R-TCP: A Framework to Optimize TCP Performance Over Rate-Limiting Networks

Shengtong Zhu *, Yan Liu^, Lingfeng Guo^, Jack Yiu Bun Lee *
The Chinese University of Hong Kong *, Independent Researcher ^

Introduction of Rate-Limited Mobile Networks

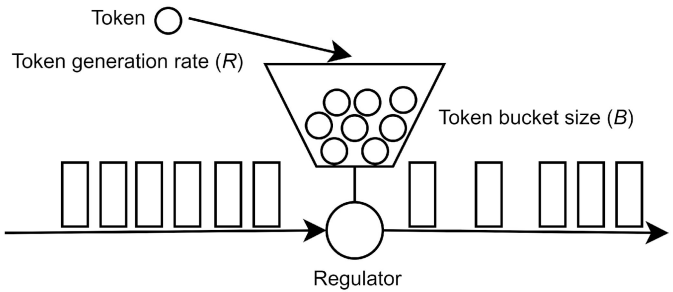
- Data quota
  - The amount of data transfer allowed at full speed
  - Usually on a monthly basis
- If data quota is exhausted
  - The subscriber will be subject to a much lower bandwidth

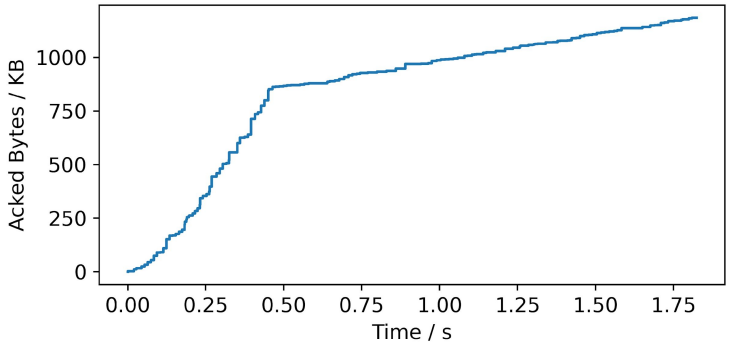
- Rate-limited mobile data plans around the world

Operator / Region served	Data quota	Rate limit beyond quota
Verizon / USA	5/25/100/150 GB	600 Kbps
T-Mobile / USA	20 GB	1.5 Mbps
Optus / Australia	20/80/200/500 GB	1.5 Mbps
Vodafone / New Zealand	5/12/40/100 GB	1.2 Mbps
EE / UK	2/5/25/125 GB	0.5 Mbps
HKBN / HK	0 GB	2 Mbps
BELL / Canada	10/25/45/50 GB	512 Kbps
TELUS / Canada	25/45/50 GB	512 Kbps

Implementation of Mobile Network Rate-Limiting

- Rate limiter implementation
  - Token bucket
    - Token bucket size : *B* bytes
    - Token generation rate: *R* bps
    - Tokens are replenished into the bucket at the fixed rate *R*
    - Bucket is full -> Further generated tokens are discarded
    - Transmission of a packet consumes tokens in the bucket equal to the packet size
    - Insufficient tokens for transmitting the packet
      - Traffic policer: Drop the packet
      - Traffic shaper: Queue the packet in the buffer

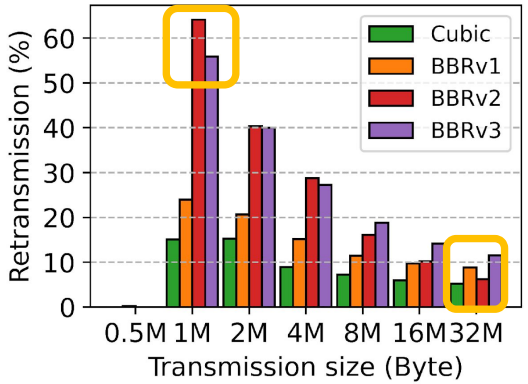




Impact of Mobile Network Rate-Limiting

- A significant problem investigated with the industry collaborator
  - Mobile network rate limiting can cause **excessive retransmission** at the transport layer
- 4G Rate limiting network (Bucket size: 730 KB, Rate limit: 2 Mbps)
  - For files of **1 MB**, retransmission rates exceed **50%** in the cases of BBRv2 and BBRv3
  - For files of **32 MB**, retransmission rates exceed **10%** in the cases of BBRv3

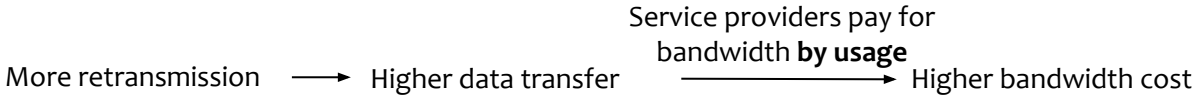
The normal retransmission rate under non-rate-limiting network should be less than 1%



Idle for 15s between transfers. Each data point is an average of 30 runs

Impact of Mobile Network Rate-Limiting

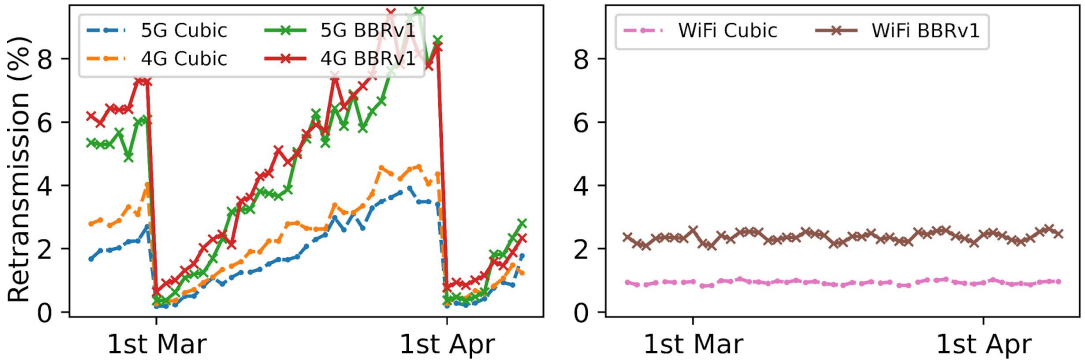
- Bandwidth cost estimation
  - Peak bandwidth usage: 100 Tbps
  - Average bandwidth usage: 20% peak bandwidth usage
  - Transmission cost of Tencent CDN: USD 50 / TB



1% increase in bandwidth consumption -> more than **USD 3M** extra bandwidth cost *per month*

Mobile Network Rate-Limiting in Practice

- Retransmission rates measured from a production short-video service (6 million requests per day)
  - Sawtooth pattern in **mobile networks**:
    - The **lowest** at the **beginning** of the month, progressively **increases** towards the **end** of the month
    - Coincide with the way that data quota is managed in the measured region
      - Data quota is reset at the beginning of the month
      - As the month progresses, more users exhausted data quotas and became subject to rate-limited service
      - Explain why the retransmission rate is so low at the beginning of a month
  - WiFi networks do not show such pattern
    - Eliminate other possible contributing factors for sawtooth pattern in mobile networks

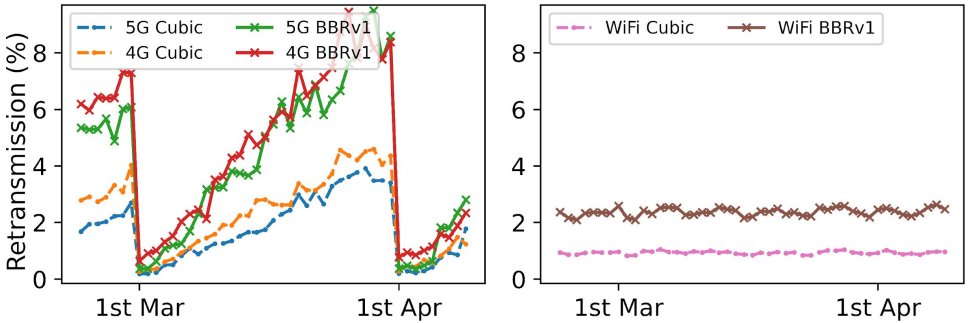


Mobile Network Rate-Limiting in Practice

- Estimate the proportion of flows that are subject to rate limiting
  - Assume all flows are not subject to rate limiting on the 1st of the month
  - The average retransmission rate for a TCP flow under rate limiting : 29.6%
    - Obtain from a top-10 production short-video service

The proportion of rate-limited flows on the last day: 28%

Noting the nearly linear increase across a month, rate limiting poses a challenge



R-TCP

Detection Trigger

Avoids overhead on normal flows

Paramter Estimation

Use curve fitting to estimate token bucket parameters {B, R}

Classification

Two conditions must both hold:

- Abrupt rate drop
- Conve convergence

Retransmission Optimization

Cap CWnd for Cubic
Cap Pacing rate for BBR

R-TCP Rate Limiting Detection -- Detection Trigger

- Many flows not subject to rate limiting, not perform processing on those flows

- The most significant impact of rate limiting is the excessive retransmissions
  - Motivate the use of packet loss to trigger rate limiting detection
  - Parameter estimation and detection **only begin**, if the packet loss rate in α (e.g., 7) rounds exceeds β (e.g., 0.2):
    - l (d) are the number of lost (delivered) packets during α rounds

$$l/(d+l) > \beta$$

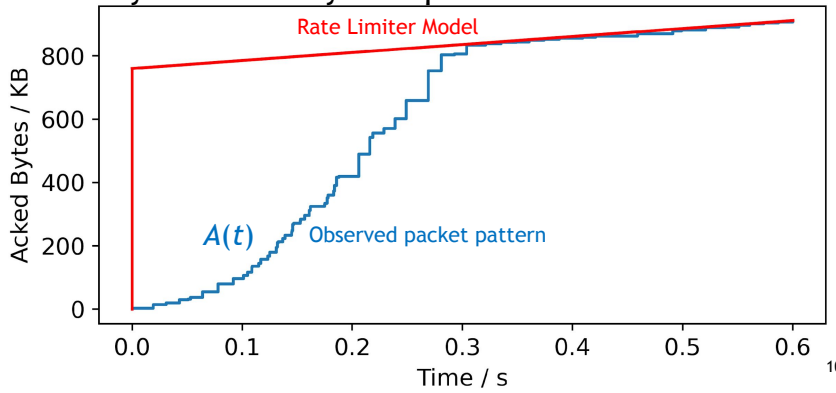
- Flows not subject to rate limiting are likely to exhibit lower packet loss rates
  - Not trigger the detection trigger, so not incur any further processing overheads

R-TCP Rate Limiting Detection – Parameter Estimation

- Principle
  - Find the **rate limiter model** that best fits the **observed packet pattern**
  - The best-fitting model's parameters are then used to detect rate limiting
  - If rate limiting is present, the best-fitting model's parameters are the estimated result of rate limiter

- **Observed packet pattern:** $A(t)$
  - Actual cumulative data function
  - Count the number of bytes reported by ACK to have been delivered during the time (0, t]
  - t_i: the arrival time of ACK packet i s_i: the number of bytes acked by ACK packet i

$$A(t_i) = \sum_{j=1}^{i} s_j$$



R-TCP Rate Limiting Detection – Parameter Estimation

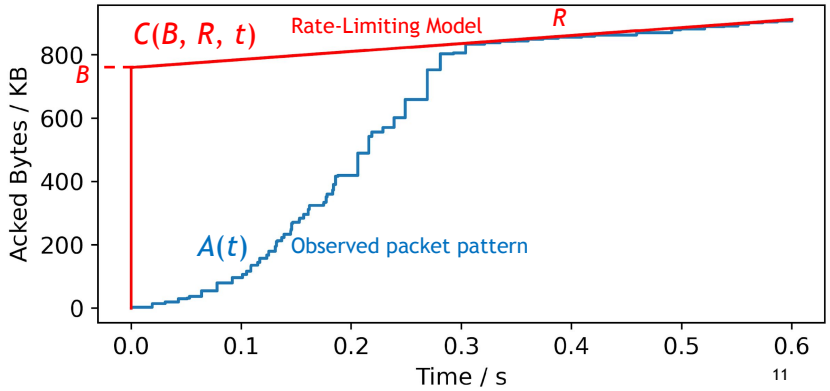
- Principle
  - Find the **rate limiter model** that best fits the **observed packet pattern**
  - The best-fitting model's parameters are then used to detect rate limiting
  - If rate limiting is present, the best-fitting model's parameters are the estimated result of rate limiter

- **Rate-Limiter Model:** $C(B, R, t)$
  - *B:* Bucket size *R*: Rate limit
  - Maximum cumulative data transmission function
  - Specify the maximum amount of data that can be delivered from 0 s to time *t* under $\{B, R\}$

$$C(B, R, t) = B + Rt$$

- **A(t) must be bounded from above by C(B, R, t)**
  - Actual cumulative function ≤ Maximum cumulative function

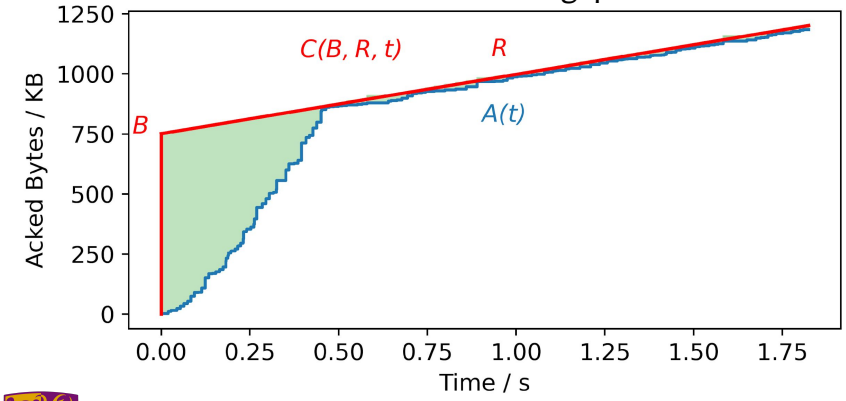
$$A(t_i) \leq C(B, R, t_i)$$



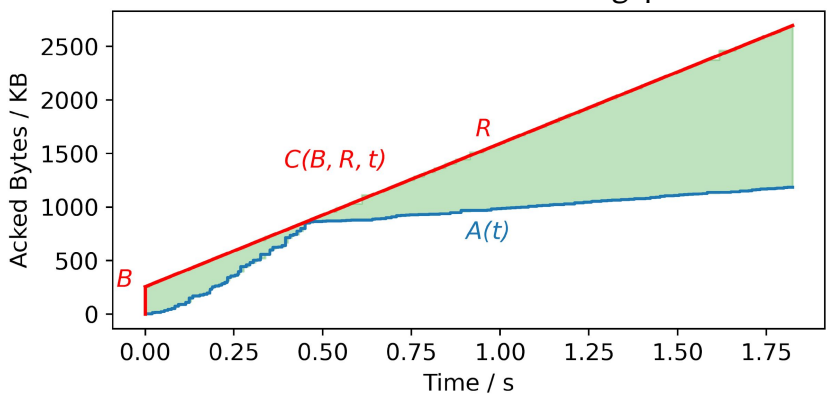
R-TCP Rate Limiting Detection – Parameter Estimation

- Challenge
  - The parameters {*B*, *R*} are not known a priori
  - Find the best-fitting parameters of rate limiter model *C*(*B*, *R*, *t*) based on the observed *A*(*t*) curve
- R-TCP's principle
  - Find {*B*, *R*} such that the **area gap** between *A*(*t*) and *C*(*B*, *R*, *t*) is minimized

Minimize the area gap



Not minimize the area gap



R-TCP Rate Limiting Detection – Classification

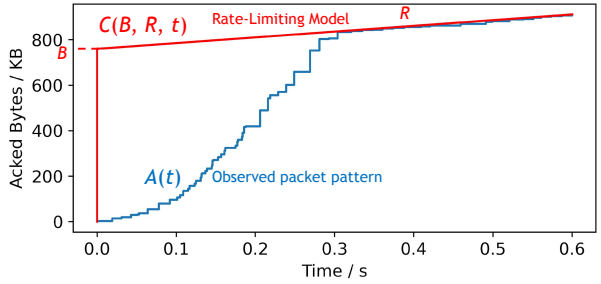
- After detection is triggered, use the best-fit {*B*, *R*} tuple to classify
  - Two conditions must be satisfied simultaneously

- 1. Abrupt rate reduction
  - After the tokens are exhausted, the rate is limited to the rate lim
  - Compare the estimated rate limit with the rate before detection
    - Rate before detection triggers, R_h:

$$R_h = A(t_h)/t_h$$

    - *R* is required to be smaller than R_h by a factor of Θ:

$$R/R_h < \theta$$



- 2. Convergence
  - After the tokens are exhausted, $A(t)$ should converge to $C(t)$
    - The estimated {*B*, *R*} in ε (e.g. 10) consecutive rounds is required to remain unchanged

R-TCP Performance Optimization

- Designed in conjunction with the employed congestion control algorithms (CCAs)
  - Regulate transmission to keep $A(t)$ within the confine of the rate limiter $C(B, R, t)$
    - Control the CWnd or pacing rate

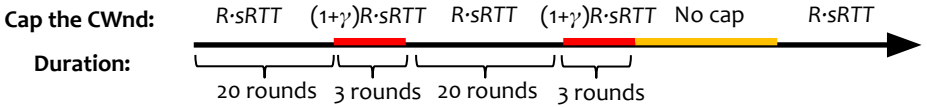
- TCP-Cubic optimization
  - Constrain the inflight packets to one BDP by capping the CWnd

$$cwnd = \min(cwnd, R \cdot sRTT)$$

  - Cater to potential estimation errors
    - Once every η (e.g. 20) sRTT rounds, the CWnd cap is increased by γ (e.g. 20%) for three rounds

$$cwnd = \min(cwnd, (1+\gamma)R \cdot sRTT)$$

    - During these three rounds
      - If $\{B, R\}$ remains unchanged, the original CWnd cap is reapplied
      - Otherwise, the estimation result may be inaccurate
        - The upper limit is lifted, allowing CWnd to grow according to the original Cubic algorithm
        - Once $\{B, R\}$ remains unchanged for three consecutive rounds, it will reactive the CWnd cap



R-TCP Performance Optimization

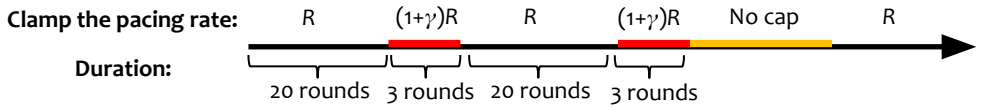
- TCP-BBR optimization
  - Clamp the applied pacing rate, denoted by r^*, to the estimated rate limit R:
    - Original pacing rate: r

$$r^* = \min(r, R)$$

    - Cater to potential estimation errors
      - Once every η (e.g. 20) sRTT rounds, the upper limit R is increased by γ (e.g. 20%) for three rounds

$$r^* = \min(r, (1+\gamma)R)$$

    - During these three rounds
      - If $\{B, R\}$ remains unchanged, the original pacing rate cap is reapplied
      - Otherwise, the estimation result may be inaccurate
        - The pacing rate cap is lifted, allowing BBR to enter probing phase to probe for bandwidth normally
        - Once $\{B, R\}$ remains unchanged for three consecutive rounds, it will reactive the pacing rate cap



R-TCP Performance Evaluation

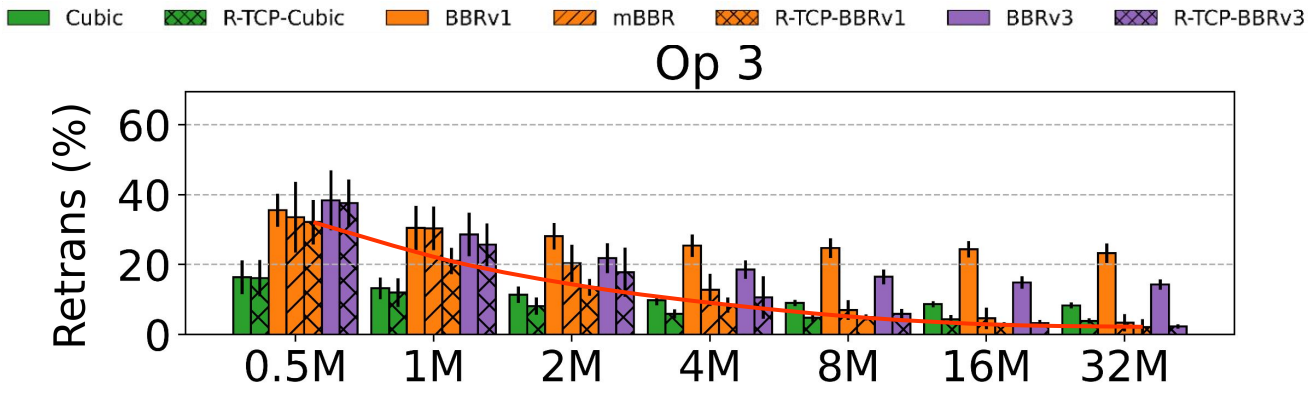
- Implemented R-TCP into Cubic, BBRv1, BBRv3
  - Become R-TCP-Cubic, R-TCP-BBRv1, R-TCP-BBRv3, respectively
    - dynamically loaded/unloaded into/from the Linux kernel in the same way as standard modules
  - Apart from the TCP congestion control module, no other kernel modifications are needed

- Experiments
  - File download (single transfer)
  - On-demand video streaming (multi-transfer)
  - A production short-video app (concurrent transfers)

	Op 1	Op 2	Op 3	Op 4	Op 5
Network type	4G	5G	5G	4G	4G
Mean RTT (ms)	36	18	21	45	36
Advertised *R* (Mbps)	2	1	1	–	–
Est. *B* Median (KB)	733	385	223	–	–
Est. *B* Std (KB)	1.42	61.03	27.05	–	–
Est. *R* Median (Mbps)	2.07	1.3	1.04	–	–
Est. *R* Std (Mbps)	0.0004	0.049	0.0003	–	–

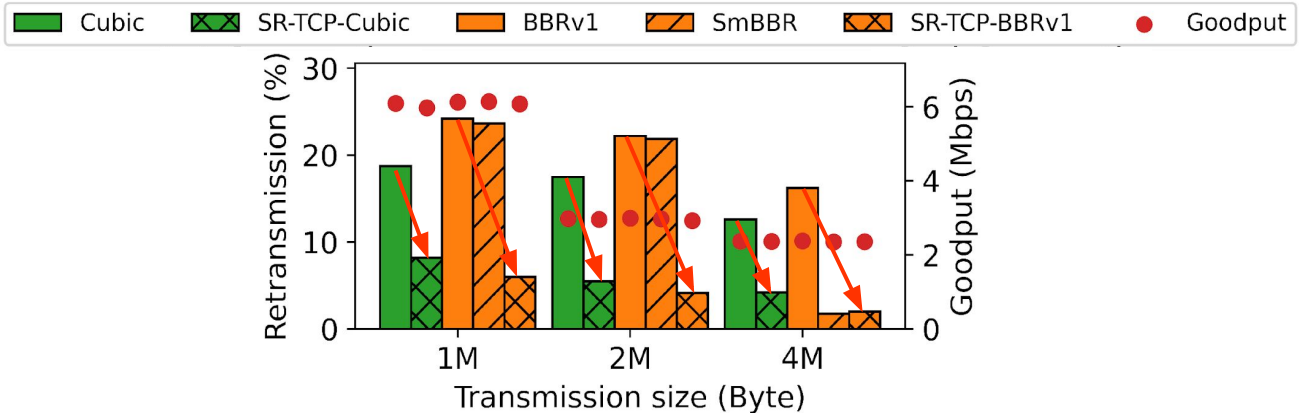
R-TCP Performance Evaluation

- R-TCP reduced the retransmission percentages of **Cubic by 57%, BBRv1 by 80%, and BBRv3 by 76%** at 32-MB file size and averaged over all three operators
- The reduction in retransmission decreases as the file size decreases
  - R-TCP takes a certain amount of data transfer to detect rate limiting and then perform optimization.
  - The amount of data needed is most influenced by the token bucket size



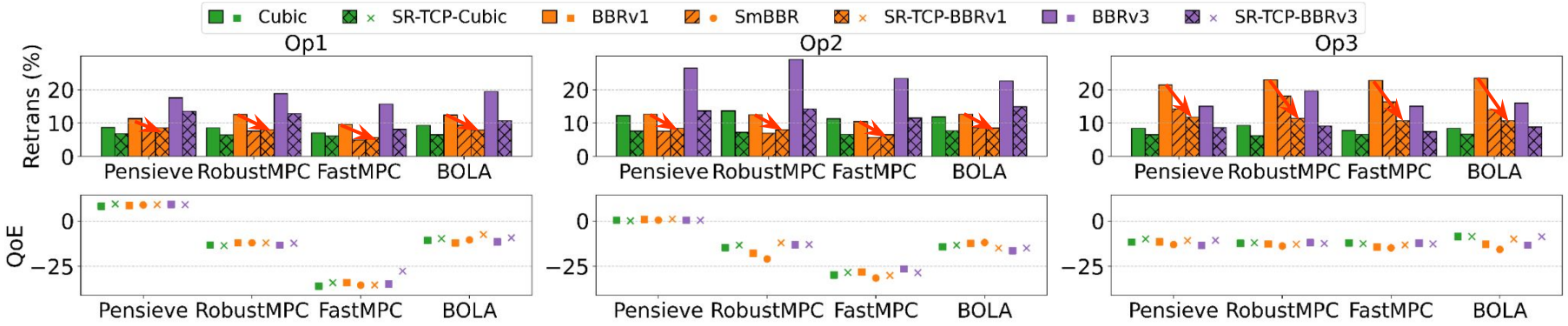
R-TCP Performance Evaluation

- Downloading files of the same size
  - Integrate R-TCP with Stateful to cache and use the detection results from a previous TCP flow to optimize subsequent TCP flows to the same client right from the beginning
  - Focus on the second connection where stateful optimization can be applied
  - R-TCP achieves up to 88% lower retransmission
    - Integrate with Stateful to apply optimization early on to prevent the initial burst loss



R-TCP Performance Evaluation

- DASH Video Streaming
  - Reduce retransmission averaged over all configurations, with no noticeable tradeoff on QoE
    - 28.4% (vs. Cubic), 39.9% (vs. BBRv1), 43.5% (vs. BBRv3)
  - QoE is primarily determined by the ABR algorithms in Op 1 & 2
    - Accumulated tokens at the beginning misleads the overestimation of bandwidth
  - QoE is similar in Op 3
    - Small bucket size leads to not severe overestimation



R-TCP Performance Evaluation

- Short-video applications
  - Each viewing session played 60 videos.
  - The results are averaged over 30 viewing sessions
    - R-TCP reduces the retransmission by 40% (vs. BBRv1)and 60% (vs. Cubic)
    - Streaming performance under R-TCP is similar to BBRv1 and Cubic
    - SmBBR exhibits more than double the rebuffering

		Avg. Retrans. (%)	Rebuf. Count Ratio	Rebuf. Dur. Ratio	Avg. Bitrate (Kbps)
Op 1	BBRv1	13.5	2×10^{-7}	0.7×10^{-3}	572
	SmBBR	6.9	7×10^{-7}	2×10^{-3}	548
	SR-TCP-BBRv1	6.6	3×10^{-7}	0.5×10^{-3}	565
	Cubic	8.9	2×10^{-7}	4×10^{-4}	580
	SR-TCP-Cubic	5.3	8×10^{-8}	2×10^{-4}	584
Op 2	BBRv1	18.4	0.4×10^{-5}	1×10^{-2}	533
	SmBBR	7.3	1×10^{-5}	1×10^{-2}	542
	SR-TCP-BBRv1	7.3	0.4×10^{-5}	1×10^{-2}	537
	Cubic	12.8	4×10^{-6}	1×10^{-2}	536
	SR-TCP-Cubic	8.1	4×10^{-6}	1×10^{-2}	535

Rebuffer count (duration) ratio :
Total number (duration) of rebuffering events / total viewing duration

The advantages of R-TCP:
Be able to be applied to different CCAs

Retransmission reduction is accomplished without sacrificing QoE

Thank you

Shengtong Zhu

Department of Information Engineering
The Chinese University of Hong Kong