بسم الله الرحمن الرحیم

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

محمد مهدی سوری

حمیدرضا آذرباد

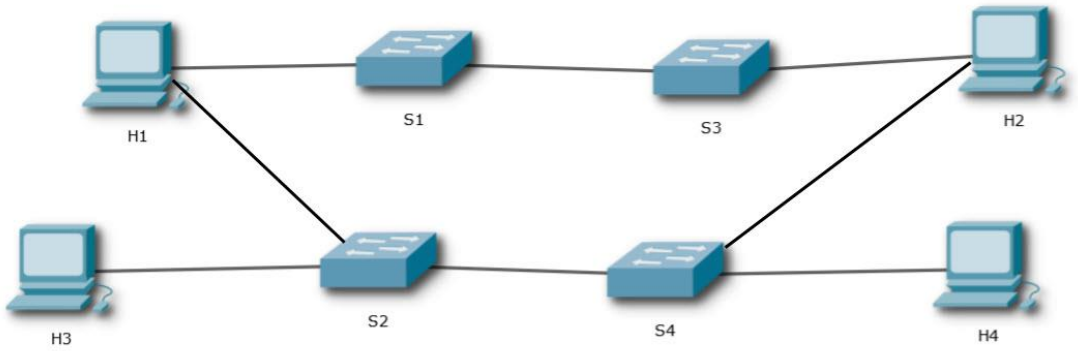
مقدمه

در این پروژه قصد داریم که با شبیه سازی TCP چند مسیره (MPTCP) درون محیط شبیه ساز Mininet، 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 چهار هاست را ایجاد می کنیم سپس چهار سوئیچ را نیز ایجاد می کنیم. هاست شماره یک (h1) را به سوئیچ های شماره یک (s1) و دو (s2) متصل می کنیم. هاست شماره دو (h2) را نیز به سوئیچ های شماره سه (s3) و چهار (s4) متصل می کنیم. بین هاست شماره یک (h1) و دو (h2) می خواهیم ارتباط MPTCP را برقرار کنیم. هاست سه (h3) را به سوئیچ شماره دو (s2) و هاست شماره چهار (h4) را نیز به سوئیچ شماره چهار (s4) متصل می کنیم. بین هاست شماره سه (h3) و چهار (h4) می خواهیم ارتباط TCP معمولی را برقرار کنیم. بین سوئیچ های شماره یک (s1) و سه (s3) یک لینک و بین سوئیچ های شماره دو (s2) و چهار (s4) نیز یک لینک برقرار می کنیم. شکل زیر این ارتباطات را نشان می دهد.



به غیر از لینک های بین دو سوئیچ همه لینک ها سرعت و تاخیر یکسان به اندازه 100Mbps و 0.1ms را دارند. سرعت لینک بین سوئیچ یک (s1) و سه (s3) 10Mbps و تاخیر آن 20ms است و سرعت لینک بین سوئیچ دو (s2) و چهار (s4) 20Mbps است و تاخیر آن را طی آزمایش تغییر می دهیم.

علت سریع بودن لینک های بیرون از دو سوئیچ این است که bottleneck شبکه را لینک های بین دو سوئیچ تشکیل دهند.

سناریو

ابتدا از h1 به h2 به مدت 50 ثانیه داده ارسال می کنیم. در همین حین پس از 10 ثانیه از h3 به h4 نیز 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. (برای افزایش دقت هر آزمایش را 20 بار تکرار نموده و سپس تاخیر لینک بین دو سوئیچ دو و چهار را تغییر داده و نتایج را با نمودار های دارای error bar به نمایش می گذاریم)

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 wVegas, BALIA, OLIA, LIA, آمده است.

TABLE I
SUMMARY OF MULTIPATH-COUPLED CONGESTION CONTROLS DEPLOYED IN THE MPTCP LINUX KERNEL IMPLEMENTATIONS.

	Alpha Parameter	Congestion Window Increase	Congestion Window Decrease
LIA:	$\alpha = w_{\text{total}} \dfrac{\max\left(w_r/\tau_r^2\right)}{\left[\sum_{k\in\mathcal{R}}(w_k/\tau_k)\right]^2}$	For each ACK received on subflow r, $w_r = w_r + \min\left(\dfrac{\alpha}{w_{\text{total}}}, \dfrac{1}{w_r}\right)$	For each packet loss on subflow r, $w_r = w_r - \dfrac{w_r}{2}$, as in standard TCP [18]
OLIA:	$\alpha_r = \begin{cases} \frac{1/\lvert\mathcal{P}\rvert}{\lvert\mathcal{C}\rvert}, & \text{if } r \in \mathcal{C}, \\ \frac{1/\lvert\mathcal{P}\rvert}{\lvert\mathcal{W}\rvert}, & \text{if } r \in \mathcal{W}, \lvert\mathcal{C}\rvert > 0, \\ 0, & \text{otherwise} \end{cases}$	For each ACK received on subflow r, $w_r = w_r + \left(\dfrac{w_r/\tau_r^2}{\left[\sum_{k\in\mathcal{R}}(w_k/\tau_k)\right]^2} + \dfrac{\alpha_r}{w_r}\right)$	For each packet loss on subflow r, $w_r = w_r - \dfrac{w_r}{2}$, as in standard TCP [18]
BALIA:	$\alpha_r = \dfrac{\max\{x\}}{x_r}$, where $x_r = \dfrac{w_r}{\tau_r}$, and $x = \left[\sum_{k\in\mathcal{P}}(w_k/\tau_k)\right]^2$	For each ACK received on subflow r, $w_r = w_r + \left[\dfrac{x_r}{\tau_r\ x_k}\left(\dfrac{1+\alpha_r}{2}\right)\left(\dfrac{4+\alpha_r}{5}\right)\right]$	For each packet loss on subflow r, $w_r = w_r - \left[\dfrac{w_r}{2}\ \min\left(\alpha_r, \dfrac{3}{2}\right)\right]$
wVegas:	**if** $(\delta_r > \alpha_r)$, **then** $\alpha_r = \omega_r\ \alpha_{\text{total}}$, where $\delta_r = \left(\dfrac{w_r}{\hat{\tau}_r} - \dfrac{w_r}{\bar{\tau}_r}\right)\hat{\tau}_r$, $\omega_r = \dfrac{x_r}{\sum_{k\in\mathcal{R}} x_k}$, and $x_i = \dfrac{w_i}{\bar{\tau}_i}$	For each transmission round on subflow r, $w_r = \begin{cases} w_r - 1, & \text{if } \delta_r > \alpha_r, \\ w_r + 1, & \text{if } \delta_r < \alpha_r \end{cases}$ **if** $(q_r > \bar{\tau}_r - \hat{\tau}_r)$, **then** $q_r = \bar{\tau}_r - \hat{\tau}_r$ **if** $(\bar{\tau}_r - \hat{\tau}_r \geq 2q_r)$, **then** $w_r = w_r\ \dfrac{\hat{\tau}_r}{2\bar{\tau}_r}$	For each packet loss on subflow r, $w_r = w_r - \dfrac{w_r}{2}$, as in standard TCP [18]

- **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.

2. 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**

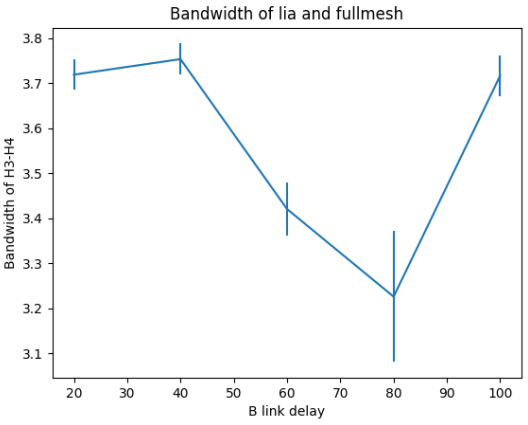
در این الگوریتم به ازای هر زوج آدرس IP کلاینت و سرور یک زیر جریان ترافیکی ایجاد می شود.

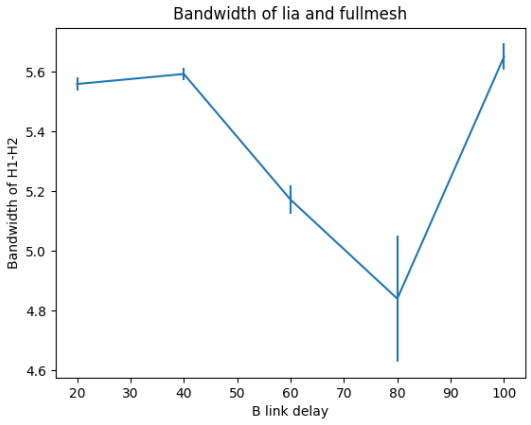
2. **الگوریتم ndiffports**

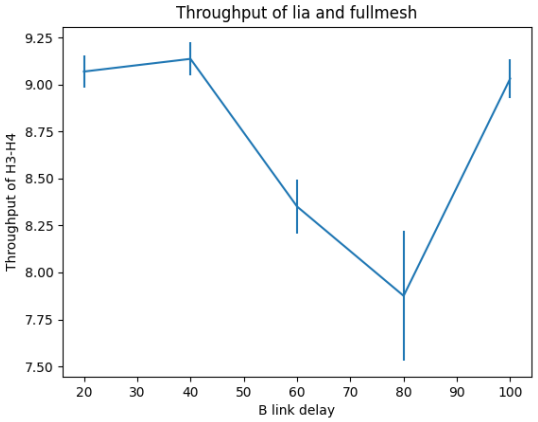
در این الگوریتم به ازای یک زوج آدرس IP کلاینت و سرور بیش از یک زیر جریان ترافیکی (به صورت پیشفرض دو) ایجاد می شود. روش این کار این است که شماره پورت source را متفاوت قرار می دهد.

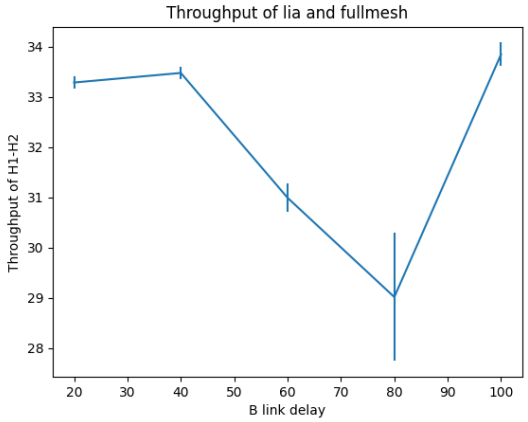
آزمایش

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

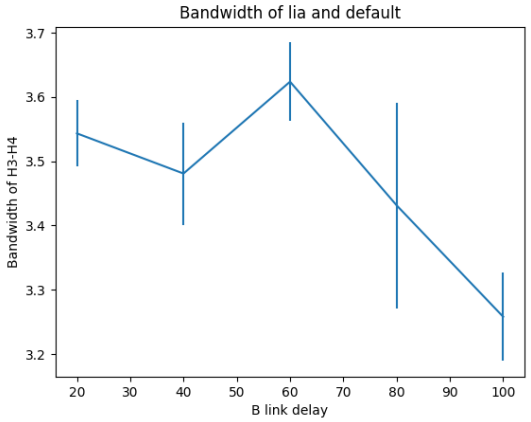


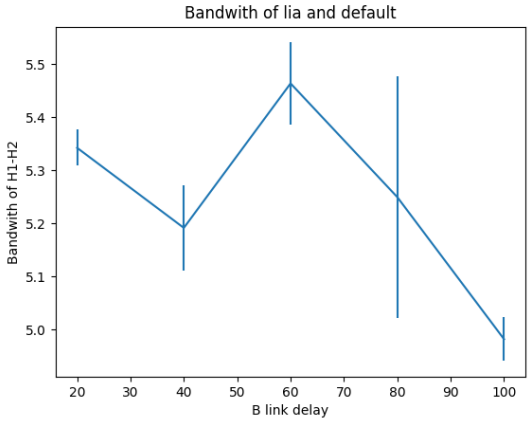


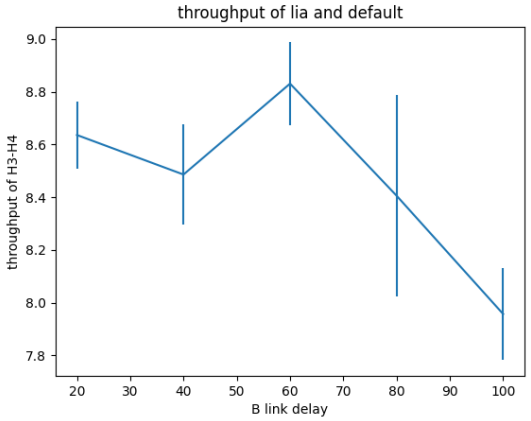


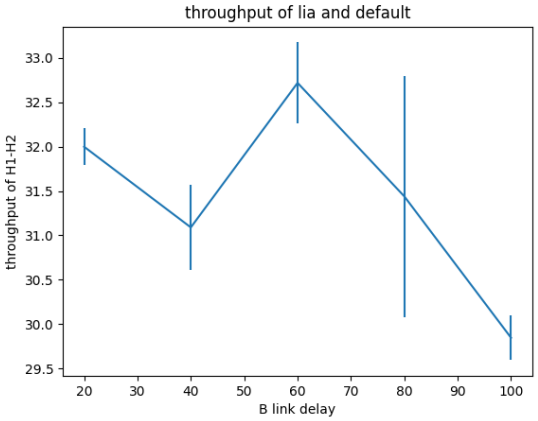


حالت 2 : کنترل ازدحام : LIA، مدیریت مسیر : default، زمانبند : LR









جزئیات نتایج در فایل های پیوست آورده شده است.