به نام خدا

تمرین صفرم | سیستم‌های توزیع‌شده

مدرس: صابر صالح | ۲-۱۳۹۹

نکات مهم

لطفا ابتدا به نکات زیر توجه کنید:

- برای پیاده‌سازی این تمرین از زبان Python3 استفاده نمایید.

- شما باید جواب این تمرین را به صورت یک فایل در صفحه‌ی درس در سامانه Quera بارگذاری نمایید.

- رمز کلاس درس در سامانه Quera، EE539 است.

- مهلت ارسال تمرین تا پایان روز **۱۳۹۹/۱۲/۱۵** می‌باشد.

موفق باشید.

۱ مقدمه

سوکت‌ها نقاط پایانی ارسال و دریافت داده هستند که پردازه‌های مختلف در یک سیستم توزیع‌شده به کمک آنها می‌توانند با یکدیگر ارتباط برقرار کنند و داده ارسال یا دریافت نمایند.
هدف این تمرین آن است که با برنامه‌نویسی سوکت آشنا شده و یک ارتباط ساده را برقرار نمایید.

۲ شرح تمرین

در این تمرین می‌خواهیم ارتباط بین دو گره را پیاده‌سازی کنیم. بدین صورت که هر گره توانایی ارسال و دریافت اطلاعات را داشته باشد اما در ابتدا گره شروع‌کننده باید مشخص باشد. بنابراین فرمت ورودی به صورت زیر است:

```
<r/s> IP1 PORT1 IP2 PORT2
```

ورودی اول یک حرف r یا S است. ورودی S به این معناست که این گره ارتباط را آغاز خواهد کرد. در ادامه، به ترتیب آی‌پی و درگاهی که گره روی آن منتظر دریافت پیام است و آی‌پی گرهی که قرار است به آن وصل شود، می‌آید. اجرا به این صورت است که ابتدا یک‌بار برنامه شما گیرنده و برای مثال ورودی

```
r 192.168.142.137 54329 192.168.142.139 54328
```

و سپس به عنوان فرستنده(شروع‌کننده) با ورودی زیر اجرا می‌شود.

```
s 192.168.142.139 54328 192.168.142.137 54329
```

در اجرای اول گره روی ۱۹۲.۱۶۸.۱۴۲.۱۳۷ و درگاه ۵۴۳۲۹ منتظر دریافت می‌گردد و در اجرای دوم، گره (شروع‌کننده‌ی ارتباط) به درگاه ۵۴۳۲۹ از ۱۹۲.۱۶۸.۱۴۲.۱۳۷ پیامی را ارسال کرده و خود روی آدرس ۱۹۲.۱۶۸.۱۴۲.۱۳۹

درگاه ۵۴۳۲۸ منتظر پاسخ می‌ماند. بنابراین در این تمرین ارتباط گره‌ها با یکدیگر به صورت همتابه‌همتا[1] است.

چند نکته:

۱. توجه کنید که برنامه‌ی شما در دو ترمینال به عنوان دو پردازه‌ی مختلف اجرا خواهد شد.

۲. آدرس آی‌پی به منظور برقرای ارتباط میان دو سیستم مجزا لازم است.

۳. جهت تست و بررسی برروی یک سیستم می‌توانید جای آیپی ها مقدار localhost یا ۱۲۷.۰.۰.۱ را قرار دهید.

۳ قالب ارسال/دریافت پیام

برای یکسان‌سازی نحوه‌ی ورودی و خروجی از قالب زیر استفاده کنید. برای ارسال پیام باید داده‌ی ارسالی را به این شکل تبدیل کرده و ارسال نمایید. در هنگام دریافت نیز طبق همین قالب، داده‌ی مورد نظر را استخراج کنید.

```
{"type": "start", "value": "hi"}
```

بنابراین هر پیام با دوتایی type و value مشخص می‌شود.

٤ سناریو ارسال و دریافت

پس از مشخص کردن شروع‌کننده‌ی ارتباط و درگاه‌های مربوطه پیام‌هایی با قالب مذکور فرستاده می‌شود. این پیام‌ها به همراه پاسخ مناسب آن‌ها در جدول زیر آورده شده است. چنانچه یک گره پیامی دریافت کند طبق این جدول پاسخ آن را ارسال می‌نماید.

[1] peer-to-peer

نوع	پیام دریافت شده	پاسخ
start	hello	hi
end	goodbye	bye

ترتیب ارسال این پیام‌ها به شرح زیر است:

۱. شروع‌کننده‌ی ارتباط که در ابتدا با دریافت s از ورودی مشخص شده، "hello" را به مقصد مورد نظر ارسال می‌کند.

۲. پس از دریافت پاسخ مناسب، به منظور پایان دادن ارتباط "goodbye" را به مقصد می‌فرستد اما ارتباط را تا زمانی که پاسخ مربوطه ("bye") دریافت نشده قطع نمی‌کند.

سناریوی مطرح شده به صورت خلاصه:

```
node1: {"type": "start", "value": "hello"}
node2: {"type": "start", "value": "hi"}
node1: {"type": "end", "value": "goodbye"}
node2: {"type": "end", "value": "bye"}
finished
```

هر گره پس از دریافت یک پیام باید آن را در خروجی چاپ نماید. ابتدا نوع پیام و سپس داده‌ی آن را چاپ کنید.

```
start hello
```

۵ راهنمای تبدیل قالب مورد نظر

برای تبدیل دستورات ورودی به محتوای بسته‌ی ارسالی در پایتون می‌توانید از کد زیر که یک شیء از نوع Message را به آرایه‌ای از نوع بایت برای ارسال در سوکت تبدیل می‌کند استفاده نمایید:

```
import json


class Message:
    def __init__(self, type, value):
        self.type = type
        self.value = value


m = Message("start", "hi")
byte_array = json.dumps(m.__dict__).encode("utf-8")
socket.send(byte_array)
```

تبدیل بسته‌ی دریافتی به شیء از نوع Message:

```
m = Message(**json.loads(received_data, encoding="utf-8"))
```