DOI:10.11918/j.issn.0367-6234.201901020

BBR 拥塞控制算法在无线网络中的性能改进

董瀚泽[1,2],郭志川[1,2]

(1. 中国科学院声学研究所国家网络新媒体工程技术研究中心, 北京 100190; 2. 中国科学院大学, 北京 100049)

摘 要: 传统的拥塞控制算法已经不能满足当前复杂的网络环境,谷歌提出的 BBR 算法(Bottleneck Bandwidth and Round-Trip)为拥塞控制提供了一种新思路,它可以在具有一定丢包率的网络链路上充分利用带宽,并保证较低的时延.但是该算法存在以下问题:首先,当无线网络的时延剧烈抖动时,BBR 具有很低的传输速率,即便网络不丢包且此时未发生拥塞,这一问题在以往的论文中还没有人提出过;其次,BBR 对网络带宽的降低不够敏感.本文详细分析以上问题出现的原因,进而提出改进 BBR 算法:通过比较 RTT 的均值和标准差判断网络时延的抖动程度,在时延抖动很剧烈时,使用 RTT 的均值取代最小 RTT 来计算拥塞窗口;在网络不稳定时,降低 PROBE_BW 状态中平稳阶段的时间长度.在实际网络中的实验表明,改进后的 BBR 算法几乎不受时延波动的影响,随着时延波动程度的提高,改进后算法的传输速率基本保持不变,在 BBR 几乎不能工作时仍能保持正常的传输速率;而且改进后的 BBR 算法在网络不稳定时能够更快地探测到网络带宽的降低并收敛.

关键词: BBR 优化;拥塞控制;无线网络;收敛速率

中图分类号: TP393.06 **文献标志码:** A **文章编号:** 0367-6234(2019)11-0063-05

Performance improvement of BBR congestion control algorithm in wireless network

DONG Hanze[1,2], GUO Zhichuan[1]

(1. National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China; 2. University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Since traditional congestion control algorithm cannot fulfill the requirements of current complex networks, Google provides a new way named bottleneck bandwidth and round-trip (BBR) for congestion control. The purpose of BBR is to maximize throughput and minimize delay on a bottleneck link with packet loss, but it has some disadvantages: First, when the delay of wireless network is severely jittered, even if there is no packet loss or congestion, the delivery rate of BBR is very low. This issue has not been raised in previous studies. Second, BBR is not sensitive enough to the bandwidth drop. This paper analyzes the causes of the above problems in detail and proposes an optimized BBR, which can judge the jitter degree of network by comparing the mean and standard deviation of RTT. When the jitter of delay was severe, the mean RTT was used instead of minimum RTT to calculate congestion window. When the network was unstable, the length of stationary phase in PROBE_BW was reduced. Experiments in the real network showed that the optimized BBR was almost unaffected by delay fluctuations, and could maintain high bandwidth utilization when BBR could hardly work. Besides, it could detect bandwidth degradation and converge faster than BBR when the network was unstable.

Keywords: optimization of bottleneck bandwidth and round-trip time; congestion control; wireless network; convergence rate

目前广泛使用的拥塞控制算法如 New-Reno[1]、CUBIC[2]都属于基于丢包的算法,即将丢包作为网络链路发生拥塞的信号.在互联网的初期,路由器和交换机上只有很浅的队列,并且网络环境比较简单,带宽较低,这个策略可以很好地实现网络上的拥塞控制.但是随着互联网的发展,如今的网络的情况已经大有不同:

首先,摩尔定律使得存储器的价格越来越便宜,网络设备上的缓存队列越来越深.一个 TCP 连接在建立后会逐渐增大拥塞窗口,先填充链路,再填充路由器等网络中间设备的缓存队列,当链路上的深队列被填满后网络设备会把多余的数据包丢弃,此时才会发生丢包.实际上在队列被填满之前,网络上已经发生了拥塞.因此,基于丢包的算法不但容易造成丢包而且会带来较大的时延[3].

收稿日期: 2019-01-03
基金项目: 国家科技重大专项"新一代宽带无线移动通信网"子课题"5G 与信息中心网络(ICN)融合技术研发"(2017ZX03001019)
作者简介: 董瀚泽(1995—),男,硕士研究生
通信作者: 郭志川,guozc@dsp.ac.cn

其次,近些年来 Wi-Fi、4G LTE 等无线网络已经得到了广泛应用,无线网的信号不稳定,经常会出现噪声丢包;广域网也有较高的丢包率. 而基于丢包的算法不能区分噪声丢包和拥塞丢包,只要有丢包事件发生就会减小拥塞窗口. 以 CUBIC 为例,当丢包率为 0.01% 时,性能已经下降了一半,当丢包率超过 1% 时,已经几乎不能工作了.

因此,由于噪声丢包和深队列等原因的存在,丢包与拥塞已经不再具有紧密的联系.

最近,谷歌提出的 BBR 算法[4-5]为拥塞控制提供了一种全新的思路,它致力于解决两个问题:降低网络链路上的队列深度,从而降低传输时延;实现在具有一定丢包率的网络链路上充分利用带宽.

BBR 会探测网络链路的最大带宽 $B_{\max}$ 和最小往返时延 $T_{\min}$,根据这两个值计算拥塞窗口和数据发送速率,以最大化吞吐率并最小化时延. BBR 只运行于发送端,不需要对协议、接收端、网络做任何改变,因此可以方便地运行于大多数传输协议中. 谷歌已经将它部署在内部的 B4 广域网和 YouTube 服务器上,运行结果表明,与 CUBIC 相比,BBR 能够显著提高吞吐率并降低时延. 文献[6-7]测试了 BBR 在移动网络中的性能,结果表明 BBR 的性能要优于 CUBIC.

但是在一些无线网络环境下,当网络的时延抖动剧烈时,BBR 的性能会急剧下降. 本文发现这是因为 BBR 关于 RTT 的一个重要假设在网络时延剧烈抖动时不成立,并提出了相应的优化措施. 本文还针对 BBR 对网络带宽的降低不够敏感的问题进行了优化. 在实际网络中的测试结果表明,改进的 BBR 算法在时延剧烈抖动时仍可以保持高吞吐率,并在带宽降低时具有更快的收敛速率.

1 BBR 拥塞控制算法

1.1 BBR 的设计思想

吞吐率和 RTT 可以反映出一条网络链路的性能. 图 1 显示了一条链路的 RTT 和吞吐率随 inflight 增加的变化趋势[4],inflight 为链路上正在传输的数据,即所有已发送但还未确认的数据包.

图中区域被两条虚线分为三部分. 左侧部分,网络链路上的数据较少,RTT 一直维持在最小值,吞吐率随链路中数据量的增大而提高;中间部分,网络链路已经被填满,吞吐率达到最高值,不能继续增长,多发送的数据会被填充到网络中间设备的缓存队列中,因此 RTT 会逐渐增大;右侧部分,缓存队列也被填满,此后多发送的数据包会被路由器丢弃,发生丢包.

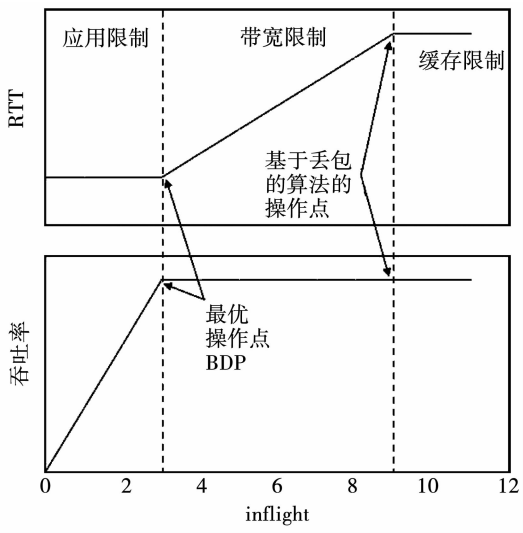

图 1 RTT、吞吐率与 inflight 的关系

Fig. 1 Relation of RTT and throughput with inflight

基于丢包的拥塞控制算法工作于右侧的虚线与两条曲线的交点处,此处可以充分利用带宽,但以较高的时延和频繁丢包作为代价. 文献[8]证明了左侧虚线的交点是拥塞控制的最优操作点,此处 inflight 的值为吞吐率与 RTT 的乘积(bandwidth-delay product, BDP),可以同时保证最大的传输速率与最小的时延. 但不幸的是,Jaffe[9]证明不可能使用一个分布式算法来收敛到这个最优点.

BBR 巧妙地解决了这一问题,如图 2,BBR 设计了一个状态机,使用状态机的两个状态交替测量最大带宽 $B_{\max}$ 和最小往返时延 $T_{\min}$. 相应地,BDP 为

$$B_{\mathrm{BDP}} = B_{\max} \cdot T_{\min}. \tag{1}$$

从而使算法收敛到最优点附近.

BBR 主要维护两个控制参数:传输速率 pacing_rate 和拥塞窗口 cwnd. 传输速率等于当前带宽乘以增益系数 pacing_gain,是最主要的参数,控制数据的发送速率,防止发送窗口中的数据一次性发出而阻塞链路. 拥塞窗口等于 BDP 乘以增益系数 cwnd_gain,控制当前链路上正在传输的数据总量.

1.2 算法流程

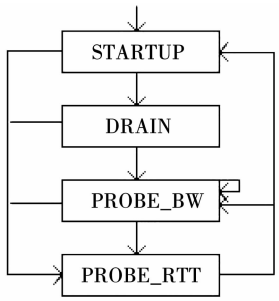

图 2 BBR 的状态机模型

Fig. 2 BBR state machine

BBR 的主体由一个状态机构成,如图 2 所示,

共有 4 个状态：STARTUP、DRAIN、PROBE_BW、PROBE_RTT 其中 BBR 生命周期的绝大部分时间都处于 PROBE_BW 阶段，BBR 使用一个增益数组[1.25,0.75,1,1,1,1,1,1]周期性地改变 pacing_gain，以探测网络链路的可用带宽.

2　BBR 存在的问题

2.1　时延剧烈抖动时传输速率低

在无线网络中 BBR 的传输速率有时会突然降低. 通过实验发现，当网络的时延抖动较为剧烈时，即使网络的丢包率为 0 且未发生拥塞，BBR 的传输速率会还是很低. 如图 4 中 BBR 的传输曲线所示，实验中链路带宽为 3 Mbps，但实际的传输速率仅仅有 0.5 Mbps.

因此这一问题主要是由时延的剧烈抖动所引起的. 可知，BBR 的拥塞窗口的大小为

$$\text{cwnd} = \text{cwnd\_gain} \cdot B_{\max} \cdot T_{\min}. \tag{2}$$

即拥塞窗口取决于探测到的最大带宽和最小往返时延，而探测到的 RTT 满足

$$R_{\text{RTT}} = T_{\text{prop}} + \mu. \tag{3}$$

式中，T_{prop}是链路固有的往返时延，μ 是抖动量，受链路上队列深度等因素的影响.

BBR 使用$T_{\min}$来估算拥塞窗口基于一个假设：RTT 的抖动程度 μ 远小于链路的固有往返时延 T_{prop}，因此一段时间内探测到的最小 RTT 最接近真实的 RTT，即

$$T_{\min} \approx T_{\text{prop}}. \tag{4}$$

有线网络的网络环境比较稳定，这一假设基本成立，但是它不适用于 4G、Wi-Fi 等无线网络.

以视频服务的场景为例. 用户使用手机在 Wi-Fi 或 4G 网络中观看视频，当视频内容提供商部署了边缘 CDN 服务器时，用户与服务器之间的最小时延很小，为几毫秒到十几毫秒；而 Wi-Fi 网络是很不稳定的，其时延会在 4 ms ~ 80 ms 之间波动[10]. 因此，当用户在 Wi-Fi 网络中访问边缘 CDN 服务器时，链路的“固有 RTT”与 RTT 的抖动量处于同一数量级. 此时式(4)不成立，探测到的$T_{\min}$要远小于链路的真实时延.

$T_{\min}$过小会带来一系列影响，如图 3 所示. $T_{\min}$变小使得拥塞窗口变小，若拥塞窗口过小，BBR 的实际发送速率受到拥塞窗口的限制而无法达到 pacing_rate，则 PROBE_BW 状态的 pacing_gain 数组不起作用，无法通过增大发送速率来探测更高的可用带宽. 因此$B_{\max}$不变甚至减小，反过来又导致拥塞窗口变小. 这样形成了一个恶性循环，使得 BBR 最终只能以很低的速率传输数据，无法有效利用网络带宽.

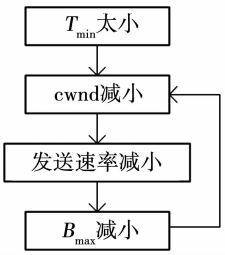

图 3　$T_{\min}$太小带来的影响

Fig. 3　The impact of too small $T_{\min}$

随着移动互联网和 5G[11] 的快速发展，据思科估计，到 2022 年，Wi-Fi 和移动网络将占 IP 流量的 71%[12]. 因此，解决这一问题具有非常重要的现实意义.

2.2　对带宽下降不敏感

PROBE_BW 状态的 pacing_gain 数组为[1.25,0.75,1,1,1,1,1,1]. 在一个周期中，先使用 1.25 倍的增益来探测可用的带宽，再将增益降为 0.75 以排空链路上累积的队列，最后是连续 6 轮的增益为 1 的平稳阶段. 这个平稳阶段是为了保证公平性，在此期间 BBR 不能抢占带宽，使得共享资源的其它连接有充足的时间收敛到公平的状态.

但这是以牺牲了灵活性作为代价的. BBR 使用“乘性增”的方式探测带宽，对带宽增长的反应很灵敏；但是当网络带宽降低时，它不会主动降低拥塞窗口或 pacing_gain，故需要多个周期才能收敛[5]. 若平稳阶段的时间过长会带来较长的收敛时间，在此期间的发送速率高于链路实际带宽，非常容易造成丢包或者拥塞. 在视频直播等业务中，这会造成视频卡顿，给用户带来非常差的观看体验.

3　BBR 性能改进

针对时延剧烈抖动时传输速率低的问题. 由分析可知，当时延波动比较剧烈时，探测到的$T_{\min}$远小于链路的真实时延，此时应该使用 RTT 的均值T_{smo}作为真实 RTT 的估计值，即

$$T_{\text{smo}} \approx T_{\text{prop}}. \tag{5}$$

由于标准差反映了数据的离散程度，可以统计探测到的 RTT 的标准差T_{dev}，用它来判断时延的波动程度.

BBR 中没有记录历史 RTT 的滑动窗口，如式(8)所示，可以使用指数加权移动平均值记录 RTT 的均值T_{smo}. 类似地，使用式(9)可以得到标准差T_{dev}. α，β 为新数据所占的权重.

当时延抖动比较剧烈时式(4)不成立，应改写为$T_{\text{prop}} - T_{\min} > \gamma \cdot T_{\text{prop}}$，结合式(5)可以得到

$$T_{\text{smo}} - T_{\min} > \gamma \cdot T_{\text{smo}}. \tag{6}$$

由于$T_{min} \approx T_{smo} - T_{dev}$，将其代入式(6)可以得到

$$T_{dev} > \gamma \cdot T_{smo}. \tag{7}$$

则当 RTT 波动剧烈时，式(7)成立，此时使用 T_{smo} 计算 BDP；否则，依然使用 T_{min}，如式(10)所示. γ 决定了使用 T_{smo} 替代 T_{min} 的阈值. 若 γ 值过小，BBR 在网络条件较好时也会激进地发送数据，容易引起网络拥塞；若 γ 值过大，则当无线网络的时延波动较为剧烈时 BBR 仍会使用 T_{min}，导致传输速率下降. 文献[13]测量了从天津到北京、香港、美国的有线网络的 RTT，其统计结果显示有线网络中 RTT 标准差与均值的比值不超过 0.3，故本文的实验取 $\gamma = 0.3$.

针对 BBR 对带宽下降不敏感的问题. 在网络不稳定时，可以降低平稳阶段的时间长度，使 BBR 及时地探测到链路带宽的改变. 由于 RTT 能够忠实地反映网络的波动状态，依然可以使用 T_{smo} 和 T_{dev} 判断网络的状态，如式(11)，当 $T_{dev} > \gamma \cdot T_{smo}$ 时认为网络是不稳定的，将 pacing_gain 数组中增益为 1 的数量减为 3 个；否则，保持原数组不变.

$$T_{smo} = (1-\alpha) \cdot T_{smo} + \alpha \cdot RTT, \tag{8}$$

$$T_{dev} = (1-\beta) \cdot T_{dev} + \beta \cdot abs(T_{smo} - RTT), \tag{9}$$

$$B_{BDP} = \begin{cases} T_{smo} \cdot B_{max}, T_{dev} > \gamma \cdot T_{smo}; \\ T_{min} \cdot B_{max}, T_{dev} \leqslant \gamma \cdot T_{smo}. \end{cases} \tag{10}$$

pacing_gain 数组为

$$\begin{cases} [1.25, 0.75, 1, 1, 1], T_{dev} > \gamma \cdot T_{smo}; \\ [1.25, 0.75, 1, 1, 1, 1, 1, 1], T_{dev} \leqslant \gamma \cdot T_{smo}. \end{cases} \tag{11}$$

4 实验和性能分析

为了验证改进后的 BBR 算法的性能，本文在实际网络中进行实验. 使用 QUIC 协议在两台 Ubuntu16.04 主机之间传输文件，QUIC 中使用 BBR 或改进的 BBR 作为拥塞控制算法，对比两者性能. 链路上使用网损仪限制带宽和时延.

算法的具体参数设置如下：$\alpha = 0.125$，$\beta = 0.25$，$\gamma = 0.3$. 其中 α，β 的值参考了 QUIC 源码[14]（QUIC 中统计了 T_{smo} 和 T_{dev} 的值）.

4.1 时延剧烈抖动的网络中的传输速率对比

图 4 显示了 BBR 与改进 BBR 的传输速率随时间的变化. 实验中设置链路带宽为 3 Mbps，设置了呈正态分布变化的往返时延，其均值为 60 ms，标准差为 40 ms.

BBR 在 STARTUP 阶段，由于增益系数较大，且还未采样到特别小的 RTT，传输速率可以短暂地增长到一个较大的值. 但随着采样到的 RTT 值不断减小，拥塞窗口不断减小，导致传输速率逐渐降低. 此后在 PROBE_BW 状态中，虽然 pacing_gain 会周期性地增大为 1.25，但此时拥塞窗口非常小，传输速率受到拥塞窗口的限制，无法探测到更高的带宽，传输速率逐渐降低.

虚线为改进后的算法，由于使用 T_{smo} 取代 T_{min} 计算 BDP，拥塞窗口的大小近似于链路的真实容量，因此传输速率主要受 pacing_rate 控制，出现了 PROBE_BW 状态的周期性波动，并且基本维持在 3Mbps 附近，可以充分利用链路的带宽.

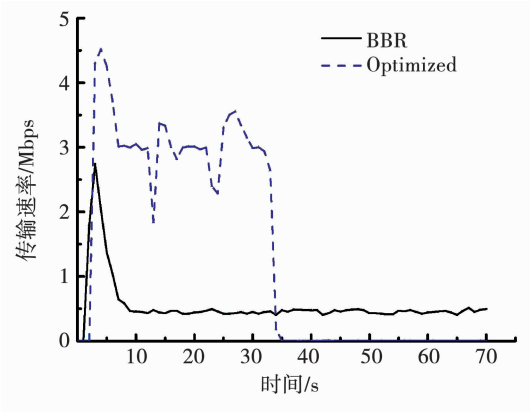

图 4 传输速率随时间的变化

Fig. 4 Delivery rate over time

图 5 为改变网络时延的抖动程度，BBR 和改进 BBR 的平均传输速率的变化曲线. 实验中设置链路带宽为 3 Mbps，时延的均值为 60 ms，将时延的标准差从 0 ~ 60 ms.

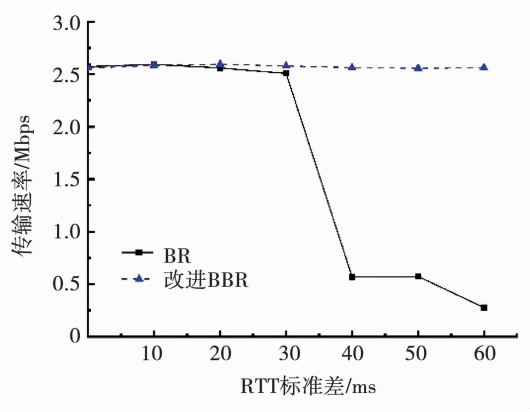

图 5 不同时延抖动程度下的平均传输速率

Fig. 5 Average delivery rate under different jitter degrees

当时延的标准差为 20 ms 时，BBR 的性能已经开始下降，增大到 30 ms 以上时，BBR 的传输速率迅速降低到 0.5 Mbps. 此后时延抖动程度继续增大，传输速率却一直维持在 0.5 Mbps 附近. 这是因为实验中将时延的最小值设为定值 1 ms，所以 BBR 探测到的 T_{min} 是恒定的，计算得到的拥塞窗口的大小不会改变，因此平均传输速率也基本保持不变. 这表明：在 RTT 抖动剧烈时，是 T_{min} 限制了 BBR 的性能. 而优化后的 BBR 的平均传输速率稳定在 2.55 M 附

近，基本不受 RTT 波动的影响，能够充分利用网络带宽.

4.2　带宽改变时的收敛速度对比

图 6 显示带宽改变时 BBR 和优化后的 BBR 的传输速率随时间的变化曲线. 实验中设置链路带宽为 5 ~ 10 Mbps，周期 20 s 的方波，时延为 60 ms.

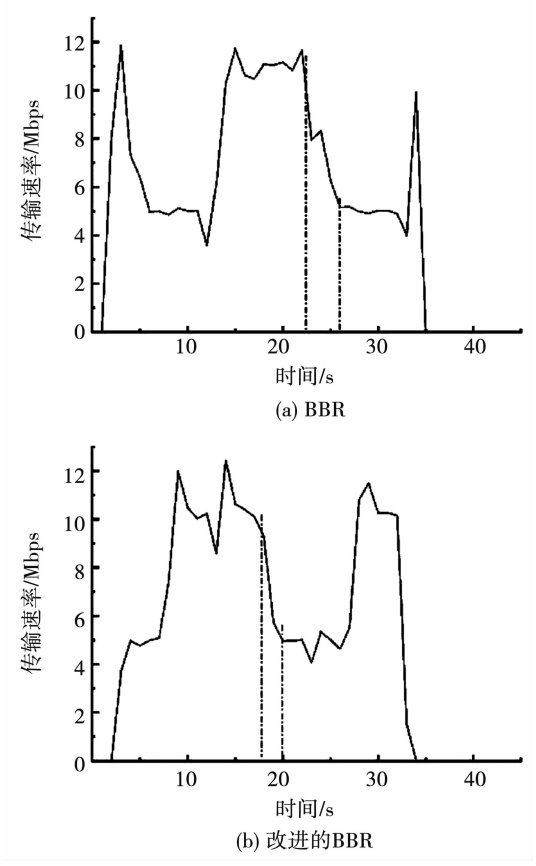

图 6　带宽改变时传输速率随时间的变化

Fig. 6　Delivery rate over time when bandwidth changes

a) 中网络带宽在 13 s 附近升高到 10 M，在23 s 附近降为 5 M；b) 中网络带宽在 8 s 附近升高，在 18 s附近降低. 可以看到当带宽升高时，两者都能很快地收敛. 但当带宽降低时，优化后的 BBR 的收敛时间只有 BBR 的一半，收敛速度显著提高.

5　结　论

当无线网络的时延剧烈抖动时，即使网络丢包率很低且未发生拥塞，BBR 算法的传输速率还是很低. 针对这一缺陷，本文提出了改进 BBR 算法：通过计算 RTT 的标准差，当标准差较大时认为网络的时延抖动剧烈，使用 RTT 的均值代替最小 RTT 来计算拥塞窗口. BBR 还存在对网络带宽降低不敏感的问题，本文根据网络状态灵活调节 PROBE_BW 状态的 pacing_gain 数组中平稳阶段的时间长度，使得算法在网络不稳定时能够更快地收敛.

实验网络测试表明，优化后的 BBR 不受时延抖动影响，可以保持高吞吐率；而且在带宽降低时，算法的收敛速度显著提高.

参考文献

[1] FLOYD S, HENDERSON T, GURTOV A. The NewReno modification to TCP's fast recovery algorithm[R]. No. RFC 3782, 2004

[2] HA S, RHEE I, XU L. CUBIC: A new TCP-friendly high-speed TCP variant[J]. ACM SIGOPS Operating Systems Review, 2008, 42(5): 64. DOI:10.1145/1400097.1400105

[3] GETTYS J, NICHOLS K. Bufferbloat: Dark buffers in the internet [J]. Communications of the ACM, 2012, 55(1): 57. DOI:10.1145/2063166.2071893

[4] CARDWELL N, CHENG Y, GUNN C S, et al. BBR: Congestion-based congestion control[J]. ACM Queue, 2016, 14(5): 50. DOI:10.1145/3012426.3022184

[5] CARDWELL N, CHENG Y, GUNN C S, et al. BBR: Congestion-based congestion control[J]. Communications of the ACM, 2017, 60(2): 58. DOI:10.1145/3009824

[6] LI F, CHUNG J W, JIANG X, et al. TCP CUBIC versus BBR on the highway[C]//International Conference on Passive and Active Network Measurement. Springer, Cham, 2018: 269

[7] ATXUTEGI E, LIBERAL F, HAILE H K, et al. On the use of TCP BBR in cellular networks[J]. IEEE Communications Magazine, 2018, 56(3): 172. DOI:10.1109/MCOM.2018.1700725

[8] KLEINROCK L. Power and deterministic rules of thumb for probabilistic problems in computer communications [C]// Proceedings of the International Conference on Communications, 1979, 43: 1

[9] JAFFE J. Flow control power is nondecentralizable[J]. IEEE Transactions on Communications, 1981, 29(9): 1301. DOI:10.1109/TCOM.1981.1095152

[10] CARDWELL N, CHENG Y, STEPHEN C, et al. BBR congestion control work at Google[DB/OL]. (2018-3)[2019-1-3]. https://datatracker. ietf. org/meeting/101/materials/slides-101-iccrg-an-update-on-bbr-work-at-google-00

[11] 张宇，解伟. 5G 移动与广播电视融合网络[J]. 网络新媒体技术，2018，7(5)：6. DOI：CNKI:SUN:WJSY.O.2018-05-002
ZHANG Yu, XIE Wei. 5G mobile and broadcast TV convergence network[J]. Journal of Network New Media, 2018, 7(5): 6. DOI: CNKI:SUN:WJSY.O.2018-05-002

[12] CISCO. Cisco visual networking index: Forecast and trends, 2017-2022[DB/OL]. (2018-12-26)[2019-1-3]. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[13] 赵伟丰. 基于 RTT 的端到端网络拥塞控制研究[D]. 天津大学
ZHAO W F. Study on RTT-based end-to-end network congestion control[D]. Tianjin: Tianjin University, 2014

[14] Google. Chromium[2019-1-3]. https://cs.chromium.org/chromium/src/third_party/webrtc/modules/congestion_controller/bbr/rtt_stats.cc?g=0

（编辑　苗秀芝）