آزمایش 5 : کنترل ازدحام TCP و UDP

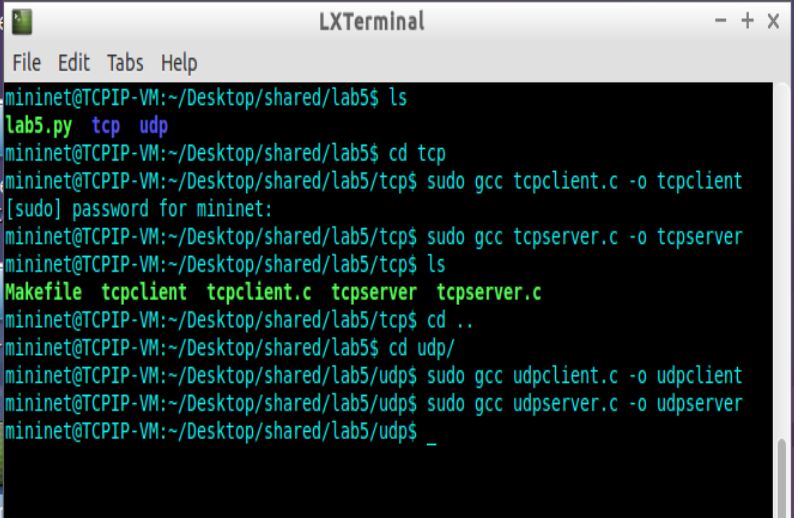
زینب باقیان

امیرمحمد درپوش

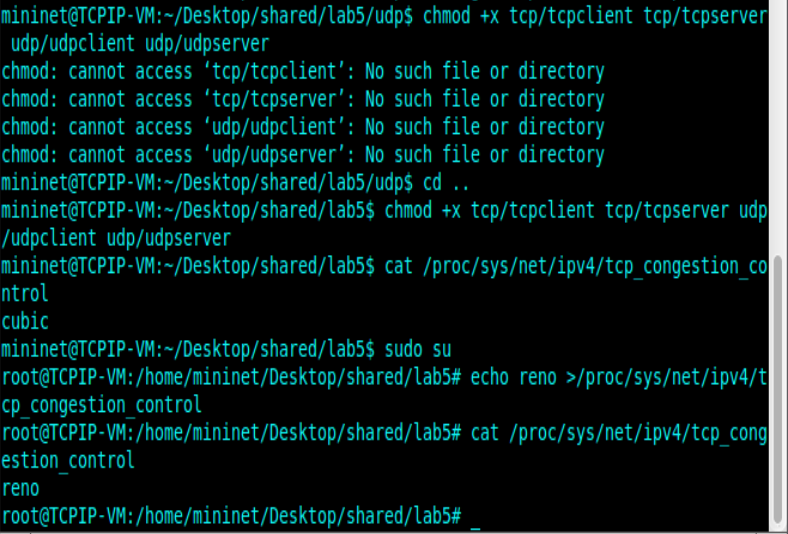
محمدصالح پژند

الف) روال آزمایش:

ابتدا برنامه را راه اندازی میکنیم:

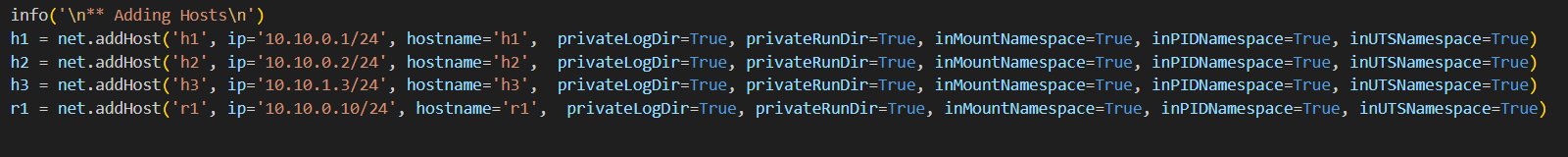


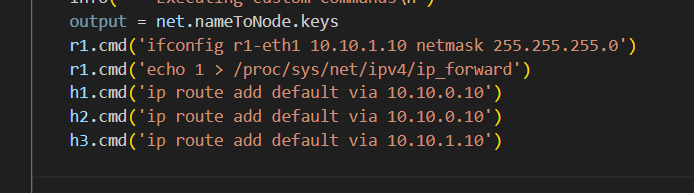
حالا مکانیزم کنترل ازدحام ماشین مجازی را چک میکنیم که در اینجا cubid است پس به reno تغییرش میدهیم:



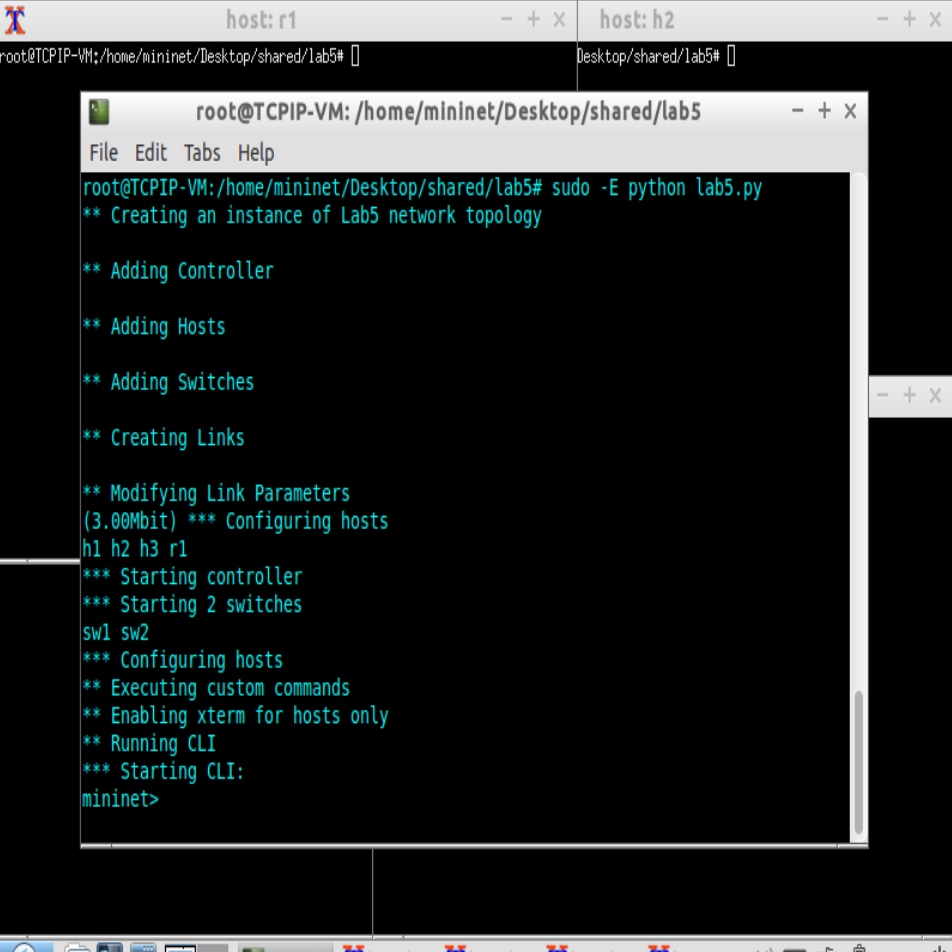
ب) جریان های TCP و UDP

فایل lab5.py را به شکل زیر تغییر میدهیم:

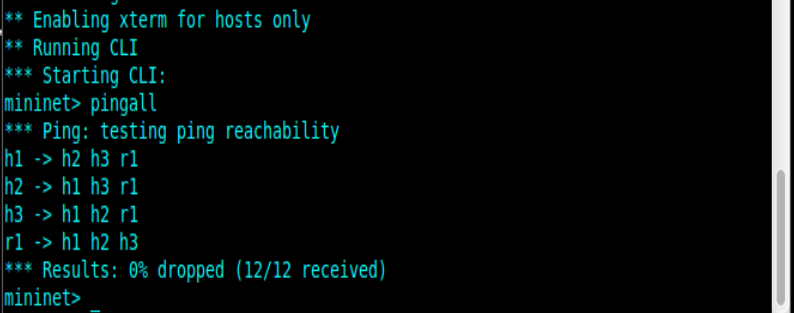




حالا اسکریپت را اجرا میکنیم تا توپولوژی ساخته شود:



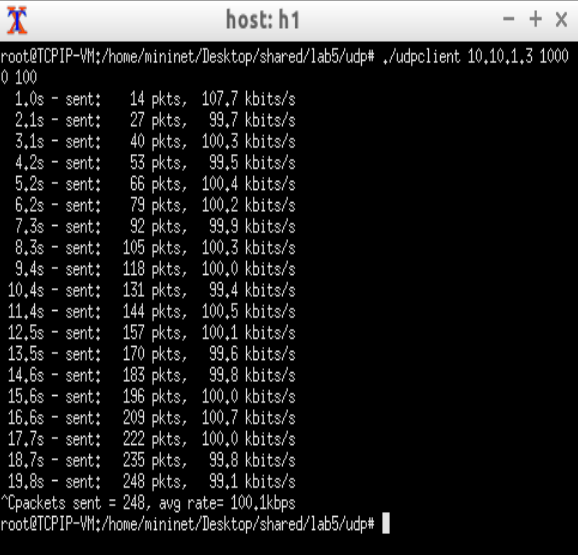
برای اطمینان از اتصال نود ها از pingall استفاده میکنیم:

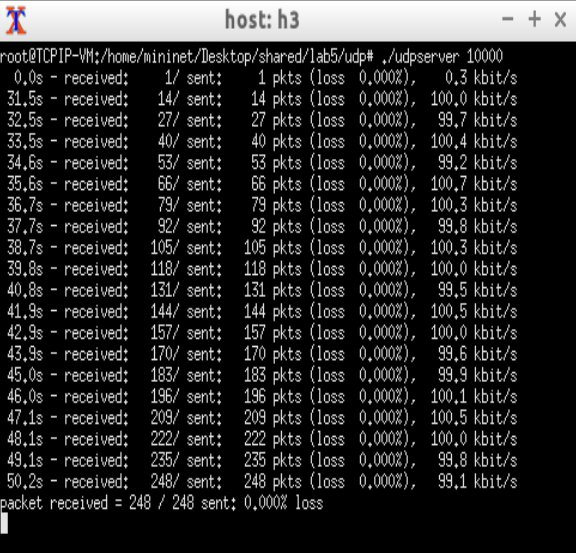


ب-1 ) سوال1:

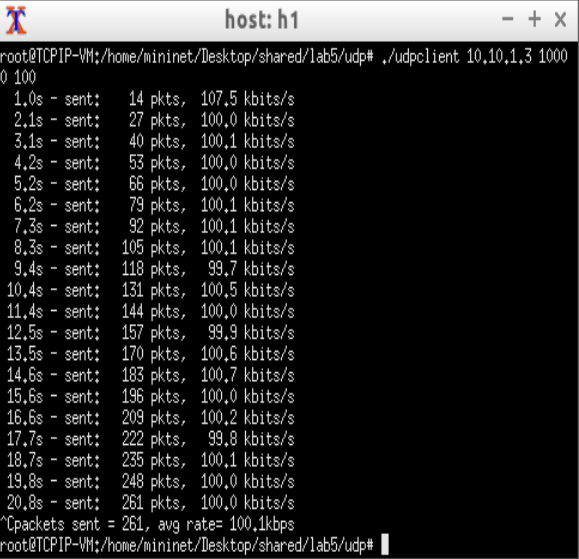
در h3 با یک سرور به پورت 10000 وصل میشویم تا گوش کند و در h1 از طرف کلاینت با پورت 10000 و نرخ 100kbps برای h3 داده میفرستیم:

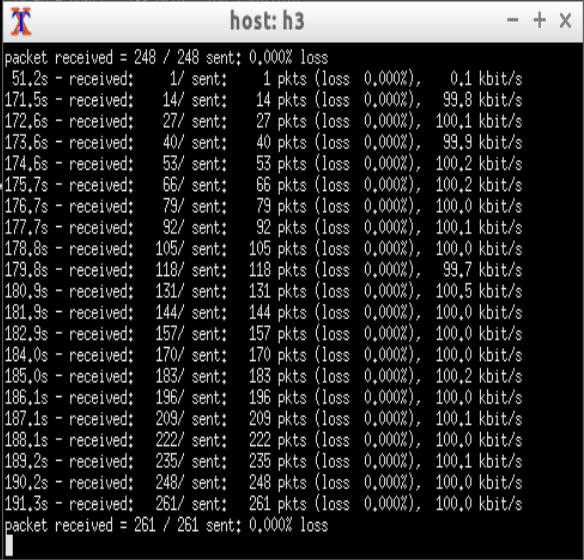
نتایج بار اول:





نتایج بار دوم:

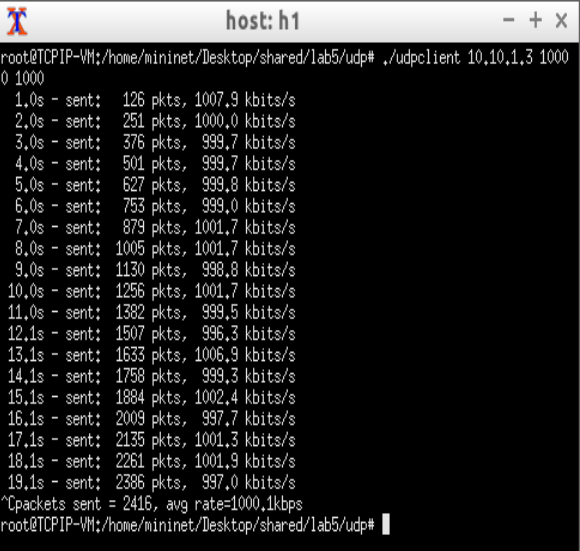


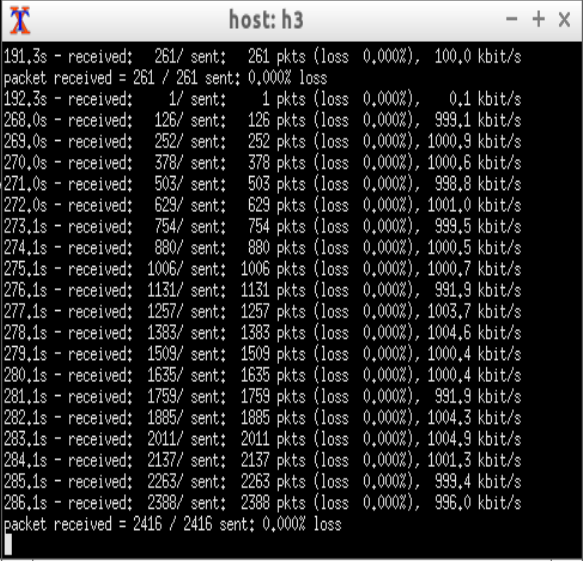


همانطور که میبینیم در هردو تکرار هیچ پکیجی loss نشده و goodput هم مقدار 100kbps را دارد.

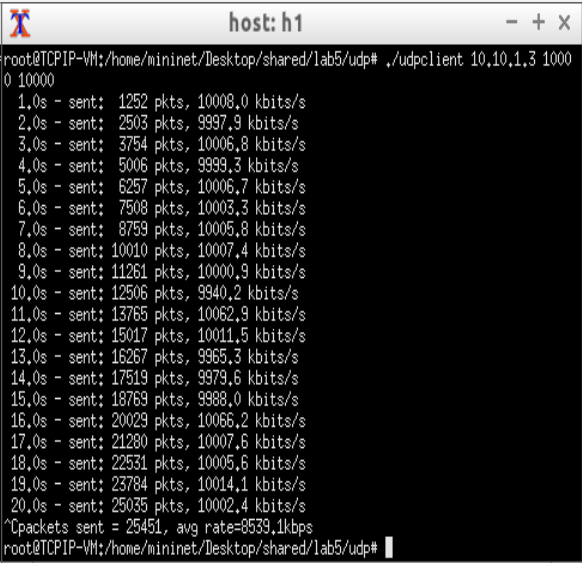
سوال2:

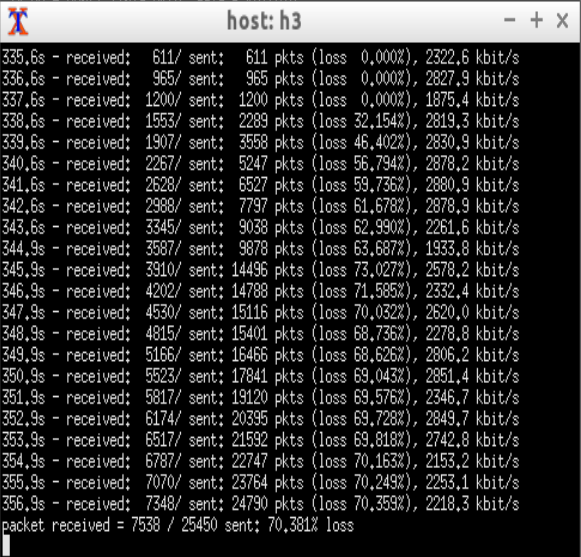
1Mbps test:



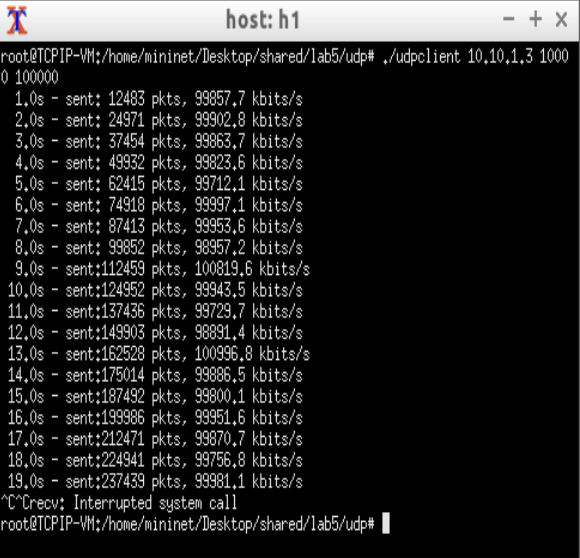


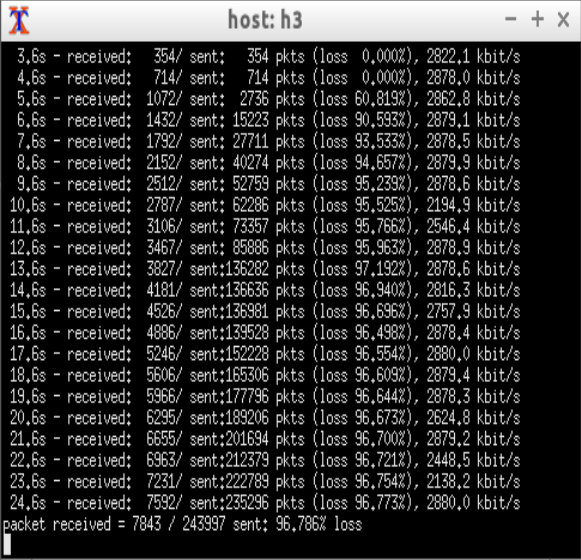
10Mbps test:



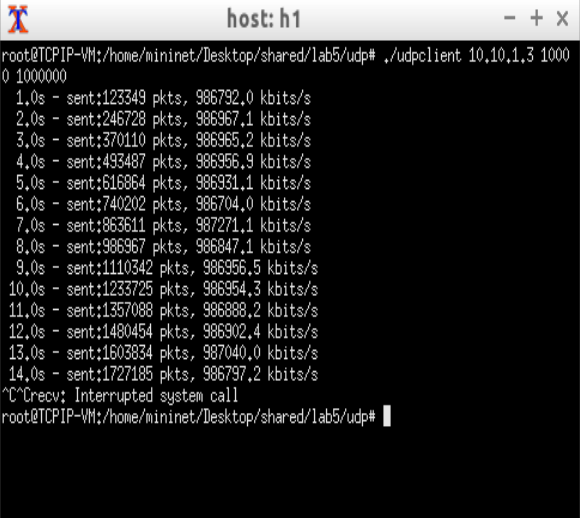


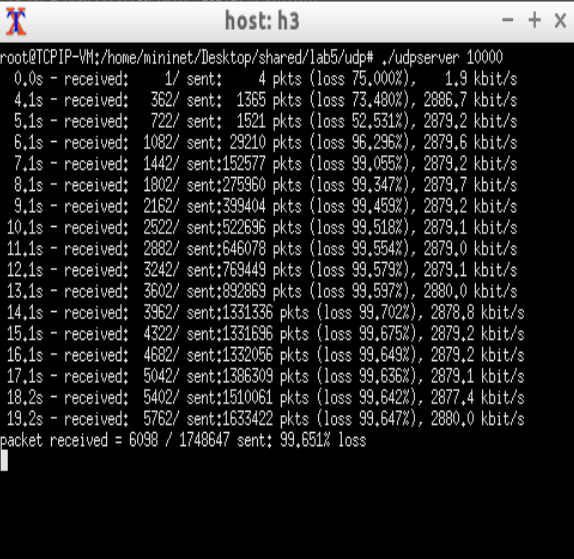
100Mbps test:





1Gbps test:



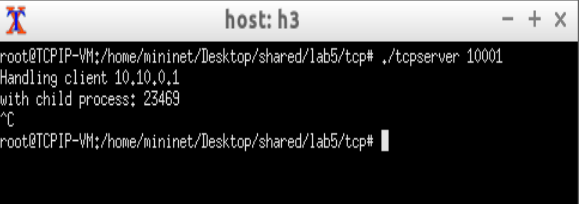


طبق نتایج در این پروتکل به ازای نرخ 10Mbps و بالاتر مقدار loss پس از چند ثانیه بشدت زیاد میشود. این امر احتمالا به دلیل پر شدن ظرفیت لینک بین r1 و sw2 است به اینگونه که با افزایش نرخ ارسال ترافیک در این لینک افزایش میابد و با کردن ظرفیت باعث drop شدن پکیج ها میشود. برای حل این موضوع میتوان با افزایش پهنای باند لینک ها به خصوص r1 و sw2 این امکان را فراهم کنیم تا حجم بیشتری ارسال و دریافت شود و بسته های کمتری loss شوند.

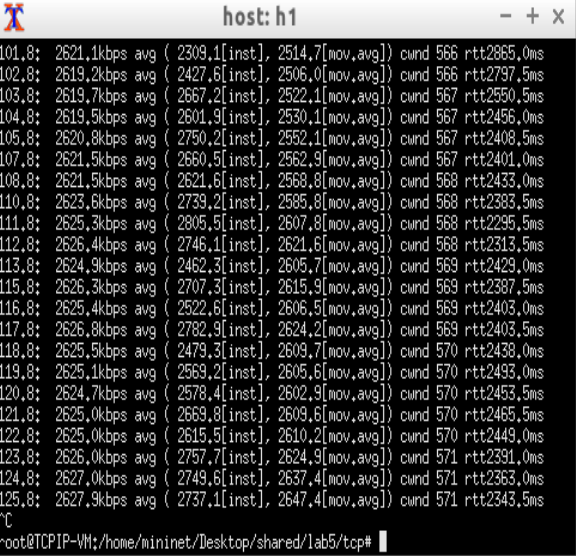
سوال 3:

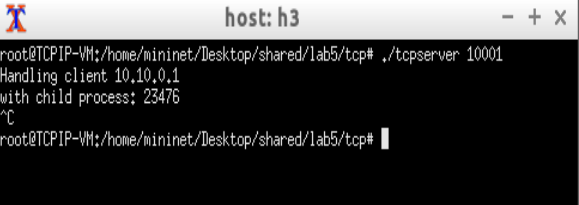
بار اول:





بار دوم:





بعد از گذشت حدود دو دقیقه که مقادیر چاپ شده به حالت پایدار رسیدند و تغییر زیادی نکردند میتوانیم ببینیم که goodput به ترتیب 2536kbps و 2627kbps برای این پروتکل می باشند که این یعنی به طور میانگین 2581.5kbps به عنوان نتیجه نهایی بدست می آید.