partov.sharif.edu/judge

ضمیمه ۱: پروتکل YANCS[۱۰]

پروتکل YANCS، پروتکلی است که پکت‌های یک Client و یک Server را از طریق یک سرور نماینده رد و بدل می‌کند. این پروتکل در لایه ۵، لایه session، عمل می‌کند و بر پایه پروتکل SOCKS نوشته شده است.

پروتکل SOCKS دو نسخه ۴ و ۵ دارد که نسخه‌ی ۴ آن تنها برای اتصالات TCP است ولی در نسخه ۵ امکان ارتباط با UDP هم اضافه شده است. با توجه به اینکه برنامه‌ی چت گروهی را می‌خواهیم تنها با UDP پیاده‌سازی کنیم از پروتکل YANCS که نسخه-ی ساده شده‌ی این پروتکل است استفاده می‌کنیم.

برای ارتباطات UDP کاربر کافی است پیغامی را به صورت زیر با پروتکل UDP برای سرور YANCS ارسال کند:

YANCS Request	Type	Destination MAC	Destination IP	Destination Port
1 byte	1 byte	6 bytes	4 bytes	2 bytes

مقدار YANCS Request همواره باید صفر باشد تا نشانگر یک درخواست معتبر YANCS باشد. همچنین مقادیر IP و Port باید به صورت network-byte-order مقداردهی شوند. مقدار Type به صورت زیر باید باشد:

Status	Description
0	Inbound Connection
1	Two-Way Connection

ارتباطات از نوع Inbound تنها این امکان را می‌دهند که امکان شروع یک اتصال از یک گره‌ی بیرونی وجود داشته باشد. در مقابل ارتباطات دوطرفه، امکان شروع ارتباط از دو طرف را ایجاد می‌کنند.

توجه کنید که سرور YANCS تنها یک درخواست را برای Two-Way Connection می‌تواند بپذیرد. در مقابل به هر تعداد درخواست از نوع Inbound قابل قبول است. در هرصورت سرور YANCS به صورت زیر پاسخ می‌دهد:

Status	Public MAC	Public IP
1 byte	6 bytes	4 bytes

سرورهای YANCS می‌توانند چند interface داشته باشند. در این تمرین سرورهای YANCS دو interface دارند. یکی از آن-ها برای اتصال به گره‌های پشت NAT هستند و دیگری برای اتصال به دنیای بیرون. سرور YANCS در پاسخ به یک درخواست باید مقادیر Public MAC و Public IP را برابر با IP و MAC اینترفیس دیگر خود که قابلیت اتصال به دنیای بیرون را دارد قرار دهد. به طوری که گره‌های موجود در دنیای بیرون تنها Public IP و Public MAC را می‌بینند.

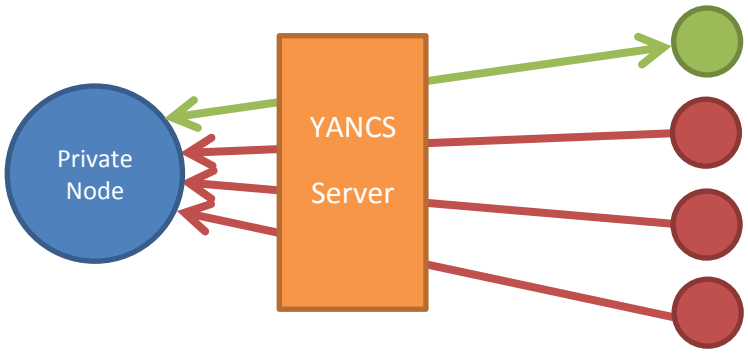
[۱۰] Yet Another NAT traversal protocol for Chat-based Services

توجه کنید مشخصات این interface بیرونی صرفاً یک اطلاع اضافه است که به گره‌ی درخواست دهنده داده می‌شود. گره باید مقصد بسته‌ها را سرور YANCS قرار دهد و در ادامه سرور با اطلاعاتی که دارد بسته را از اینترفیس بیرونی خود به مقصد مورد نظر ارسال کند. تنها استفاده‌ای که از این اطلاعات در پروتکل HDCP می‌شود برای ارسال بسته‌هایی است که نیاز به درج آدرس این گره در آن‌ها وجود دارد. در این بسته‌ها گره باید از IP و MAC عمومی سرور YANCS در پیام‌ها استفاده کند.

Status حاوی وضعیت برقراری ارتباط است که تنها دو مقدار زیر را می‌تواند داشته باشد:

Status	Description
0	Success
1	Error

برقراری ارتباط همواره موفقیت آمیز خواهد بود مگر در حالتی که یک ارتباط نامعتبر اضافه شود.
ارتباطی نامعتبر خواهد بود که در آن یک گره‌ی بیرونی را که در حال حاضر به یک گره درونی مرتبط است، به یک گره‌ی درونی دیگر مرتبط کند. در این صورت باید سرور YANCS مقدار Status را برابر ۱ قرار دهد تا ناموفق بودن درخواست را اعلام کند (مقادیر بقیه‌ی فیلدها نادیده گرفته می‌شوند). گره‌ی درونی با گرفتن این پیغام کار خود را با سرور YANCS بعدی ادامه می‌دهد.



ضمیمه ۲: ویترین[۱۱]

در این بخش چند سناریو از نحوه کار برنامه آورده شده است. خروجی‌ها بر پایه گره‌های آزمایشی هستند که در اختیار شما قرار خواهند گرفت. در نتیجه صرفاً برای بررسی یک نمونه از اجرای برنامه می‌توانید از آن‌ها استفاده کنید. به پیاده‌سازی‌های مبتنی بر توپولوژی شبکه نمره‌ای تعلق نخواهد گرفت.

در این سناریوها آدرس گره‌ها به شکل زیر است(تنها آدرس public سرور YANCS نوشته شده است):

Node 1	Node 2	Node 3	Node 4	YANCS Server 2
213.233.171.4	213.233.171.14	213.233.171.24	192.168.19.34	213.233.171.74

سناریو ۱: تشکیل چت گروهی با گره‌های Public

در Node 1 دستور init وارد می‌شود. سپس در Node 3 دستور init وارد می‌شود. در این صورت ابتدا Node 1 به Node 2 وصل می‌شود، سپس Node 3 به این دو وصل می‌شود. خروجی print-next برای این سه گره به شکل زیر خواهد شد:

Node 1	Node 2	Node 3
print-next 213.233.171.24 0:24:8c:0:20:1	**print-next** 213.233.171.4 0:24:8c:0:2:1	**print-next** 213.233.171.14 0:24:8c:0:10:1

پس از زدن دستور leave در گره دوم خروجی‌ها به شکل زیر تغییر خواهند کرد:

Node 1	Node 2	Node 3
print-next 213.233.171.24 0:24:8c:0:20:1	**print-next** Not Connected	**print-next** 213.233.171.4 0:24:8c:0:2:1

سناریو ۲: تشکیل چت گروهی با گره‌ی Private

با ادامه از سناریوی قبلی، گره ۴ام را وارد شبکه می‌کنیم. از آنجایی که این گره private است امکان اتصال به گره‌های دیگر را ندارد. با شروع درخواست به یک سرور YANCS به شبکه متصل می‌شود. خروجی جدید به شکل زیر خواهد بود:

Node 1	Node 2	Node 3	Node 4
print-next 213.233.171.74 0:24:8c:0:70:1	**print-next** Not Connected	**print-next** 213.233.171.4 0:24:8c:0:2:1	**print-next** 213.233.171.24 0:24:8c:0:20:1

[۱۱] Showcase

در این حالت خروجی سرور YANCS برای اتصالات تعریف شده به شکل زیر خواهد بود:

YANCS Server 2
print-outbounds 192.168.19.34 Outbound: 213.233.171.24 **print-inbounds** 192.168.19.34 Inbounds: 213.233.171.4

در ادامه ابتدا دستور init را برای گره ۲ اجرا می‌کنیم، سپس دستور leave را ابتدا برای گره ۱ و سپس ۳ اجرا می‌کنیم. تغییرات خروجی‌ها به شکل زیر خواهد بود:

Node 1	Node 2	Node 3	Node 4
print-next Not Connected	**print-next** 213.233.171.74 0:24:8c:0:70:1	**print-next** Not Connected	**print-next** 213.233.171.14 0:24:8c:0:10:1

YANCS Server 2
print-outbounds 192.168.19.34 Outbound: 213.233.171.14 **print-inbounds** 192.168.19.34 Inbounds: 213.233.171.4 213.233.171.14 213.233.171.24

ضمیمه ۳: ANSI escape code

برای خوانایی بیشتر خروجی‌های خود می‌توانید با روشی که در ادامه آورده شده است با رنگ‌های متفاوت در خروجی استاندارد بنویسید.

برخی کاراکترهای خاص را در صورتی که به فرمت مشخصی در ابتدا و انتهای رشته‌های خود اضافه کنید، می‌توانید رنگ نوشته را تغییر دهید:

\033[	Bold	;	Color Code	m	Text	\033[0m
\033[	1	;	32	m	Green	\033[0m

به طور مثال رشته زیر درکنسول نوشته Green را به رنگ سبز خواهد نوشت:

```
\033[1;32mGreen\033[0m
```

مقدار Color Code از جدول زیر قابل تعیین است:

Color	Code
Red	**31**
Green	**32**
Yellow	**33**
Blue	**34**
White	**35**