

دلال پیام - قسمت دوم

پرهام الوانی

۴ دی ۱۴۰۰

فهرست مطالب

۲	۱ مقدمه
۲	۲ کنترل جریان با TCP
۲	۳ بیشترین تعداد ارتباط همزمان TCP
۲	۴ پیاده‌سازی دلال پیام با UDP
۲	۱.۴ پیاده‌سازی دلال پیام با استفاده از UDP
۲	۲.۴ اطمینان از سلامت طرف مقابل
۳	۳.۴ ذخیره کردن اطلاعات طرف مقابل
۳	۵ مقایسه کنترل جریان در TCP و UDP

۱ مقدمه

در قسمت اول پروژه با دلال پیام آشنا شدید. در این قسمت قصد داریم این سامانه را در شرایط مختلف و به منظور یادگیری مفاهیم کنترل جریان و ازدحام بررسی کنیم.

۲ کنترل جریان با TCP

۱. یک نسخه از برنامه ای که در پروژه قبل نوشته اید را در حالت کلاینت subscriber برای موضوع flow-control-tcp اجرا کنید.

۲. نسخه دیگری از برنامه را در حالت کلاینت publish اجرا کنید و به صورت تناوبی (در حلقه) پیام‌هایی را توسط آن داخل سابجکت flow-control-tcp بریزید.

۳. وضعیت بسته های شبکه در فرایند بالا را با استفاده از وایرشارک بررسی و تفسیر کنید

۳ بیشترین تعداد ارتباط همزمان TCP

دلال پیام را همانطور که از قسمت پیشین توسعه داده‌اید، اجرا کنید. همانطور که پیش اشاره شد، کلاینت‌هایی که مشترک می‌شدند یک ارتباط باز با این دلال پیام را نگهداری می‌کردند. کلاینت برنامه را در حالت subscriber چندین بار (بهتر است اجرا و شروع برنامه را در قالب یک حلقه انجام دهید) اجرا کنید. حد بالای تعداد کلاینت‌های در حال اجرا چه تعداد است؟ آیا این تعداد با حداکثر تعداد پورت‌های موجود در ارتباط TCP قابل مقایسه است؟ چه عاملی باعث این محدودیت می‌شود؟

۴ پیاده‌سازی دلال پیام با UDP

قصد داریم دلال پیامی که پیشتر نوشته‌ایم، در کنار پشتیبانی از ارتباط TCP، از ارتباط UDP هم پشتیبانی کند. تفاوت اصلی در ارتباط UDP نبود یک ارتباط پایدار میان سرور و کلاینت است و این مساله در زمان مشترک شدن کلاینت‌ها روی تاپیک‌ها بیشتر خود را نشان می‌دهد چرا که سرور می‌بایست پیام‌ها را برای مشترکین فعال ارسال کند. در ادامه به این موضوعات پرداخته می‌شود.

۱.۴ پیاده‌سازی دلال پیام با استفاده از UDP

مانند قبل سرور می‌بایست بر روی یک پورت و آدرس مشخص قابل دسترسی باشد از این رو سوکتی که برای سرور در نظر گرفته می‌شود با دستور Bind روی یک آدرس و پورت مشخص بسته می‌شود.

در قسمت قبلی پیاده‌سازی TCP برای مدیریت بسته‌های رسیده به هر سوکت از یک Thread استفاده شد. در این حالت می‌توان دریافت بسته ها را همگی در یک Thread انجام داد و مدیریت و کنترل جریان پیام ها را در سرور دلال پیام انجام داد. برنامه شما در پورت ۱۲۳۴ UDP اقدام به شنود بسته‌ها می‌کند. (برخلاف ارتباط TCP که هر کلاینت با ایجاد یک سوکت اختصاصی بسته‌های خود را برای سرور ارسال می‌کند و سرور باید رسیدن بسته ها را در ترد مخصوص سوکت ایجاد شده پردازش کند). شما می‌توانید به محض رسیدن بسته‌ها از هر کلاینتی آن را در سرور خود دریافت کنید. پردازش مبدا بسته‌ها با استفاده از هدرهای بسته‌های UDP انجام می‌شود و باید از اطلاعات آن جهت مدیریت کلاینت‌ها استفاده کنید.

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # UDP
client.bind(('', 1234))
while True:
    data, addr = client.recvfrom(1024)
    print("received message:", data, addr)
```

۲.۴ اطمینان از سلامت طرف مقابل

از آنجایی که در پروتکل UDP مکانیزمی جهت تشخیص برقراری یا قطع ارتباط در لایه ارتباط (transport) وجود ندارد. از شما خواسته شده که این امکان را در لایه های بالاتر (application) فراهم کنید. به این منظور کلاینت پس از ارسال اولین بسته به سرور (معادل برقراری کانکشن در ارتباط TCP) به صورت تناوبی (در یک حلقه) اقدام به چک کردن سرور در قالب پیام‌های ping/pong که در فاز قبلی تعریف کردید می‌کند.

برای این کار لازم است که در **کلاینت** به صورت تناوبی (هر ۱۰ ثانیه یکبار) یک بسته ping برای سرور ارسال کنید. سالم بودن اتصال برای کلاینت به معنای دریافت pong بعد از ارسال ping است. سمت سرور باید بعد از شروع یک اتصال (خواندن یک پیام از آدرس جدید در سوکت) اقدام به چک کردن پیام‌های ping کلاینت کند. دریافت پیام به معنی برقرار بودن اتصال (تا ۱۰ ثانیه دیگر) است و سرور باید برای اطلاع به کلاینت pong را به همان کلاینت ارسال کند.

۳.۴ ذخیره کردن اطلاعات طرف مقابل

کلاینت‌های UDP برای ارتباط با سرور تنها نیاز به پورت و آدرس دلال پیام دارند، اما سرور برای ارسال پیام به کلاینت‌ها می‌بایست از آخرین پورت و آی پی استفاده کند که با آن از کلاینت پیام دریافت کرده است. در نظر داشته باشید که برای این امر نیاز به دستور خاصی ندارید چرا که به صورت پیشفرض یک پورت تصادفی برای کلاینت شما انتخاب شده و می‌تواند از آن برای ارسال یا دریافت اطلاعات استفاده کند.

```
import socket

client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM) # UDP
while True:
    data, addr = client.recvfrom(1024)
    print("received message:", data, addr)
```

در کد نمونه آورده شده سوکت UDP به یک پورت آزاد و تصادفی مانند پورت ۳۷۰۲۰ بسته خواهد شد و به این ترتیب بسته‌های ارسال از پورت مبدأ ۳۷۰۲۰ ارسال شده و از سوی دیگر می‌توان روی این پورت مطابق آنچه در مثال آورده شده است بسته دریافت نمود.

۵ مقایسه کنترل جریان در TCP و UDP

در قسمت قبلی با استفاده از کلاینت TCP که از ارتباط خود چیزی دریافت نمی‌کرد بحث کنترل جریان در پروتکل TCP را دیدید. در این قسمت قصد داریم همین موضوع را در پیاده‌سازی UDP ببینیم. برای این موضوع، کلاینت UDP ای دارید که روی یک موضوع مشترک شده است ولی بسته‌ای را **نمی‌خواند** دقت داشته باشید که یعنی پس از ارسال و دریافت بسته‌های لازم جهت اشتراک بر یک موضوع خاص، دیگر بسته‌ای را دریافت نمی‌کند. یک کلاینت دیگر به صورت تکراری شروع به انتشار در همان موضوع می‌کند. با استفاده از نرم‌افزار Wireshark توضیح دهید چه اتفاقی برای بسته‌های می‌افتد. آیا این بسته‌ها از دست می‌روند؟