بسم الله الرحمن الرحیم

ارزیابی مبتنی بر شبیه سازی عملکرد TCP چندمسیری در پشتیبانی از انتقال زیرجریان های ترافیکی

محمد مهدی سوری

حمیدرضا آذرباد

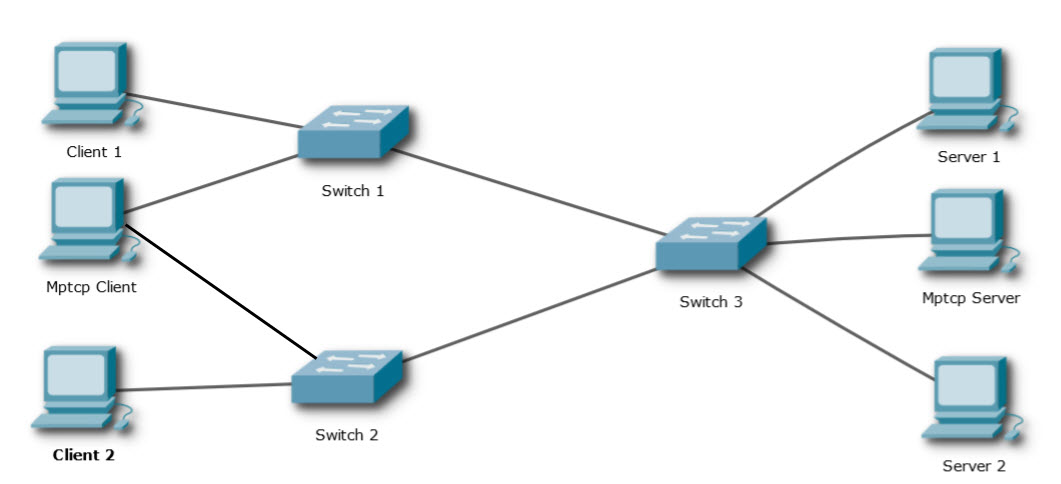
**مقدمه**

در این پروژه قصد داریم که با شبیه سازی TCP چند مسیره (MPTCP) درون محیط شبیه ساز Mininet، و با استفاده از پوسته Minitopo عملکرد این پروتکل را در حالت های مختلفی که برای کنترل ازدحام (Congestion Control)، زمانبند (Scheduler) و مدیریت مسیر (Path Manager) تحت یک توپولوژی و سناریو خاص بررسی کرده و نتایج را گزارش کنیم.

ابتدا توپولوژی و سناریو را شرح می دهیم سپس الگوریتم ها را معرفی کرده و نتایج را گزارش می کنیم.

**توپولوژی**

ابتدا سه هاست را ایجاد می کنیم به عنوان Client که دو تا از آن ها به صورت TCP معمولی عمل می کنند و سومی به صورت MPTCP عمل می کند. کلاینتی که از MPTCP استفاده می کند به دو سوئیچ متصل است و کلاینت های دیگر هر کدام به یک سوئیچ متصل می شوند. این دو سوئیچ به سوئیچ سوم که مخصوص هاست های سرور ها است متصل می شوند و این دو لینک، لینک های bottleneck مان هستند. سه سرور هم ایجاد می کنیم که یکی از آن ها مخصوص MPTCP و دو تای دیگر برای اتصالات TCP معمولی هستند. شکل زیر چگونگی اتصلات این توپولوژی را نمایش می دهد.



سرعت و تاخیر لینک های bottleneck به ترتیب 20Mbps و 20ms است. سرعت و تاخیر مابقی لینک ها به ترتیب 100Mbps و 0.1ms است.

**سناریو**

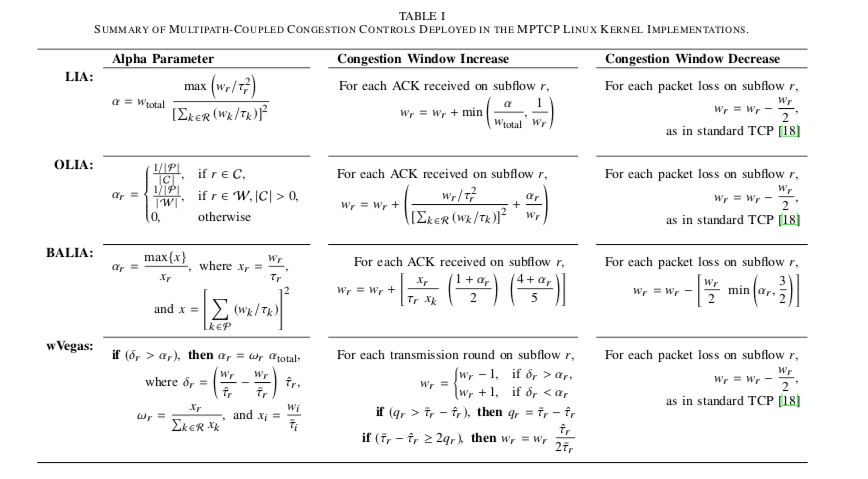
ابتدا از Mptcp client به Mptcp server به مدت 50 ثانیه داده ارسال می کنیم. در همین حین پس از 10 ثانیه از client 1 به server 1 نیز داده را به مدت 20 ثانیه به صورت TCP معمولی ارسال می کنیم. در زمان 20 ثانیه پس از شروع از client 2 نیز به server 2 داده را به مدت 20 ثانیه به صورت TCP معمولی ارسال می کنیم. سپس پهنای باند و میزان ترافیک انتقال یافته را یافته و نتایج را گزارش می کنیم.

**الگوریتم**

* **کنترل ازدحام (Congestion Control)**

به دلیل آنکه فرستادن صرفا افزایشی تعداد بسته ها موجب ازدحام شبکه و از دست رفتن بسته ها می شود لذا الگوریتم هایی پیاده سازی شده است که با توجه به وضعیت شبکه تعداد بسته های ارسالی را کم و زیاد می کند به این الگوریتم ها الگوریتم های کنترل ازدحام می گوییم.

تفاوت الگوریتم های کنترل ازدحام به دو بخش بستگی دارد یک اینکه با چه فرمولی افزایش پنجره ازدحام انجام می شود و دو اینکه زمانی که بسته ای از دست می رود که به صورت ضمنی نشاندهنده ازدحام در شبکه است پنجره ازدحام به چه صورتی کاهش می یابد. در شکل زیر این فرمول ها برای الگوریتم های LIA, OLIA, BALIA, wVegas, آمده است.



* **زمانبند (Scheduler)**

در حالت MPTCP هنگامی که می خواهیم بسته ای را ارسال کنیم باید انتخاب کنیم که از کدام زیرجریان ترافیکی آن را می خواهیم ارسال کنیم در نتیجه باید یه زمانبند داشته باشیم که طبق الگوریتم خاصی آن بسته را به زیر جریان مربوطه ارسال کند.

1. **الگوریتم LR (Lowest RTT first)**

در MPTCP الگوریتم پیشفرض است و به این صورت است که سهم بیشتر تعداد بسته های ارسالی جدید را به زیرجریانی که RTT یک بسته در آن کمتر از RTT بسته ای در زیرجریان دیگر باشد می دهد.

1. **الگوریتم Round Robin**

این الگوریتم به این صورت کار می کند که سهم تعداد بسته های جدید در هر زیرجریان برابر است.

* **مدیریت مسیر (Path Manager)**

مدیریت مسیر نیز در MPTCP فقط معنا پیدا می کند به این معنی که چه زمانی باید زیرجریان ترافیکی جدیدی اضافه کرد یا آن را حذف کرد. این کار و نحوه پیاده سازی آن توسط الگوریتم های مدیریت مسیر انجام می شود.

1. **الگوریتم Fullmesh**

در این الگوریتم به ازای هر زوج آدرس IP کلاینت و سرور یک زیر جریان ترافیکی ایجاد می شود.

1. **الگوریتم ndiffports**

در این الگوریتم به ازای یک زوج آدرس IP کلاینت و سرور بیش از یک زیر جریان ترافیکی (به صورت پیشفرض دو) ایجاد می شود. روش این کار این است که شماره پورت source را متفاوت قرار می دهد.

**نحوه اجرای پروژه**

ابتدا نرم افزار های [Vagrant](https://www.vagrantup.com/downloads) و [VirtualBox](https://www.virtualbox.org/wiki/Downloads) را دانلود و نصب کنید. سپس با دستور زیر پروژه sigcomm20\_mptp\_tutorial را clone کنید.

**git** clone https**://**github.com**/**qdeconinck**/**sigcomm20\_mptp\_tutorial.git

حال در ترمینال به پوشه sigcomm20\_mptp\_tutorial رفته و در آنجا دستور زیر را اجرا می کنیم.

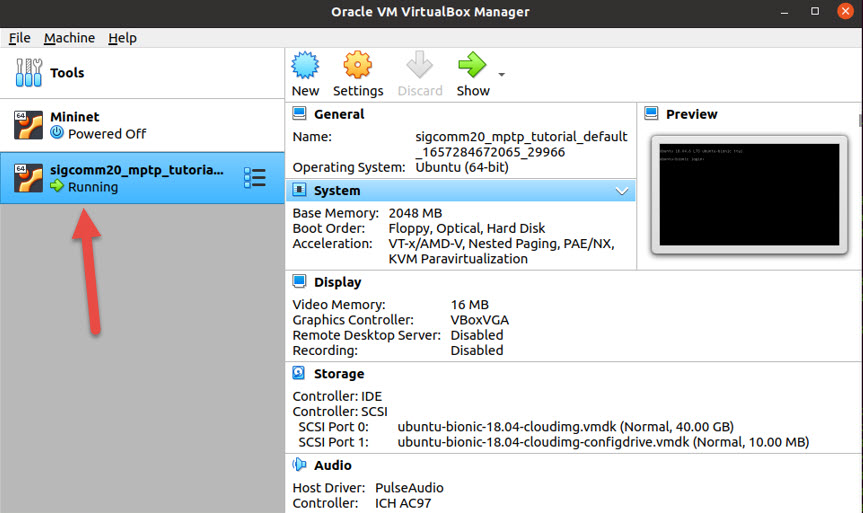
vagrant up

اگر این دستور کار نکرد می توانیم فایل اجرایی vagrant را در همان پوشه sigcomm20\_mptp\_tutorial قرار دهیم.

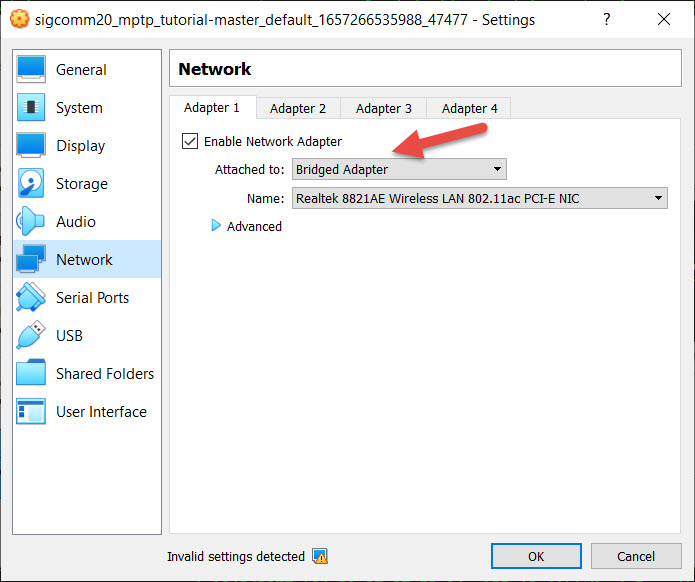
اگر سیستم عامل ویندوز را دارید قبل از دستور قبل باید دستور زیر را وارد کنید.

chcp 1252

در حالت ایده آل دستور vagrant up باید ابتدا سیستم عامل مجازی لینوکس با کرنل مجهز به MPTCP را ایجاد کرده و سپس Mininet، Minitopo، openflow و ... را ایجاد کرده و آماده به کار شود. اما در تجربه شخصی خود مشاهده کردیم که به ایراداتی بر می خورد لذا زمانی که مانند شکل زیر سیستم عامل مجازی در virtualBox ایجاد شد و دستور vagrant up تمام شد دیگر با vagrant کار نمی کنیم و از virtualBox سیستم عامل مجازی خود را کنترل می کنیم.



برای آنکه اتصال به اینترنت ماشین مجازی فعال شود باید ماشین مجازی خود را خاموش کرده و به قسمت تنظیمات آن رفته و در بخش network، آداپتور NAT را به Bridged Adaptor تغییر دهید.



سپس سیستم عامل مجازی خود را روشن می کنیم. نام کاربری و گذرواژه vagrant می باشد. سپس باید دستور زیر را بنویسیم.

**curl** www.multipath-tcp.org

اگر جواب yes را دریافت کردیم به این معنی است که ارتباط به صورت mptcp برقرار شده است و مشکلی در کرنل لینوکس مجهز به mptcp وجود نداشته است.



اما اگر به جواب no برخوردیم باید یا پروسه را از اول شروع کرده یا از روش های دیگر کرنل لینوکس مجهز به mptcp را نصب کنیم. این روش ها در سایت [multipath-tcp.org](http://multipath-tcp.org/) آمده است.



می توانیم دستورات زیر را درون سیستم عامل مجازی اجرا کرده و کرنل لینوکس مجهز به mptcp را نصب کنیم.

**wget** https**://**github.com**/**multipath-tcp**/**mptcp**/**releases**/**download**/**v0.94.7**/**linux-headers-4.14.146.mptcp\_20190924124242\_amd64.deb

**wget** https**://**github.com**/**multipath-tcp**/**mptcp**/**releases**/**download**/**v0.94.7**/**linux-image-4.14.146.mptcp\_20190924124242\_amd64.deb

**wget** https**://**github.com**/**multipath-tcp**/**mptcp**/**releases**/**download**/**v0.94.7**/**linux-libc-dev\_20190924124242\_amd64.deb

**wget** https**://**github.com**/**multipath-tcp**/**mptcp**/**releases**/**download**/**v0.94.7**/**linux-mptcp-4.14\_v0.94.7\_20190924124242\_all.deb

**sudo** dpkg **-**i linux-**\*.**deb

# The following runs the MPTCP kernel version 4.14.146 as the default one

**sudo** cat **/**etc**/**default**/**grub **|** **sed** -e "s/GRUB\_DEFAULT=0/GRUB\_DEFAULT='Advanced options for Ubuntu>Ubuntu, with Linux 4.14.146.mptcp'/" **>** tmp\_grub

**sudo** mv tmp\_grub **/**etc**/**default**/**grub

**sudo** update-grub

# Finally ask for MPTCP module loading at the loadtime

**echo** "

# Load MPTCP modules

sudo modprobe mptcp\_olia

sudo modprobe mptcp\_coupled

sudo modprobe mptcp\_balia

sudo modprobe mptcp\_wvegas

# Schedulers

sudo modprobe mptcp\_rr

sudo modprobe mptcp\_redundant

# The following line will likely not work with versions of MPTCP < 0.95

sudo modprobe mptcp\_blest

# Path managers

sudo modprobe mptcp\_ndiffports

sudo modprobe mptcp\_binder" **|** **sudo** tee -a **/**etc**/**bash.bashrc

حال باید mininet و minitopo را نصب کنیم. برای اینکار ابتدا source code ها را از git گرفته و هر کدام را نصب می کنیم.

**git** clone https**://**github.com**/**mininet**/**mininet

سپس دستورات زیر را وارد می کنیم.

**cd** mininet

**git** tag # list available versions

**git** checkout -b mininet-2.3.0 2**.**3**.**0 # or whatever version you wish to install

سپس دستور زیر را وارد می کنیم تا mininet نصب شود.

util**/**install.sh -a

تعداد پکیج هایی که دستور فوق نصب می کند زیاد است لذا مدت زمان قابل توجهی طول می کشد تا نصب شود.

اگر هنگام نصب به مشکلی خوردید چندبار دیگر هم دستور بالا را اجرا کنید. اگر هنگام clone کردن openflow به مشکل برخوردیم باید در فایل util/install.sh تغییراتی را ایجاد کنیم. در این فایل باید همه git:// ها را به https:// تغییر دهیم (5 مورد). برای مثال مانند زیر خط اول را به خط دوم تبدیل می کنیم.

**git** clone git**://**github.com**/**mininet**/**openflow

**git** clone https**://**github.com**/**mininet**/**openflow

سپس دستور util/install.sh را اجرا می کنیم.

حال دستور زیر را می نویسیم تا مطمئن شویم که mininet درست نصب شده است.

**sudo** mn

سپس exit را می نوسیم تا از محیط CLI مینینت خارج شویم.

حال به مسیر home/vagrant رفته و دستور زیر را اجرا می کنیم تا کد minitopo را دریافت کنیم.

**git** clone https**://**github.com**/**qdeconinck**/**minitopo.git

سپس دستور زیر را می نویسیم تا کد های این پروژه را نیز داشته باشیم.

**git** clone https**://**github.com**/**hamidreza7799**/**mptcp.git

برای تعیین حالت های مختلف الگوریتم های کنترل ازدحام و مدیریت مسیر و کنترل ازدحام در minitop لازم است که در فایل minitopo/experiments/iperf\_scenario.py از چهار دستور زیر استفاده کنیم که در حالت های مختلف بعضی از آن ها کامنت و بعضی از آن ها از کامنت خارج می شوند.

self**.**topo**.**command\_global**(**"modprobe mptcp\_rr && sysctl -w net.mptcp.mptcp\_scheduler=roundrobin"**)**

self**.**topo**.**command\_global**(**"modprobe mptcp\_ndiffports && sysctl -w net.mptcp.mptcp\_path\_manager=ndiffports "**)**

self**.**topo**.**command\_global**(**"echo 2 | sudo tee /sys/module/mptcp\_ndiffports/parameters/num\_subflows "**)**

برای مثال می خواهیم حالت LIA، Fullmesh، LR را اجرا کنیم. به پوشه lia\_fullmesh\_rtt می رویم و دستور های زیر را اجرا می کنیم.

**python3** configuration.py **-**m 1

**python3** configuration.py

دستور اول این خطوط را اضافه کرده و دستور دوم با توجه به پوشه ای که در آن اجرا می شود خطوط را کامنت می کند یا از کامنت خارج می سازد. (برای پوشه های بعدی فقط دستور دوم استفاده می شود)

حال دستور زیر را وارد می کنیم تا به اجرای minitopo ایرادی نگیرد.

**git** config **--**global **-**add safe.directory **/**home**/**vagrant**/**minitopo

برای اینکه بتوانیم هر یک از حالت ها را 50 بار آزمایش کنیم باید در هر پوشه دستور زیر را اجرا کنیم.

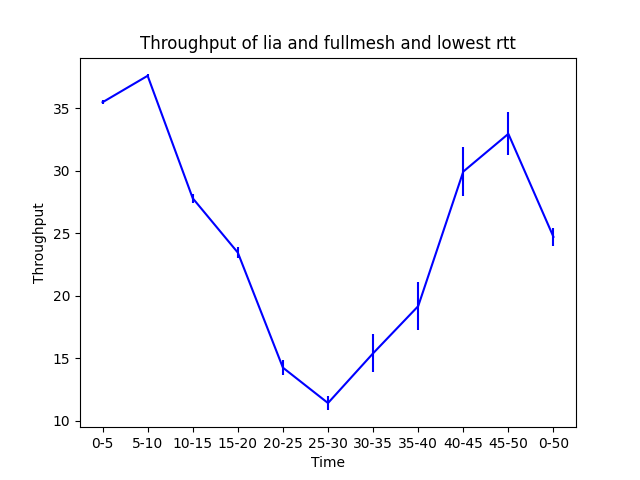
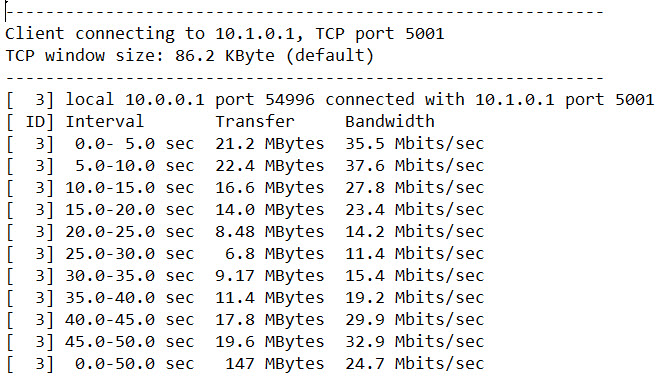
./run.sh

این دستور باعث می شود که iperf کلاینت mptcp در هر بار اجرا ذخیره گردد. در run.sh از minitopo برای شبیه سازی استفاده می شود که فایلی را به عنوان توپولوژی و فایل دیگری را به عنوان experiment دریافت می کند. فایل experiment نیز به iperf\_scenario.py اشاره می کند و برای تغییر در توپولوژی و سناریو باید این فایل را تغییر بدهیم.

**آزمایش**

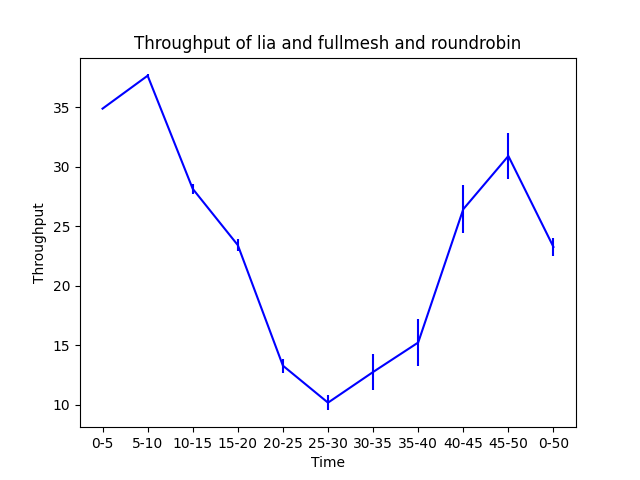
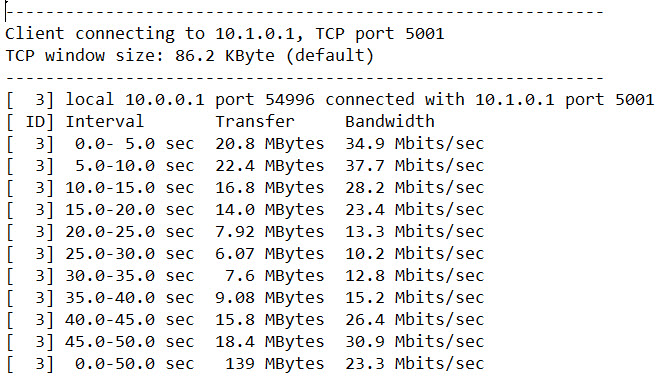
حالت 1 : کنترل ازدحام : LIA، مدیریت مسیر : Fullmesh، زمانبند : LR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



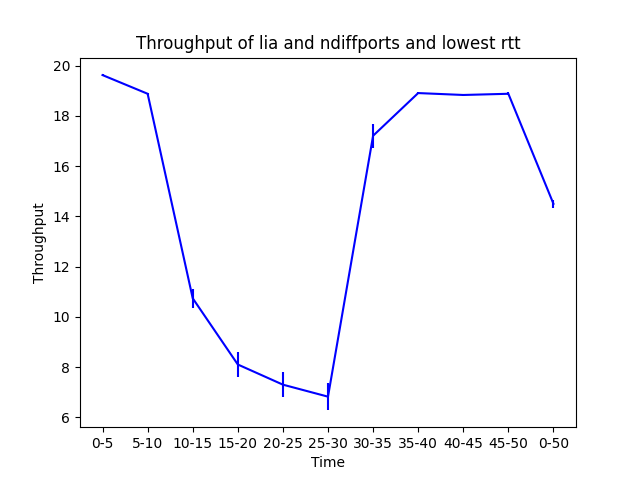
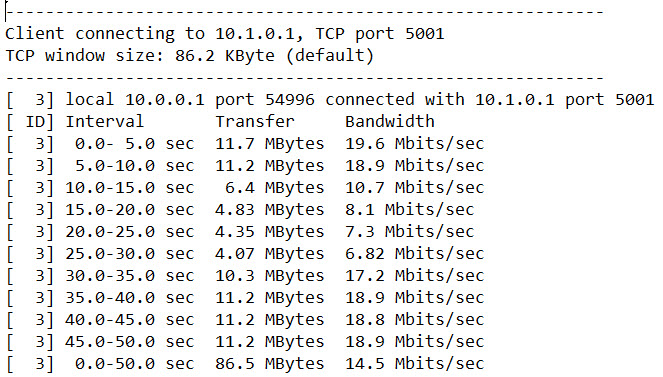
حالت 2 : کنترل ازدحام : LIA، مدیریت مسیر : Fullmesh، زمانبند : RR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



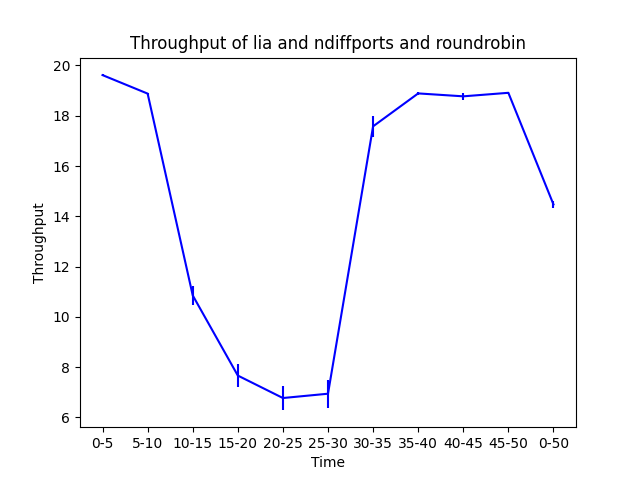
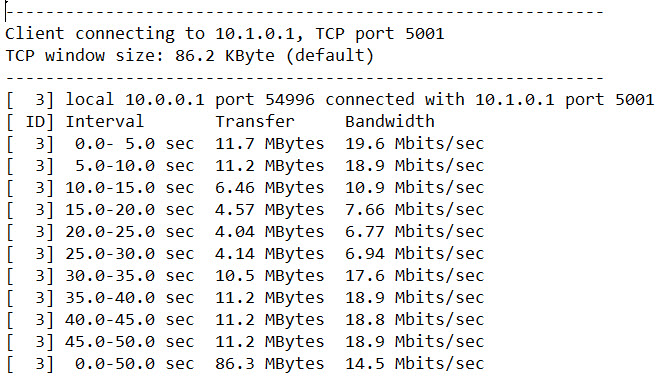
حالت 3 : کنترل ازدحام : LIA، مدیریت مسیر : ndiffports، زمانبند : LR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



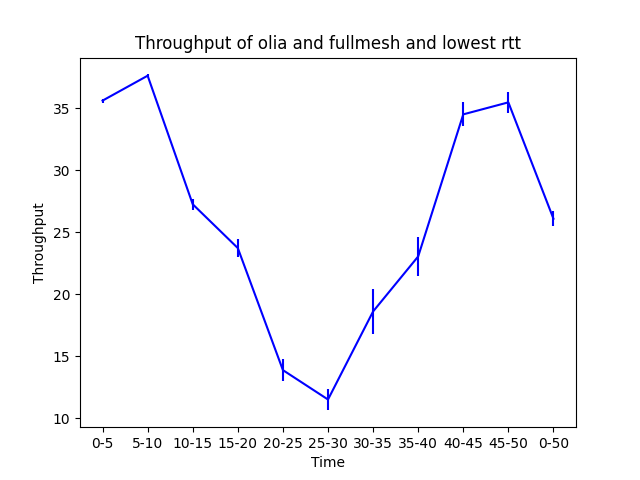
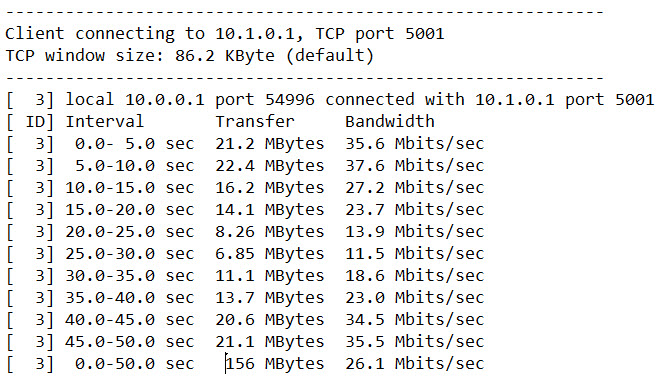
حالت 4 : کنترل ازدحام : LIA، مدیریت مسیر : ndiffports، زمانبند : RR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



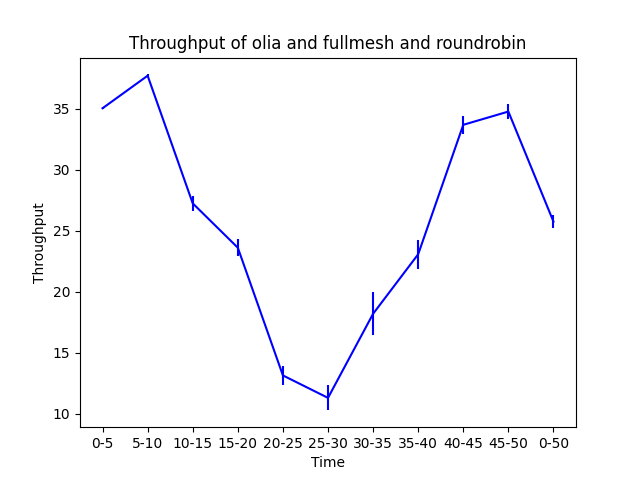
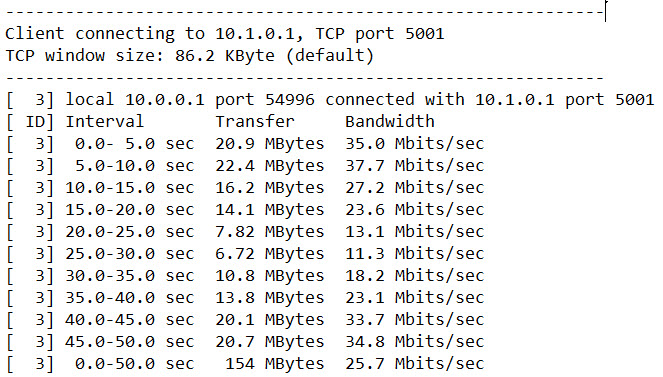
حالت 5 : کنترل ازدحام : OLIA، مدیریت مسیر : Fullmesh، زمانبند : LR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



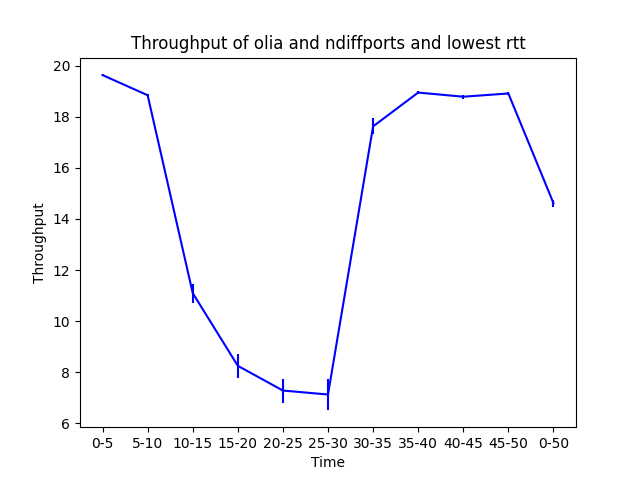
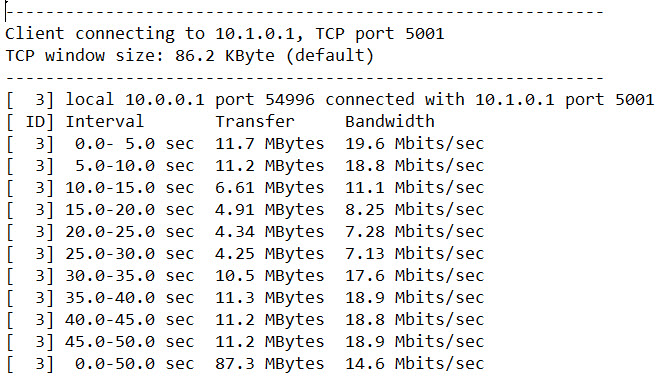
حالت 6 : کنترل ازدحام : OLIA، مدیریت مسیر : Fullmesh، زمانبند : RR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



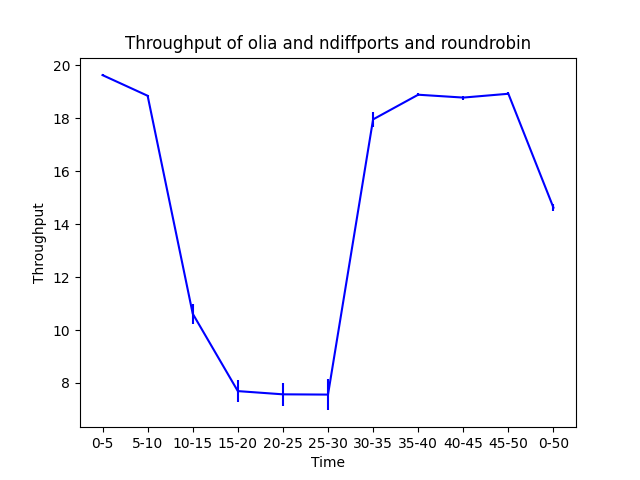
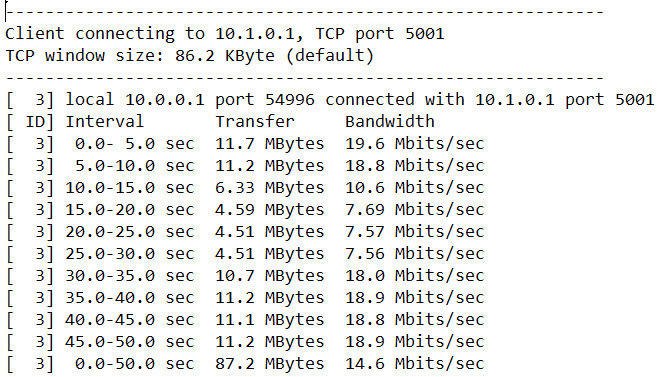
حالت 7 : کنترل ازدحام : OLIA، مدیریت مسیر : ndiffports، زمانبند : LR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



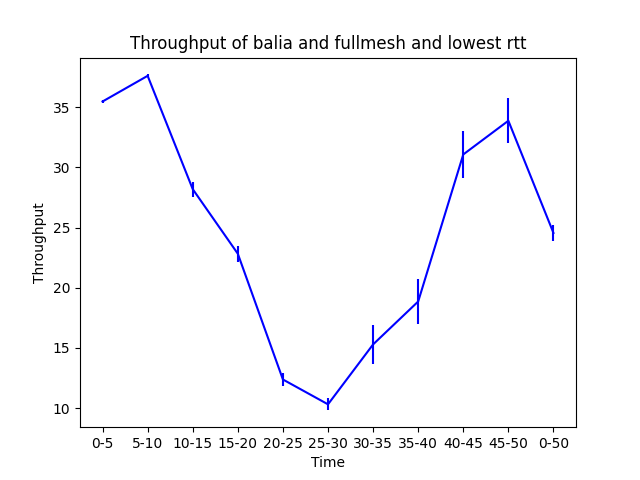
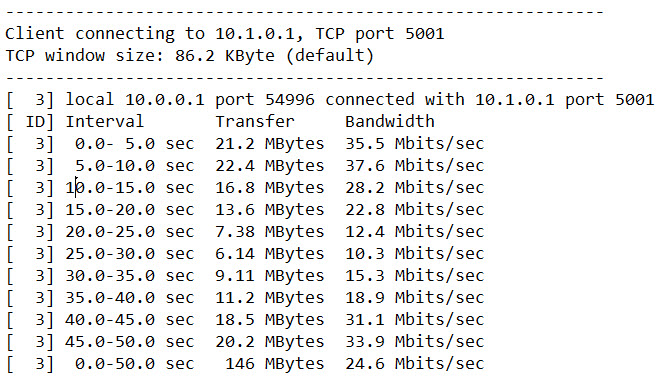
حالت 8 : کنترل ازدحام : OLIA، مدیریت مسیر : ndiffports، زمانبند : RR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



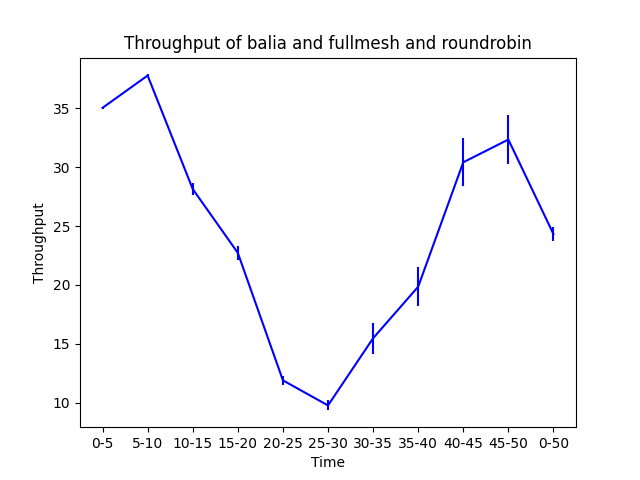
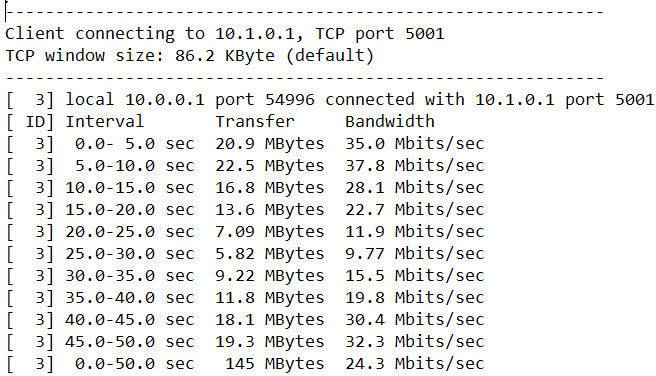
حالت 9 : کنترل ازدحام : BALIA، مدیریت مسیر : Fullmesh، زمانبند : LR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



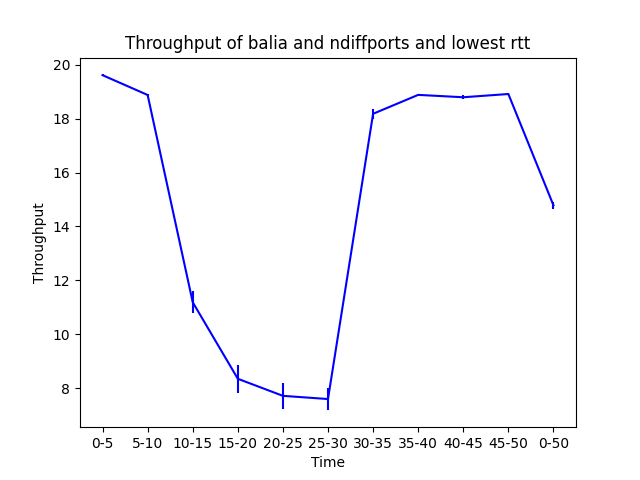
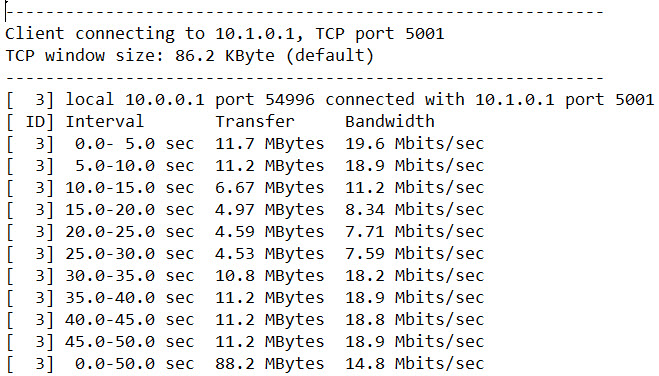
حالت 10 : کنترل ازدحام : BALIA، مدیریت مسیر : Fullmesh، زمانبند : RR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



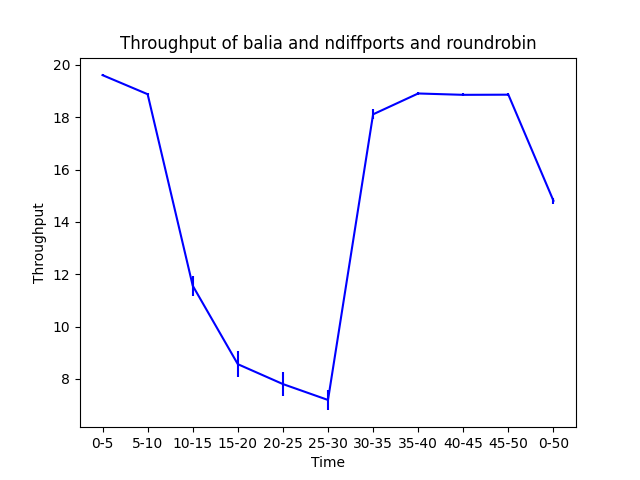
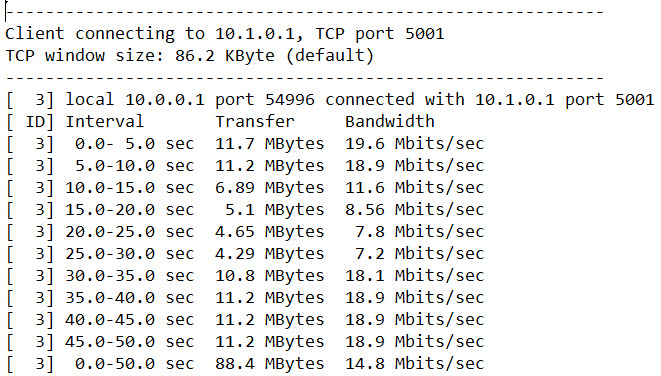
حالت 11 : کنترل ازدحام : BALIA، مدیریت مسیر : ndiffports، زمانبند : LR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



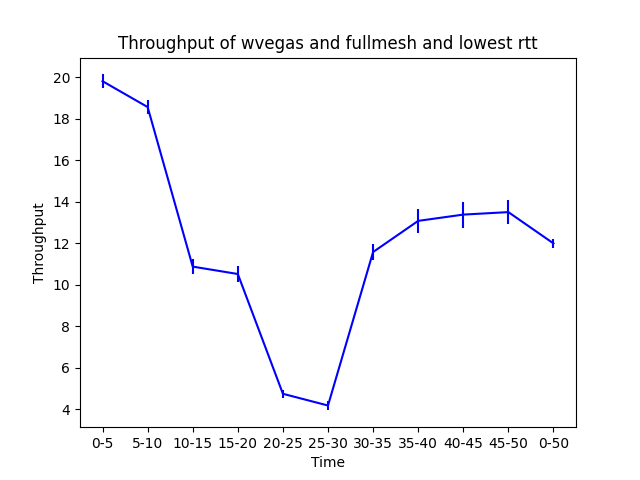
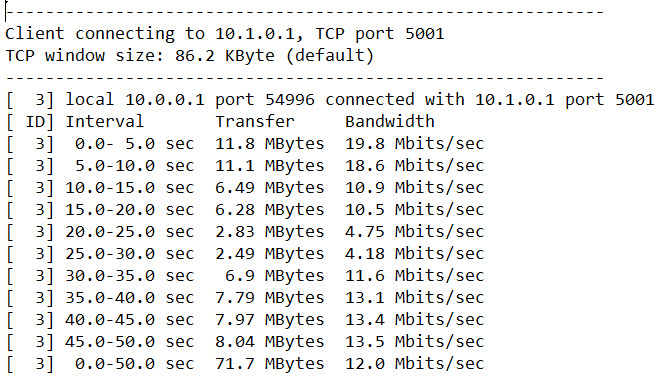
حالت 12 : کنترل ازدحام : BALIA، مدیریت مسیر : ndiffports، زمانبند : RR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



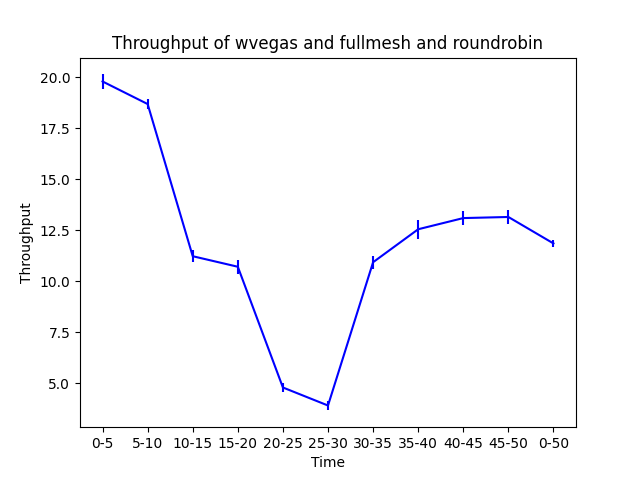
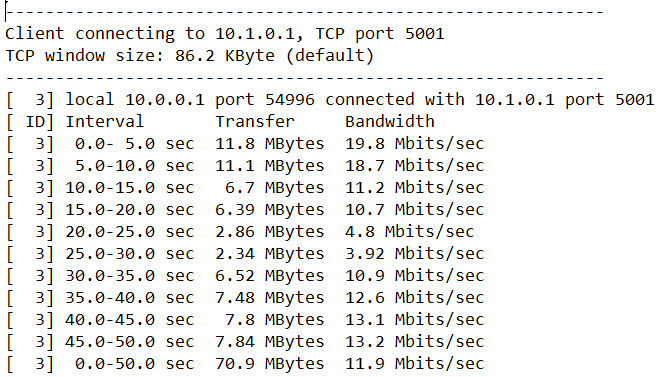
حالت 13 : کنترل ازدحام : WVEGAS، مدیریت مسیر : Fullmesh، زمانبند : LR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



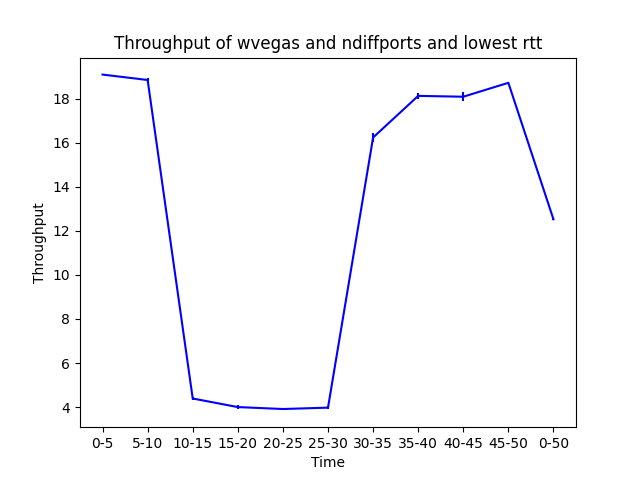
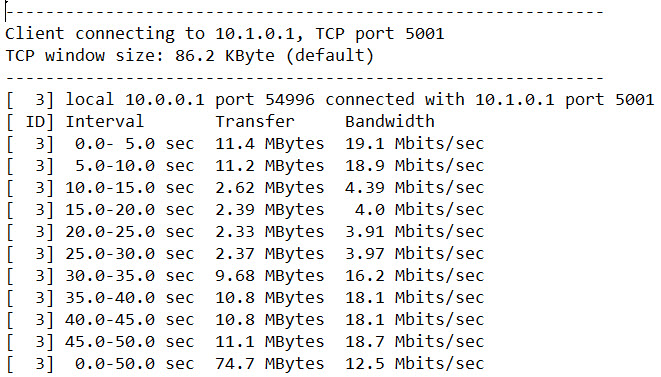
حالت 14 : کنترل ازدحام : WVEGAS، مدیریت مسیر : Fullmesh، زمانبند : RR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



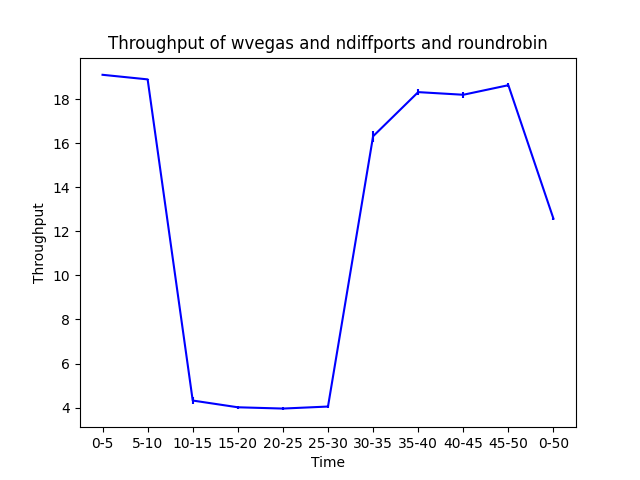
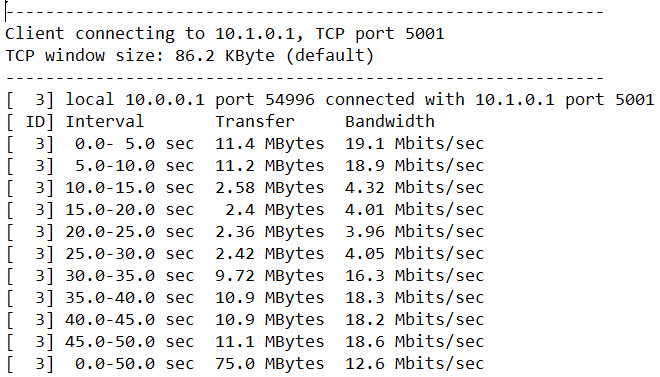
حالت 15 : کنترل ازدحام : WVEGAS، مدیریت مسیر : ndiffports، زمانبند : LR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



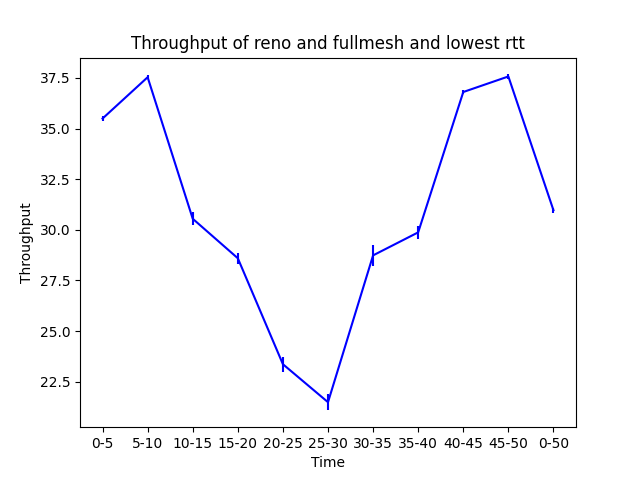
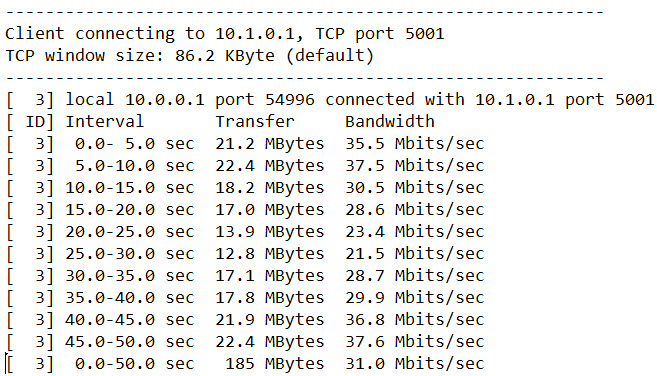
حالت 16 : کنترل ازدحام : WVEGAS، مدیریت مسیر : ndiffports، زمانبند : RR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



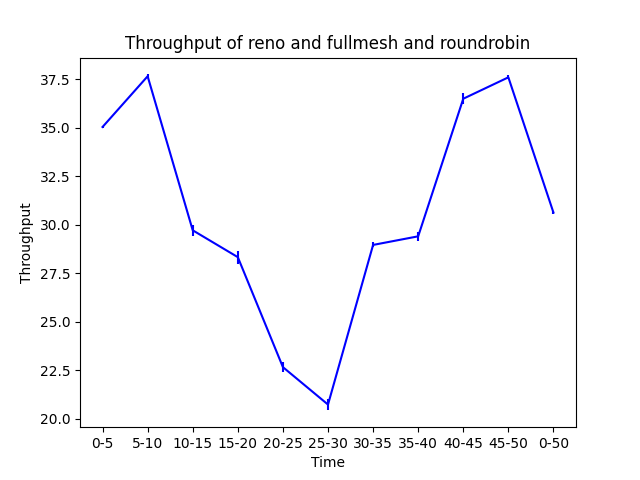
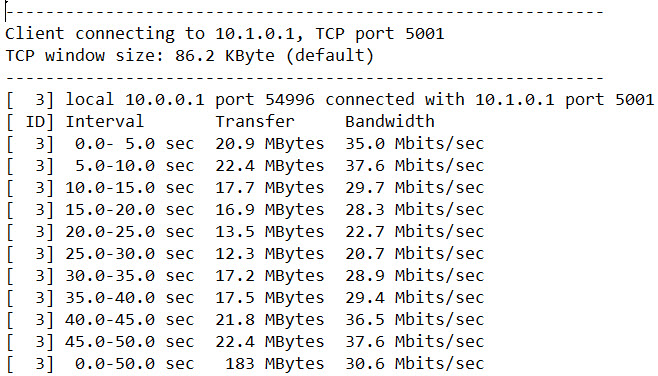
حالت 17 : کنترل ازدحام : RENO، مدیریت مسیر : Fullmesh، زمانبند : LR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



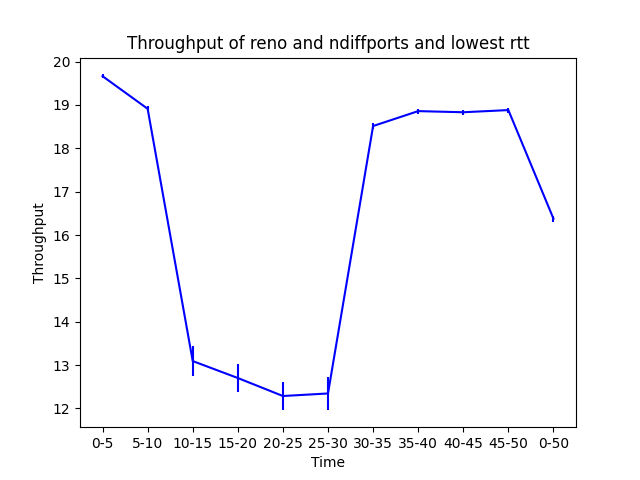
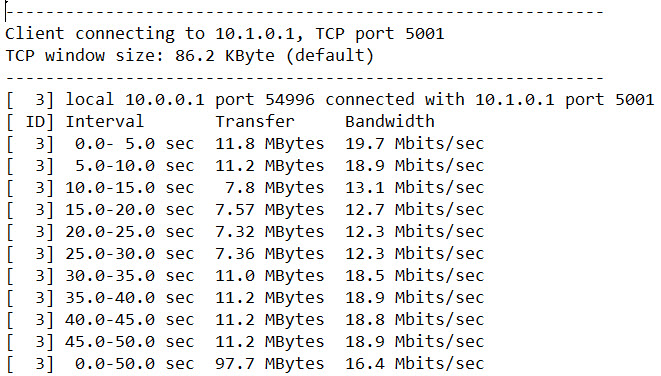
حالت 18 : کنترل ازدحام : RENO، مدیریت مسیر : Fullmesh، زمانبند : RR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



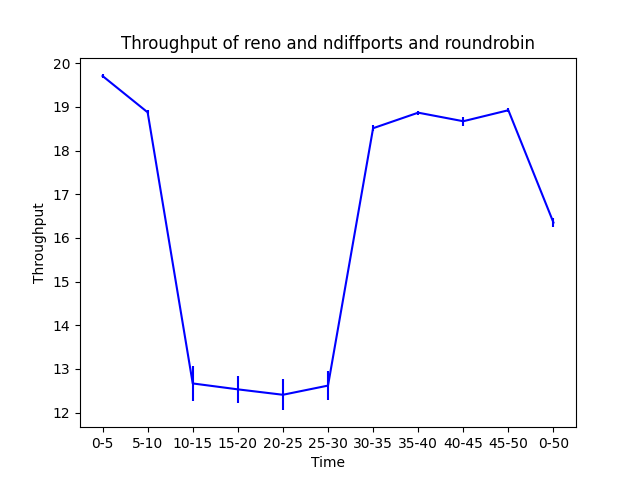
حالت 19 : کنترل ازدحام : RENO، مدیریت مسیر : ndiffports، زمانبند : LR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند



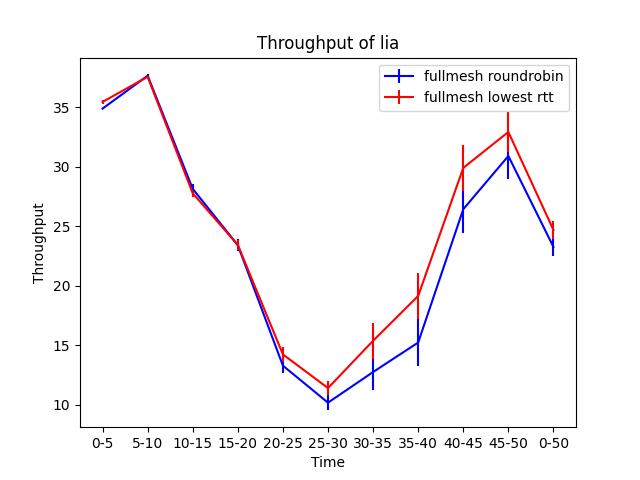
حالت 20 : کنترل ازدحام : RENO، مدیریت مسیر : ndiffports، زمانبند : RR

اعداد تصویر زیر میانگین 50 بار تکرار آزمایش را نشان می دهند

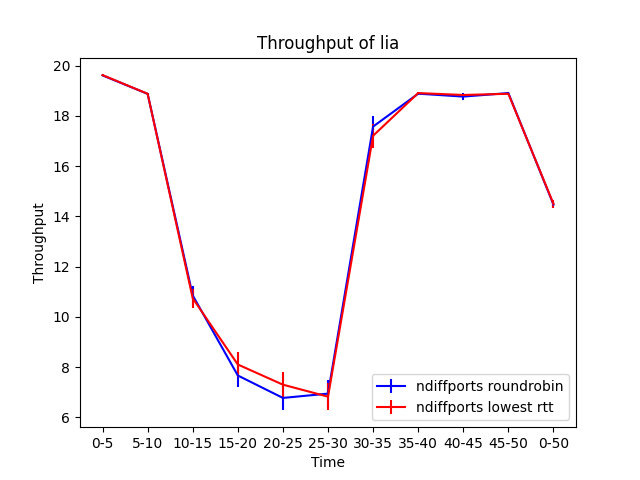


حال نمودار های حالت های قبلی را با هم ترکیب کرده تا تفاوت الگوریتم ها با یکدیگر مشخص شود.

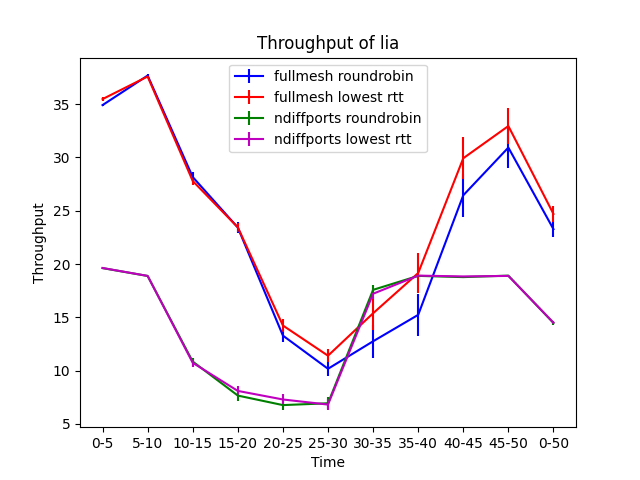
الگوریتم LIA و Fullmesh



الگوریتم LIA و ndiffports



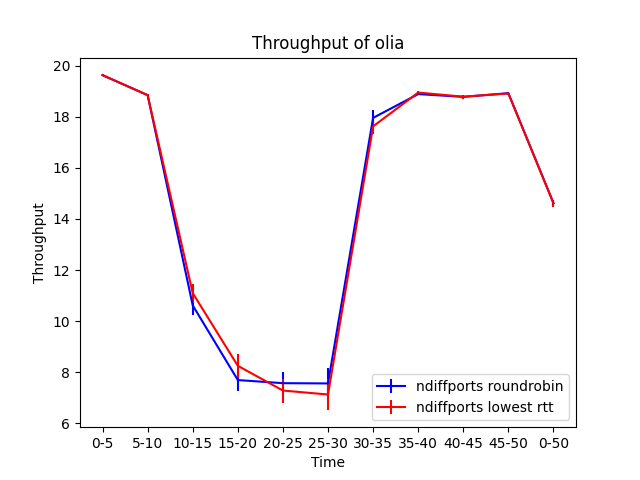
الگوریتم LIA



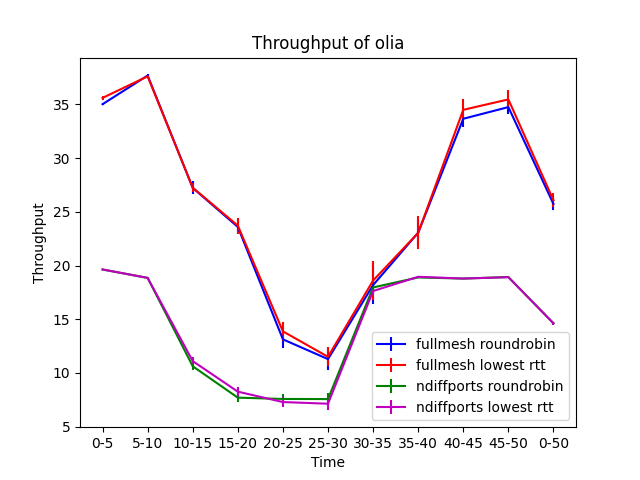
الگوریتم OLIA و Fullmesh



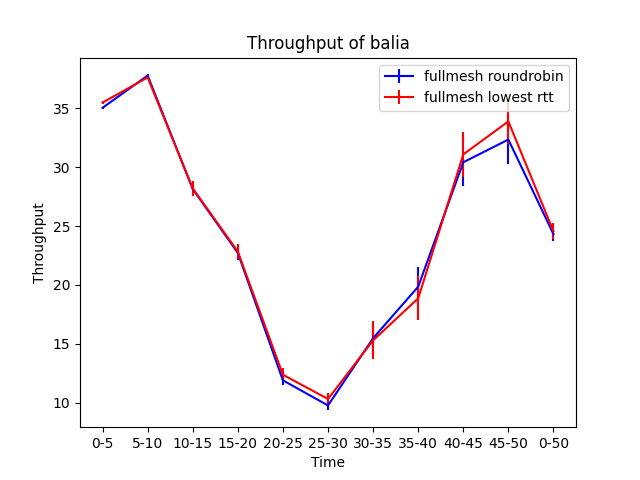
الگوریتم OLIA و ndiffports



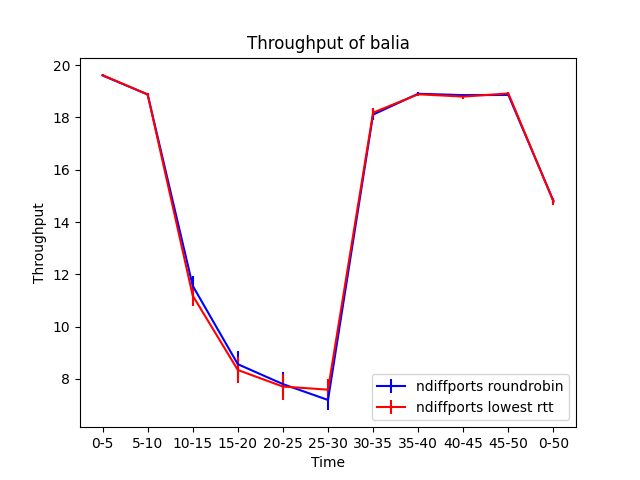
الگوریتم OLIA



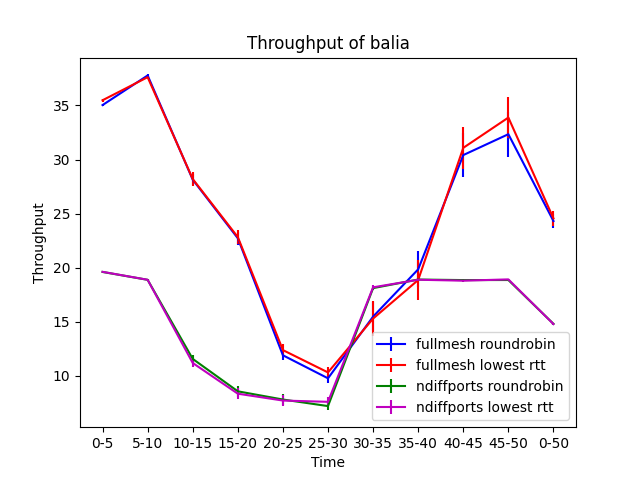
الگوریتم BALIA و Fullmesh



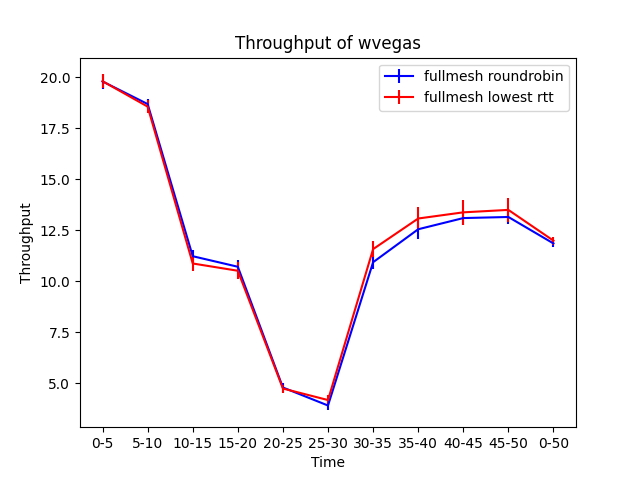
الگوریتم BALIA و ndiffports



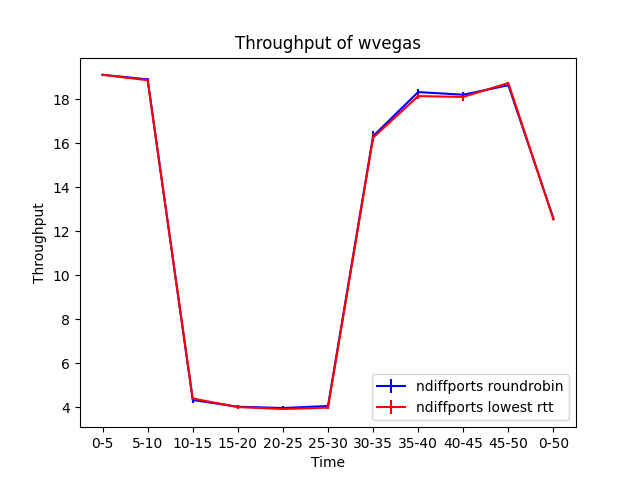
الگوریتم BALIA



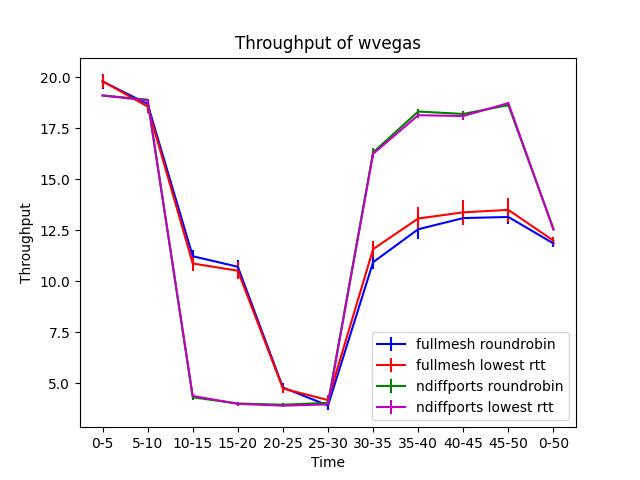
الگوریتم WVEGAS و Fullmesh



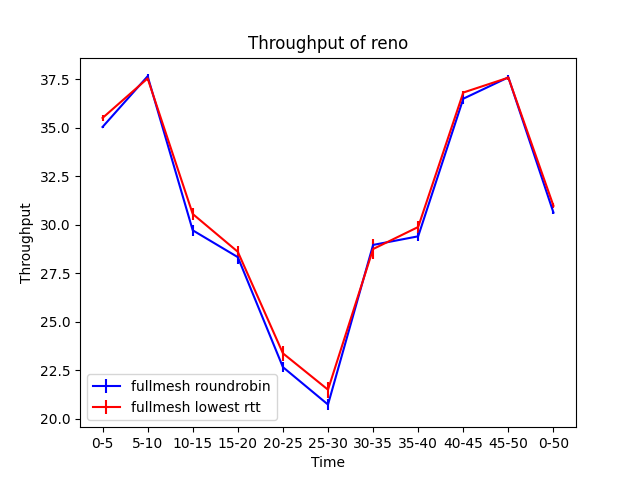
الگوریتم WVEGAS و ndiffports



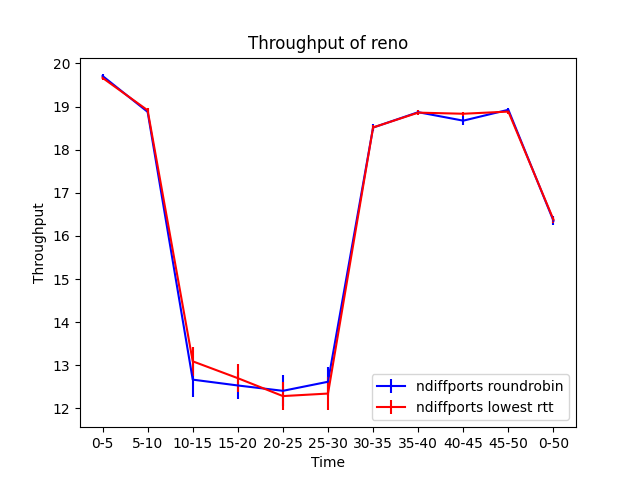
الگوریتم WVEGAS



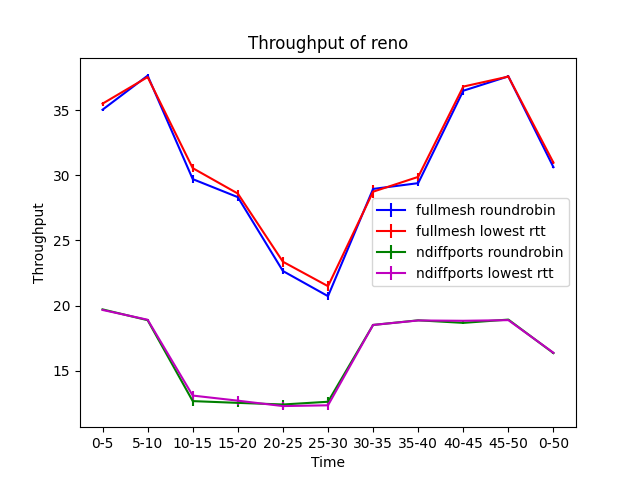
الگوریتم RENO و Fullmesh



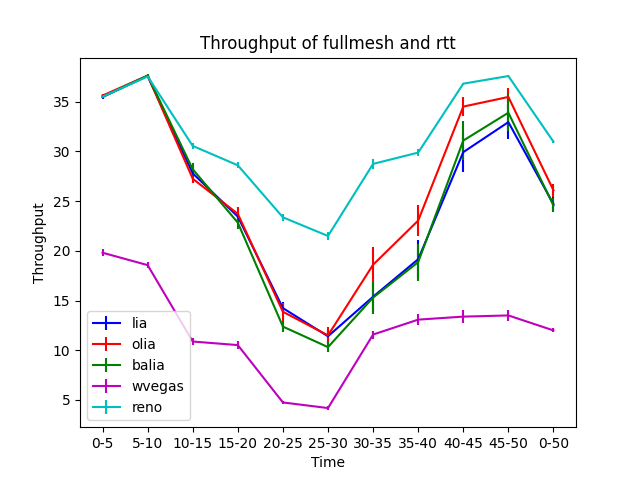
الگوریتم RENO و ndiffports



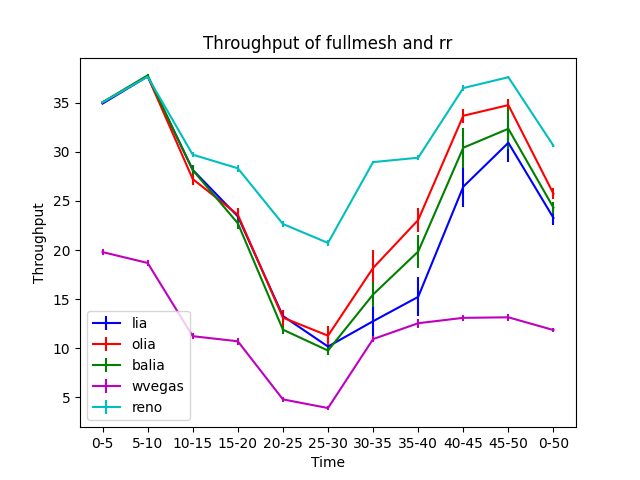
الگوریتم RENO



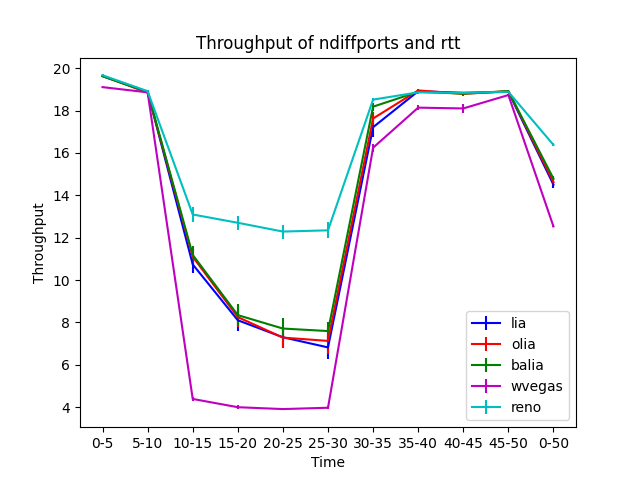
الگوریتم Fullmesh و LR



الگوریتم Fullmesh و RR



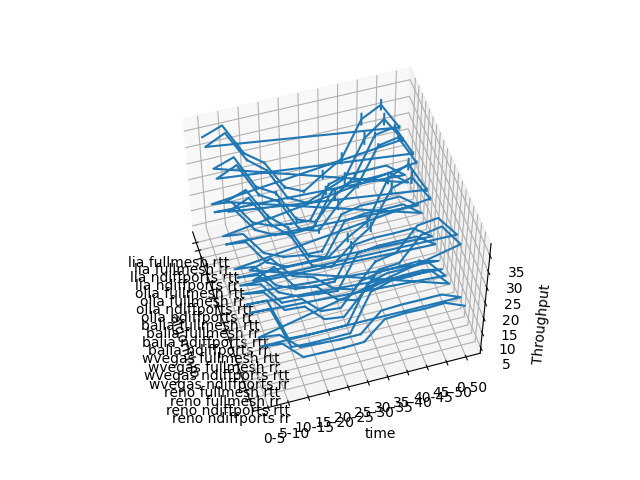
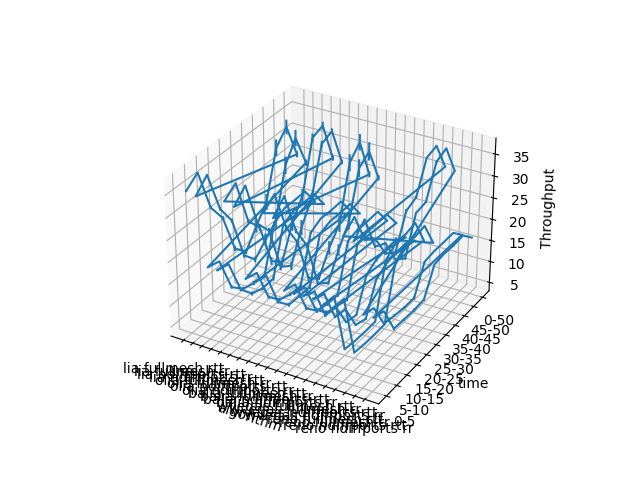
الگوریتم ndiffports و LR



الگوریتم ndiffports و RR



در پایان اگر همه نمودارها را بخواهیم به صورت سه بعدی نشان دهیم شکل های زیر ایجاد می شود.



**نتیجه گیری**

* در حالتی که الگوریتم LIA را ثابت می کنیم و الگوریتم های زمانبند و مدیریت مسیر را تغییر می دهیم نتیجه می گیریم که به طور کلی الگوریتم Fullmesh بهتر از ndiffports عمل کرده است زمانی که از Fullmesh استفاده می کنیم از دو لینک بالا و پایین در mptcp استفاده می شود و زمانی که هر دو client ها از لینکی استفاده نمی کنند حداکثر پهنای باند می تواند تا 40mbps برسد و زمانی که یکی از client ها فعال است حداکثر پهنای باند می تواند تا 30mbps برسد و زمانی که هر دو فعال باشند تا 20mbps می رسد. در ndiffports فقط از لینک بالا استفاده می شود لذا بیشترین پهنای باندی که می تواند حاصل شود 20mbps است که در حالتی که client 1 شروع به ارسال داده می کند حداکثر پهنای باند به 10mbps می رسد.
* در حالتی که الگوریتم LIA را ثابت کرده و الگوریتم Fullmesh را نیز ثابت می کنیم از مقایسه الگوریتم های زمانبند نتیجه می گیریم که الگوریتم LR و RR از شروع فرآیند تا ثانیه 25 یکسان عمل کرده اند و از ثانیه 30 تا 40 با شیب تندتری الگوریتم LR نسبت به RR رشد کرده است. در مابقی حالات نیز شیب افزایش و کاهش یکسان است.
* در حالتی که الگوریتم OLIA و ndiffports را ثابت کنیم می توانیم نتیجه بگیریم که هنگام کاهش پهنای باند دریافتی الگوریتم RR سعی کرده است که از یک مینیمم پایین تر نیاید و از خود مقاومت نشان می دهد اما الگوریتم LR کاملا به صورت نزولی کاهش یافته است.
* در حالتی که فقط الگوریتم OLIA را ثابت فرض کنیم می توانیم مشاهده کنیم که از ثانیه 30 تا 35 هر دو الگوریتم های Fullmesh و ndiffports برخلاف حالت LIA به پهنای باند نزدیک به همی رسیده اند.
* هنگامی که الگوریتم Fullmesh و LR را ثابت فرض کنیم می بینیم که الگوریتم RENO از همه بهتر کار می کند چون پهنای باند لینک به صورت مساوی بین client ها تقسیم نمی شود و mptcp client سهم بیشتری را می برد. الگوریتم wvegas از همه الگوریتم های موجود بدتر عمل می کند چرا که به صورت محتاطانه ای پنجره ازدحام را افزایش می دهد و به نظر می آید که اصلاً از لینک پایینی استفاده نمی کند. بین الگوریتم های coupled در این حالت OLIA بهتر عمل کرده چرا که فرمول افزایش پنجره ازدحام آن سریعتر افزایش می یابد. الگوریتم های LIA و BALIA عملکرد یکسانی دارند.
* در حالتی که الگوریتم های Fullmesh و RR را ثابت فرض کنیم مانند قبل است به جز اینکه الگوریتم BALIA بهتر از LIA عمل می کند.
* در حالتی که الگوریتم های ndiffports و LR را ثابت می کنیم می بینیم که RENO از همه بهتر و wvegas از همه بدتر است. الگوریتم های coupled عملکرد تقریبا یکسانی را دارند. در این حالت چون الگوریتم ndiffports استفاده می شود الگوریتم های کنترل ازدحام به غیر از ثانیه 10 تا 30 عملکرد تقریبا یکسانی دارند چون از یک لینک استفاده می کنند.
* در حالتی که الگوریتم های ndiffports و RR را ثابت می کنیم نتایج مانند حالت قبلی است.