

جلسه ۲۰ و ۲۱ مجازی

جلسه ۲۲ :

Source-Based Approaches (vegas, BBR, DCTcp) → BBR

BBR براساس vegas (Bottleneck Bandwidth and RTT) اصلاح شده است

BBR توسط حقیقین کوکل طراحی شده است. وابستگی بین data sender ها

(LH) توسط BBR انجام می شود برای مدیریت congestion ها

و طبق concept vegas، اینگونه عمل می کند / براساس <sup>میان</sup> RTT تقسیم بندی می کند

اما BBR طراحی دقیق تر دارد و ~~بیشتر~~ علاوه بر RTT بیشتر بر <sup>میان</sup> Bottleneck Bandwidth

تأثیر دارند و براساس آن نرخ خود را تنظیم می کند.

\* ۳ فاز BBR :

① packet-pacing ② Bandwidth-probing ③ RTT-probing

① packet-pacing : براساس Bottleneck Bandwidth نرخ خود را تنظیم می کند

و براساس آن ارسال را انجام می دهد.

③ **RTT-probing**: مشابه Vegas که Base RTT وجود دارد برای همین

هر بار بعد از RTT و probe می‌بینیم که اگر چه بعد از این مدت که در انتظار هستیم و مقداری نزدیک

Base RTT (بر اساس آن expected Rate را بدست آورده ایم) وجود نداشته باشد،

بنابراین به صاف برای RTT بدست می‌آوریم تا آید به سرعت شود.

\* **BBR** به روشی است داخل linux به عنوان الگوریتم قرار گرفته است و توضیح

نموده به سبب ندارد.

\* چگونه Bottleneck را بدست آوریم؟ تا جایی که ممکن است packet ارسال

می‌کنند و اگر به مشکل برخورد می‌شود، Bottleneck bandwidth را می‌بینیم

② **Bandwidth-probing**: مربوط به نرخ است.

+ **packet-pacing** spaces the packets based on the estimate of the available bandwidth.

- this eliminates bursts and unnecessary queuing, which results in a better feedback signal.

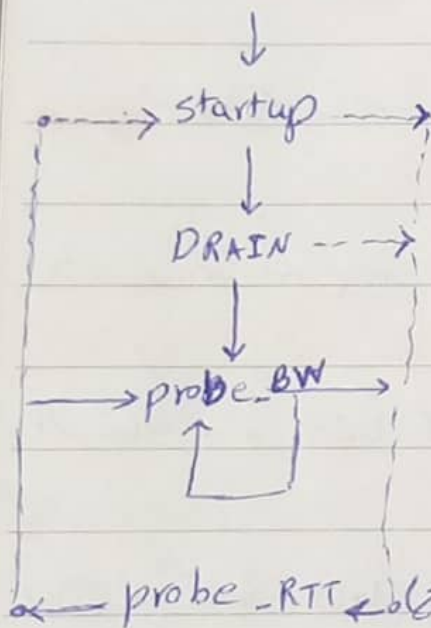


+ BBR also periodically increases its rate, thereby probing the available bandwidth.

+ similarly, BBR periodically decreases its rate, thereby probing for a new minimum RTT.

\* BBR دو فاز دارد :

① فازهای startup و DRAIN است.



اولین کاری که می‌شود وارد فاز startup می‌شود و به دنبال

Bottleneck band width می‌گردد و به دست می‌آورد.

سپس وارد فاز DRAIN می‌شود: که در اینجا به دنبال پیدا کردن (پیدا کردن) probe-RTT است.

بعد از آنکه به دست می‌آید و سپس ارسال را انجام می‌دهد (مدیریت Q ها را می‌کنند) تنظیم می‌کند.

و بعد از این که شروع به کار کرد وارد فاز STD یا دائمی می‌شود که همان probe-BW <sup>band width</sup>

است. به طور مثال probe کردن را ادامه می‌دهد و براساس آن نرخ را تنظیم می‌کند.

در فاز probe-RTT BaseRTT را به دست می‌آورد.

**Startup:** در اینجا می‌خواهد Bottleneck را بدست بیاورد و سری Data می‌فرستد و ~~مکث~~ Ack ما را می‌سپرد می‌کند و بعد از آن زمانی تصمیم می‌گیرد که این مقدار و فاصله

"windowed max-filter" گفته می‌شود.

**DRAIN:** اول نرخ را تنظیم می‌کند / Bottleneck بعد match شود

**★ چالش ۱:** در مقایسه BBR و CUBIC : BBR استفاده بهتری از resource

دارد و حتی ممکن است CUBIC به نسبت ۱۰۰ تا کمتر از BBR از Bandwidth

استفاده کند. BBR در محدوده bandwidth می‌نویسد و می‌تواند CUBIC

است استفاده است. lost کرده باشد و نرخ bandwidth را کاهش دهد.

از لحاظ ~~تفاوت~~ Fairness و ~~تفاوت~~ BBR استفاده می‌کند

از لحاظ Fairness استفاده از BBR بسیار بهتر است اما در BBR از resource بیشتری استفاده می‌کند که اگر در سیستم ورودی استفاده شود مشکل آفرین است

**② high retransmission rates:** در BBR حدود ۱٪ packet ها

(تکرار ظرفیت)

retransmitte می‌شوند چون وقتی نرخ به پای بند ارسال را انجام می‌دهیم

ممکن است سری packet ها Drop شوند و ...



این چالش ها مربوط به سال ۲۰۲۰ بوده است و حل این است اکنون این چالش نباشد!

BBR : Employed by Amazon در Cloud Front  
cloudFront

استاندارد می شود. و استاندارد گنو ۲۲ / throughput را افزایش دهد (استاندارد BBR)

در بازه های ۱۰ دقیقه ای است و در یک زمان BBR  
در شبکه اعمال شده است و latency را ۱/۴ کاهش  
داده است و در وقت latency کاهش پیدا می کند throughput افزایش می یابد

Source - Based Approaches (Vegas, BBR, DCTCP) → DCTCP

DCTCP (Datacenter TCP) : یک روش Top congestion control  
است که در کنار سیستم کاری کند. و با هدف به کارگیری در Data center ها طراحی شده است.

انواع Data center : Data center ① : Data center (در بعضی دارند) یکی Data center  
در جهان وجود دارند به هم در ارتباطند که نحوه ارتباط این ها یک مسئله است که روش BBR  
استفاده می شود. و مسئله بعدی که شبکه Inter data center است و نحوه آپدیت شدن است  
مثلاً یک Data center در اروپا هست که می تواند اطلاعات را در آسیا هم آپدیت کند.

- مسئله بعدی، ارتباط بین node داخل Datacenter است که روش DCTCP  
این موضوع را به هم پیوسته می کند. Inter Data center Network ① ⇒ انواع  
② Intra Data center Network

در DCTcp ← flow ها مشکل های مختلفی دارند.  
روش که اول استفاده می شود روش rate بوده است اما سنجی آن براساس می باشد.

\* یعنی که DCTcp را ارائه داده است مایکروسافت هستند.

\* ایده DCTcp این است که وضعیت گنگونی از congestion را بررسی می کند و اگر کم شود

Q از مقدار بیشتر شد فقط warning ارسال می کند. و نسبت تعدادی که مشکل

پیدا شده اند را نسبت آن را بررسی می کند و به همان نسبت window را کوچک می کند.

$$K > (RTT \times C) / 7$$

\* اندازه کسین در DCTcp :

↓  
link rate  
switch مورد نظر

اگر رابطه بالا برقرار باشد، بیت CE را داخل IP header به set می کنیم.

پس اگر طول بافر از مقدار بالا بیشتر شد به switch این کار را انجام می دهد.

در بیت گیرنده seence وجود دارد. اگر بیت CE به set شود و seence = false

شود (یعنی گیرنده در این state هست که هیچ congestion توسط رورتر report

نشده است) آنگاه آنرا به True تبدیل می کند و ack ها را ارسال می کند.



• اگر CE ← set شده باشد و seence = True ← تبدیل False می‌شود  
و مشکلی وجود ندارد.

• بقیه موارد : ignore the CE bit.

درست فدر سفته نرخ رخداد congestion را مشخص می‌کند. نرخ گسست های packet های

اگر اعلام congestion برایشان شده است، در نظری می‌روند و چه گسست های

packet های ارسال شده warning ← congestion دارند و بدین اساس

متغیر را عوض می‌کند.

$$\alpha \leftarrow (1-g)\alpha + gF \quad 0 < g < 1$$

$\alpha$  : نرخ مقدار ← ave ← fraction ← packet های congested برایشان رخ داده است.

$F$  : زمانی که در یک بازه زمانی بر اساس RTT گسست دریافت شده است که گسست از  
آنهاست.  $CE=1$  و گسست  $CE=0$  بوده است. آنگاه  $CE=1$  بوده است و تقسیم بر  
کل packet های گسست آنگاه  $CE$  برایشان warning رخ داده است را در آن می‌آوریم.

avoidance

DC TCP هم کارایی را Top انجام می‌دهد و اتفاقاً وقتی وارد فاز congestion شده

slow start ، additive increase و recovery را انجام می‌دهد فقط این

تفاوت congestion avoidance نرم تر می‌شود

$w \leftarrow w + 1$  if none of the packets are marked in a window ①

$w \leftarrow w(1 - \frac{\alpha}{2})$  if one or more packets are marked per window ②

① اگر هیچ packet - mark نشود  $w$  یکی اضافه می شود.

② براساس  $\alpha$  fraction به نسبتی بخوبی کاهش می دهد.

~~نکته~~ DCTcp، جزء AQM دسته بندی می شود.

- Due to use of ECN, DCTcp sometimes categorize as an AQM method.

- Before introducing DCTcp, RED (with ECN) was the major congestion control scheme in data centers.

۱- RED و ECN استفاده می شود به عنوان major congestion control در دیتا سنترها.

- A switch that supports RED can be reconfigured to do this by setting both the low and high threshold to  $K$  and marking based on instantaneous rather than average queue length.

- در دیتا سنترها، RED را می توان به صورتی که DCTcp را شبیه می کند، پیکربندی کرد.

همچنین می توان RED Enable / Disable و DCTcp Enable / Disable را تنظیم کرد.



\* نکته: روشی است که علاء congestion control را در لایه Application انجام می دهد و روی UDP سوار شده است و کاری به Top ندارد.

### « Network softwarization »

open flow: یک پروتکل در SDN است که علاء به شکل گرفتن و معرفی شدن آن

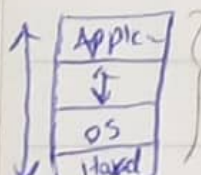
بایست شد SDN یعنی پدید آمدن SDN اینطور است که شبکه را به صورت Centralize

مدیریت می کند. ولی Centralize کردن از لحاظ عملیاتی شدنی نیست! اما

open flow باعث فکر کردن به ایده Centralize شده و این کار عملی است.

\* شبکه های سنتی به این شکل هستند که شبکه به صورت vertically است و از بالا تا پایین به هم وصل شده است و integrated طراحی شده است.

مثلاً وقتی که روتر Cisco (سیسکو) داریم و اینکه از چه Hardware استفاده می کنیم روی چه OS سوار است، چه روش هایی دارد و ... طراحی همه این لایه ها با

توجه به Hardware طراحی می شود. (هماهنگ هستند) close designing  

 و چون در هم آمیخته هستند، بهتر کار می کنند.

عیب: general نیست و نمی توان یک Application دیگر را عنوان کرد.

و اگر هر مشکل پیش بیاید باید به Cisco مراجعه کرد و اگر آپدیت بخواهد انجام شود باید در سیسکو آپدیت انجام شود.

اگر بخواسیم که وضعیت بهتری داشته باشیم باید روتر جدیدی خریداری شود.

۱) تغییر خاص نمی توان دارد. ۲) هزینه ۳) ایجاد پروتکل و به کار گیری آن خیلی کند است.

## Traditional Network Equipment:

Routing, management,  
mobility management,  
access control, VPN, ...

Feature

Feature

→ میلیون ها خط

OS

→ خیل هفتاد

Custom Forwarding Hardware

→ میلیون ها سخت افزار قوی

می توانستیم به جای این همه چیز خط که اضافه کنیم تا بدست کارکنند، کدهای قبلی را  
کدهای قبلی

پارکینگ و مجبور ... ولی می توانستیم کدها non-writeable شده اند و مجبورند  
کدهای دیگری اضافه شوند.