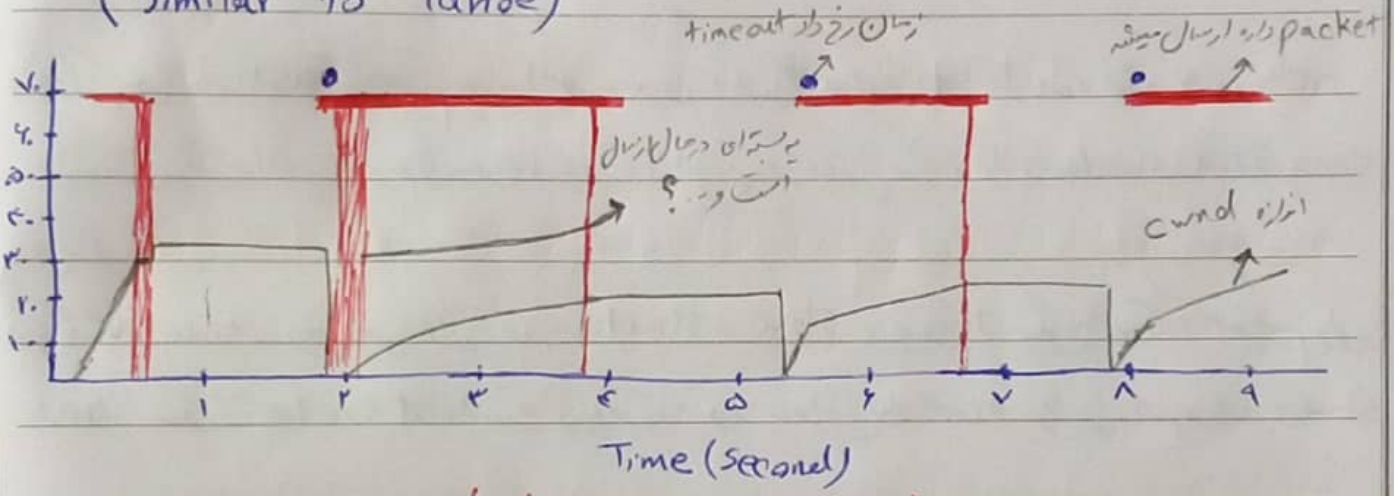
Year. Month. Date. ( )

حلسم ۱۹ : * congestion در لایه transport در کنار Top است اما در جهای دیگر هم انجام می‌شود (در Applications در کنار UDP)

Example: a trace of Top congestion control mechanism (similar to Tahoe)



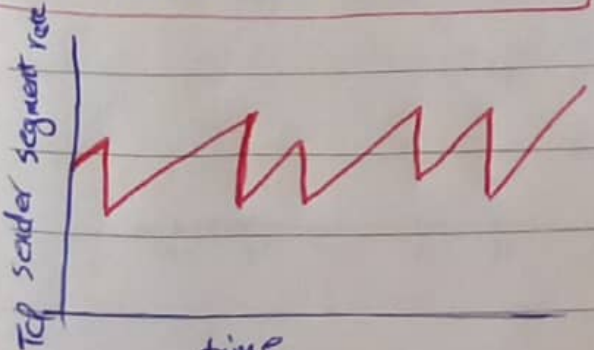
"رویکرد اصلاحی های بعدی را نگاه کن:

approach: senders can increase sending rate until packet loss (congestion) occurs, then decrease sending rate on loss event.

Additive Increase
increase sending rate by 1 maximum segment size every RTT until loss detected

Multiplicative Decrease
cut sending rate in half at each loss event

: AIMD



AIMD sawtooth behavior: probing for bandwidth

AIMD براساس آزمایش و خطا طراحی شده است.

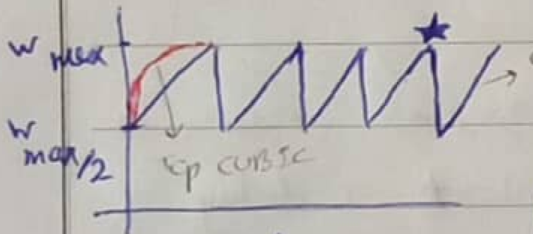
۳ خاصیت مهم AIMD: به صورت distribute کار می‌کند ، اگر یکسری Device وجود داشته باشند که همه به یک شکل رفتار کنند شبکه stable خواهد بود و مشکلی پیش نمی‌آید.

۴) مشکل stability (پایداری) هم ندارد و overflow ندارد، به نوعی بهینه هم هست.

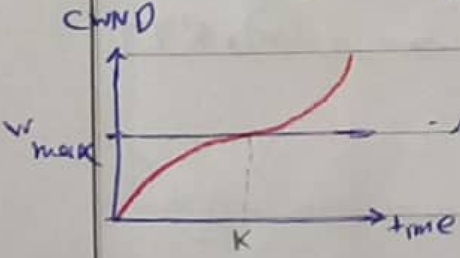
TCP CUBIC :

* is there a better way than AIMD to "probe" for usable bandwidth?

مطابق زیر شکل ، کل دیتای ارسالی در شبکه



* ما می‌دانیم که سر قله‌ها (★) به مشکل برخورد می‌کنیم و می‌توانیم تصمیم بگیریم که دیتاها را سریع‌تر ارسال کنیم (با ارسال نمایی). و وقتی نزدیک ★ می‌شود، شیب کم می‌شود یعنی probe می‌کند، ببیند Congestion رخ می‌دهد یا نه؟ و با احتیاط بیشتری کار می‌کند.

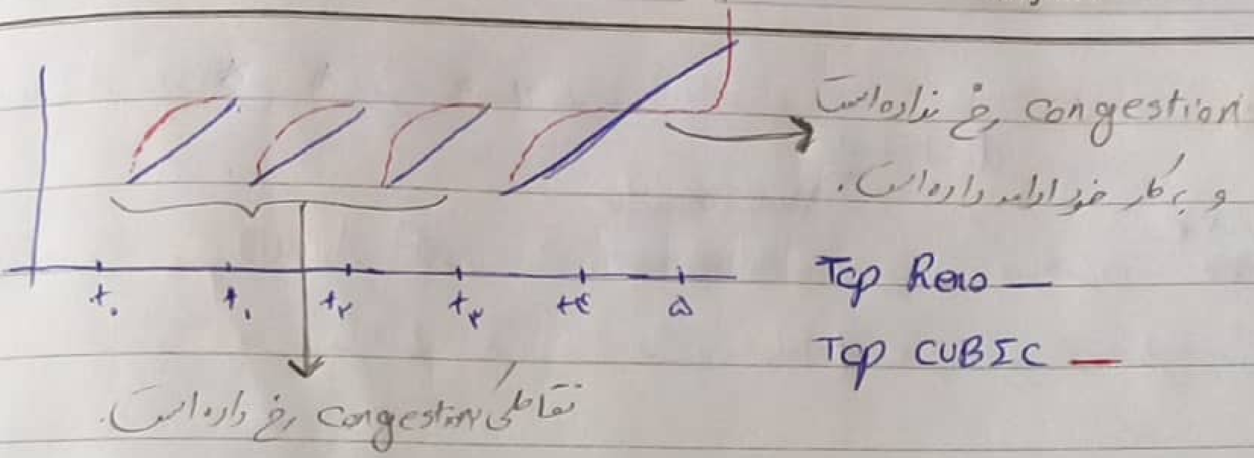


* در CUBIC رابطه window با زمان به صورت درجه ۳ می‌باشد. یعنی با احتیاط به w_{max} می‌رسد و اگر مشکلی نبود، ادامه می‌دهد.

$$CWND(t) = C \times (t - K)^3 + w_{max}$$

هرچه فاصله بیشتر باشد رشد سریع انجام می‌شود و هرچه فاصله کمتر باشد (تا نقطه K) : با احتیاط بیشتر و کندتر حرکت می‌کند.

Year. Month. Date. ( ) Subject :



* Tcp CUBIC به صورت default در linux و windows10 استفاده می‌شود.

*مهم: Bandwidth_Delay product (BDP) *

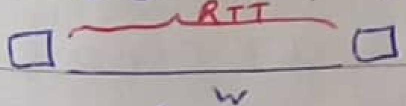
یکی از انگیزه‌های Tcp CUBIC ← BDP بوده است.

- معیار BDP چیست؟ _what does the value of BDP tell us?

وقتی روی یک شبکه می‌خواهیم اطلاعاتی را ارسال کنیم وقتی بحث performance بود، (delay ، lost ، throughput) معیارهای اصلی بودند.

حال اگر فرض کنیم لینک شبکه مورد استفاده ما، پهنای باند و RTT مشخصی دارد. اگر پهنای باند لینک را بین دو نقطه W در نظر بگیریم:

W x RTT (W: پهنای باند لینک) در مبحث Tcp تعداد packet ها در پهنای باند را نشان می‌دهد.



یکی از شبکه‌ها وجود دارند که W x RTT (W: پهنای باند لینک) مقدار بزرگی برایشان محسوب می‌شود. (بخصوص به خاطر بزرگ بودن RTT که delay های زیادی متحمل می‌شود)

چالشی که وجود دارد: اگر congestion window را کوچک در نظر بگیریم و خیلی صبر کنیم فرض می‌شود بسته‌ها از دست رفته‌اند و از resource ها هم استفاده‌ای نشده است!

Year. Month. Date. ( ) Subject :

- اما ما می خواهیم از resource ها استفاده کنیم و سریعاً اعلام congestion کنیم!

مثلاً می خواهیم تعداد packet را در شبکه push کنیم و سریعاً هم نگوییم time out رخ داد.

این موردی است که عملاً در آن Tcp CUBIC رخ داده است، و این شبکه ها

Long-Fat Network (LFN) گفته می شود. شبکه هایی که BDP بزرگی دارد.

12.5 KB

وقتی که دیدند نرخ ها و فاصله ها بیشتر شد، روشی به نام CUBIC طراحی کردند تا از resource

بیشتری استفاده کند.

این معیار بسیار مهمی است (BDP) که در بسیاری از طراحی ها استفاده می شود.

و در بسیاری از طراحی congestion control هم BDP مورد استفاده است. یعنی به نحوی

سایز Buffer را با توجه به RTT مشخص می کند. و این معیاری است برای طراحی های

پیشرفته تر در advance Tcp.

ارسال اطلاعات

- CUBIC در دهه اول به صورت حریصانه (aggressive) افزایش را انجام

می دهد وقتی به congestion نزدیک شد، از سطح حریصانه خود کم می کند.

TALASH

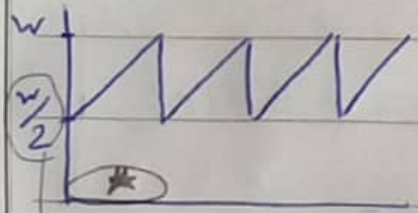
- CUBIC یک Fairness را رعایت می‌کند.

- This allows CUBIC to behave fairly when competing with short-RTT flows, which will have Acks arriving more frequently.

در شبکه‌ای که RTT خیلی بزرگی دارد ممکن است timeout رخ دهد و اینکه دیگر چیزی نفرستد، اما در فاز short-RTT این کار را جبران می‌کند چون timeout دیرتر رخ می‌دهد و duplicate رخ می‌دهد. و بیشتر در حال ارسال short-RTT flow است. که CUBIC این توانایی را ایجاد می‌کند که هر زمانی که RTT بزرگ است و زمان رفت و برگشت طولانی است با نرخ بالاتری رفت و برگشت را انجام دهد و از resource بیشتری استفاده کند چون احتمال وجود timeout بیشتر است بنابراین دیگه داره اطلاعات بیشتری ارسال می‌گیریم.

- avg. TCP Reno throughput as function of window size, RTT?

فرض می‌کنیم در STD state (؟) هستیم و slow start هم وجود ندارد و به صورت AIMD ، run می‌شود.



- چقدر اطلاعات در واحد زمان ارسال می‌شود؟

$$\text{Throughput} = \frac{\frac{W}{2}}{RTT} \text{ و } \frac{W}{RTT}$$

$$\text{Throughput} = \frac{3}{4}\frac{W}{RTT} \frac{\text{byte}}{\text{sec}}$$

(*) ممکن است در این بازه چندین RTT وجود داشته باشد.

Year. Month. Date. ( ) Subject :

Tcp splitting, optimizing The performance of cloud services

Tcp splitting: مثلاً وقتی می‌خواهیم به یک سرور وصل شویم و سرچ انجام دهیم،

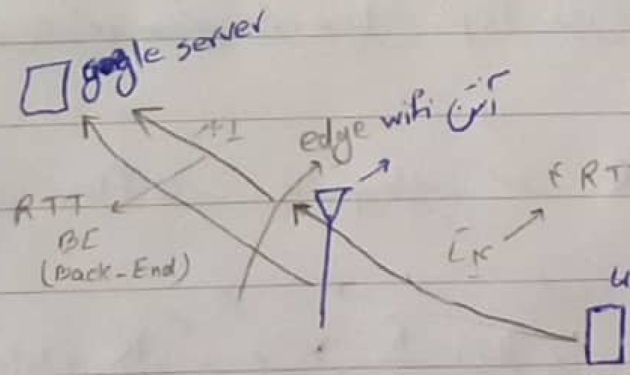
برای این کار، به access point در دانشگاه وصل می‌شویم، و بعد access point با سرور ارتباط برقرار می‌کند

برای این کار ۳ بار RTT نیاز است و ابتدای این کار هم باید Tcp connection باز شود که جمعاً

۴ تا RTT مورد نیاز است. که ممکن است زمان زیادی نیاز باشد چون سرورها

در ۱۹ مورد قبلی وجود دارند. کاری که می‌توان انجام داد این است که connection ها را

split کنیم (بشکنیم).



user google server, user یعنی اینکه ۴ RTT بین user و google server تکرار شود. بهتر است

user و wifi ← ۴ بار RTT انجام شود و هر وقت به صورت مجزا Tcp connection بود

با ایجاد نقطه میانی ← Edge | نقطه نقطه Edge |

بعد از Edge تا google server ← Tcp connection مجزای انجام می‌شود

که گوگل رخ می‌دهد

$$\text{delay کل} = RTT_{(BE)} + 4\,RTT_{(FE)} + \text{processing time}$$

* اگر این کار انجام نمی‌شد ۴ $RTT_{(BE)}$ داشتیم.

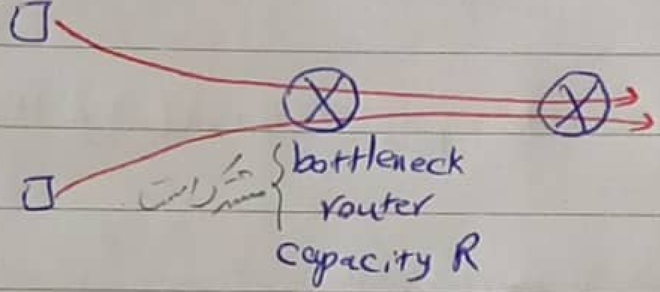
TALASH

Year. Month. Date. ( ) Subject :

ACAMS و google این کار را انجام می‌دهند و bink سعی می‌کند انجام دهد ولی به نتیجه‌ای نمی‌رسد!

* TCP Fairness:

- Fairness goal: چیزی که flow ها به صورت عادلانه از شبکه استفاده می‌کنند.

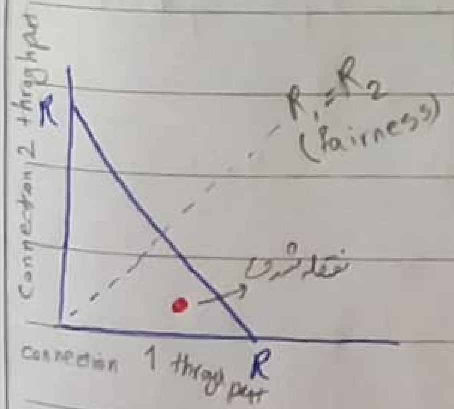


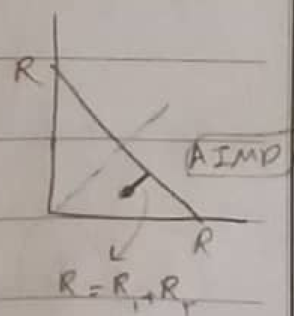
+ به طور میانگین $\frac{R}{K}$ به همه TCP ها تخصیص داده می‌شود.

* برای تحلیل TCP فرض می‌کنیم در فاز AIMD هستیم:

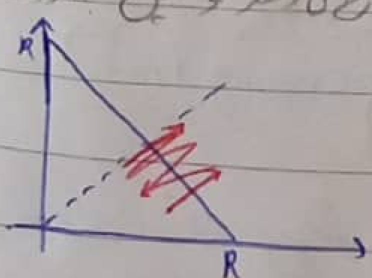
اگر ۲ تا flow داشته باشیم و ظرفیت لینک R می‌باشد،

یکی به اندازه $\frac{R}{2}$ و یکی کمتر از $\frac{R}{2}$ ارسال می‌کند.





وقتی ۲ • \ به مسیر congestion رخ می‌دهد و سپس به صورت خطی



کاهش پیدا می‌کند B این کار را انجام می‌دهد تا

به $R_1 = R_2$ برسد.

TCP Reno ← به صورت Fair عمل می‌کند ۱۷ اسلاید بعدی را ببین ۵