JCBI
Journal of Computing & Biomedical Informatics
ISSN: 2710 - 1606
Volume 11 Issue 01
2026

Research Article
https://doi.org/10.56979/1101/2026/1515

The Impact of Reverse-Path Bandwidth Asymmetry on TCP Congestion Control: A Telemedicine-Motivated ns-3 Study

Abdulazaz Albalawi[1*]

[1]Department of Computer Science, University College of Duba, University of Tabuk, Tabuk 47913, Saudi Arabia.
[*]Corresponding Author: Abdulazaz Albalawi. Email: aalhamodi@ut.edu.sa

Received: April 20, 2026 **Accepted:** May 30, 2026

Abstract: Telemedicine services in underserved rural regions often rely on asymmetric access technologies. These include Low Earth Orbit (LEO) satellite broadband and 5G Fixed Wireless Access (FWA), which prioritize downstream over upstream bandwidth. This asymmetry can affect reliable transport protocols such as the Transmission Control Protocol (TCP), on which most telemedicine services depend. The impact of bandwidth asymmetry on TCP was studied extensively in the late 1990s and early 2000s, and recent studies have revisited it for modern congestion control algorithms (CCAs), mainly within specific access technologies. In this paper, we revisit the bandwidth asymmetry problem in TCP, isolating reverse-path bandwidth asymmetry as a single controlled variable across three CCAs, namely NewReno, CUBIC, and BBRv1, in the ns-3 simulator (ns-3.47). We also isolate the impact of asymmetry on the ACK-clock mechanism itself, using a fixed-window approach that decouples ACK-clock dynamics from CCA behavior. Our results show that the fixed-window baseline converges to the ACK-clock ceiling imposed by the reverse path, while the adaptive algorithms diverge sharply. We also find that CUBIC's time-based window growth makes it achieve the highest goodput under severe asymmetry, sustaining an average goodput of 7.75 Mbps on the 10 Mbps forward path at a 300:1 ratio. On the other hand, BBRv1 converges to the ACK-clock ceiling without a single packet loss, achieving the lowest average goodput among all configurations at 1.72 Mbps. The findings indicate that TCP performance in asymmetric access networks depends strongly on how each algorithm consumes the ACK stream.

Keywords: bandwidth asymmetry; congestion control algorithms; TCP; ACK clocking; reverse-path constraint; telemedicine; ns-3; CUBIC; BBR

1. Introduction

Telemedicine and telehealth refer to the use of digital technologies to provide healthcare services independent of physical distance between physicians and patients [1]. These services exhibit different delivery models, such as virtual clinical visits, remote patient monitoring through biometric sensors, and health device management [2]. Each service depends on a continuous data exchange between the patient site and the care provider [2, 3]. The quality of such exchange is bounded by the underlying telecommunication infrastructure, whose robustness is critical to telemedicine services, particularly in underserved rural regions [4, 5, 6]. However, the network technologies available in these regions are inherently asymmetric, which can affect telemedicine services.

Modern network technologies that can offer symmetric data rates, such as Fiber-to-the-Home (FTTH), are economically unfeasible in rural deployments [7]. These regions therefore rely on asymmetric wireless technologies that require minimal infrastructure, such as Low Earth Orbit (LEO) satellite broadband and 5G Fixed Wireless Access (FWA) [8, 9, 10], as well as legacy internet access technologies such as Asymmetric Digital Subscriber Line (ADSL) [11]. These technologies, by design, prioritize downstream bandwidth over upstream bandwidth. This is either due to physical limitations or to economic feasibility, given human

consumption habits. Since telemedicine services often run concurrently in the same home, this can compound the problem. For example, upstream sensor data from biometric sensors competes with feedback traffic from downstream flows, such as video consultations. This can make the effective asymmetry worse than the underlying network technologies suggest. The consequences extend to the transport layer itself, as most telemedicine services depend on reliable transport protocols such as TCP to ensure the complete and correct delivery of clinical data.

The reliability of TCP in these telemedicine services is achieved by relying on feedback from the receiver in the form of Acknowledgments (ACKs), which confirm successful data delivery and drive a self-clocking mechanism (also known as the ACK clock), in which the arrival of ACKs paces the transmission of new data. Such a mechanism binds the sender's effective throughput to the rate at which ACK packets arrive, a rate that can be constrained by asymmetric links. The ACK clock also shapes the behavior of congestion control algorithms (CCAs), which regulate the sending rate based on network conditions. These algorithms differ in how they exploit the ACK stream. NewReno [12] depends on ACK arrivals to increment its congestion window. CUBIC [13], in contrast, grows its window toward a target size based on the time elapsed since the last congestion event, rather than incrementing per ACK received. BBRv1 [14] takes yet another approach, using the ACK stream to estimate round-trip time and delivery rate, building an explicit model of the path that governs its sending rate. Regardless of how they regulate their sending rate, these algorithms depend on the ACK stream, either directly or indirectly. Such dependency can impair a CCA's ability to accurately estimate the available capacity when the reverse path is constrained, as in the asymmetric access technologies described above.

The impact of bandwidth asymmetry on TCP performance was extensively studied in the late 1990s and early 2000s [15–18]. Recent studies have revisited reverse-path effects for modern algorithms, including CUBIC and different BBR versions, mainly within specific access technologies such as wireless LANs [19] and LEO satellite links [20], or under a reverse path congested by cross-traffic [21]. These studies examined the problem from complementary angles and in specific settings. In this paper, we revisit the bandwidth asymmetry problem, systematically varying the forward-to-reverse bandwidth ratio as a single controlled variable across NewReno, CUBIC, and BBRv1. The simulations use ns-3.47 [22], a recent stable release of the simulator. We also isolate the effect of asymmetry on the ACK-clock mechanism itself, adopting a fixed-window approach from the literature [16, 23] that decouples ACK-clock dynamics from CCA behavior.

The rest of the paper is organized as follows. Section 2 describes related work, and Section 3 discusses the details of our experimental methodology. Section 4 presents our simulation results. Finally, we present our conclusions in Section 5.

2. Related work

Underserved rural areas have come to rely on remote healthcare services, as access to in-person care is limited [5]. Studies in this domain recognized bandwidth as a limiting factor of service quality [4, 6]. This limitation stems mostly from the asymmetric network technologies available in these areas. The bandwidth asymmetry persists even with modern technologies such as LEO satellite [8, 9], just as with legacy access technologies such as ADSL [11]. In addition, telemedicine services are increasingly operating using IoT-based technology, which tends to follow a sensor-to-cloud architecture in which bulk data flows in the upstream direction [3]. This can magnify the asymmetry problem, especially when sensor data coexists with other telemedicine services that rely on TCP [24]. This is because upstream data will compete with feedback traffic (ACKs) from downstream applications operating over TCP.

The performance degradation of TCP caused by a constrained ACK path under bandwidth asymmetry was first studied in the 1990s [15, 17]. This early work characterized the problem and identified the core mechanisms behind the degradation. It led to mitigation solutions such as ACK filtering, ACK reconstruction, and header compression [25]. Since then, multiple studies have analyzed the impact of this problem on different TCP variants. For example, several works studied Tahoe and Reno using either ns-2 simulation or real testbeds [16, 26, 27]. More recent evaluations of CUBIC and BBR have relied mainly on testbed emulation and field measurements [28, 29], while ns-3 studies have focused on implementing and validating these algorithms and on their fairness across round-trip times and buffer sizes [30–32]. Other

studies evaluated this problem for modern congestion control algorithms in specific access technologies such as Wi-Fi [19].

Collectively, these studies have addressed this problem from complementary angles and in specific settings. In this paper, we revisit the bandwidth asymmetry problem in TCP, isolating reverse-path bandwidth asymmetry as a single controlled variable rather than within a particular access technology. In addition, we adopt a fixed-window approach as a baseline reference in our simulation. This baseline exposes the impact of ACK stream dynamics on the sender's sending rate [16, 23]. We characterize NewReno [12], CUBIC [13], and BBRv1 [14] across the ratio range of deployed access networks, extended to more extreme ratios to test these algorithms under severe asymmetry. While later BBR versions (BBRv2 and BBRv3) revise several mechanisms, including bandwidth probing and ACK aggregation estimation [33], we select BBRv1 as it is the BBR version natively implemented and validated in ns-3.

3. Methodology

This section details the experimental design for our evaluation using the ns-3 simulator. We adopted ns-3.47, a recent stable release. Recent ns-3 releases incorporate accumulated bug fixes, refined implementations of modern TCP variants, and general improvements to the TCP stack. They therefore reflect the behavior of modern Linux TCP stacks more closely than older releases.

While asymmetry in a network may exist in delay, loss rate, or bandwidth, this study focuses on bandwidth asymmetry to provide a controlled evaluation and isolate its effect on TCP performance. For our bandwidth-asymmetric topology, we used a single-bottleneck dumbbell topology with edge links provisioned at 100 Mbps in both directions, each with a 2 ms one-way delay. The forward data path of the bottleneck is fixed at 10 Mbps with a 50 ms one-way delay, giving a total round-trip time (RTT) of 108 ms and a bandwidth-delay product (BDP) of around 93 segments (seg). The reverse acknowledgment path is the single experimental variable, and its bandwidth is swept from 10 Mbps down to 0.033 Mbps. This produces forward-to-reverse asymmetry ratios of 1:1, 10:1, 50:1, 100:1, and around 300:1. These ratios therefore cover a wide range of practical asymmetry scenarios, including deployed access networks and cases where reverse-path capacity is reduced by competing traffic. We included the 300:1 ratio as an experimental upper bound, representing severely constrained uplinks.

To examine the impact of ACK-clocking behavior in TCP under a bandwidth-asymmetric topology, we adopted a fixed-window scheme that decouples congestion control dynamics from the underlying acknowledgment feedback loop [16, 23]. This allows us to examine how the timing of each ACK arrival on a bandwidth-asymmetric reverse path can constrain the sender's throughput. The window is set to around 1.1 × BDP of the forward path, that is, 100 segments. The small margin above the BDP ensures the forward path remains fully utilized under ACK jitter. In addition to the fixed-window scheme, we evaluate three TCP variants: NewReno, CUBIC, and BBRv1. Each algorithm interacts with the ACK stream differently. CUBIC is included as the default TCP CCA in Linux, and NewReno as the standard loss-based baseline. BBRv1 is evaluated because it is the only BBR variant with a stable model in ns-3, and because its delivery-rate estimation depends directly on the ACK stream, making it the variant of greatest interest under reverse-path constraints. Later BBR versions are left to future work.

Table 1 summarizes our experimental parameters, in which propagation delays, queue sizes, segment size, and protocol configuration are held constant across the ratio sweep, where each scenario runs for 120 s. The ratio sweep uses the Linux default delayed ACK behavior (DelAckCount = 2, one ACK per two segments), and the baseline scenario additionally compares DelAckCount = 1 and 2, allowing us to isolate the effect of the delayed ACK mechanism itself. The forward queue is set at 100 packets, matching the fixed-window setting, and the reverse queue at 25 packets. The three CCAs use identical loss recovery settings. Proportional Rate Reduction (PRR) and Selective Acknowledgment (SACK) remain at their ns-3 defaults, with no CCA-specific modification. The fixed-window scheme, in contrast, implements no recovery state. It only tracks whether packets have left the network, since its purpose is to expose the raw ACK-clock dynamics. Therefore, differences among the three CCAs stem from their congestion control behavior, while the fixed-window scheme provides the recovery-free reference point.

Table 1. Simulation topology and link parameters.

Parameter	Value

Topology	Dumbbell (sender — R1 — R2 — receiver)
Edge links	100 Mbps, 2 ms one-way delay
Bottleneck, forward (data)	10 Mbps, 50 ms one-way delay
Bottleneck, reverse (ACK) — swept	10 / 1 / 0.2 / 0.1 / 0.033 Mbps, 50 ms delay
Forward queue (drop-tail)	100 packets
Reverse queue (drop-tail)	25 packets
Segment size	1448 B
Delayed ACK (DelAckCount)	2 (one ACK per two segments); baseline sweep: 1 vs. 2
CCAs	Fixed-window / NewReno / CUBIC / BBRv1
Loss recovery	PRR, SACK enabled (ns-3 defaults)
ns-3 release	3.47
Duration	120 s per scenario

4. Results and discussion

4.1. Baseline validation

The first section establishes the baseline evaluation, in which the fixed-window scheme and the three CCAs are assessed under a symmetric 10 Mbps bottleneck using the topology parameters of Section 3, with two ACK stream configurations (DelAckCount = 1 vs. 2, i.e., ACK every segment vs. every second segment). The goal of this scenario is to evaluate the impact of the delayed ACK mechanism in isolation, before bandwidth asymmetry is introduced next. Under a symmetric bottleneck link, the results are influenced solely by the data path parameters (the 10 Mbps bottleneck link and 100-packet queue) and the delayed ACK setting, since the reverse path introduces no loss or queuing. With symmetric link capacities, enabling delayed ACKs (DelAckCount = 2) simply halves the ACK rate relative to per-segment acknowledgment, without any further thinning or compression of the ACK stream.

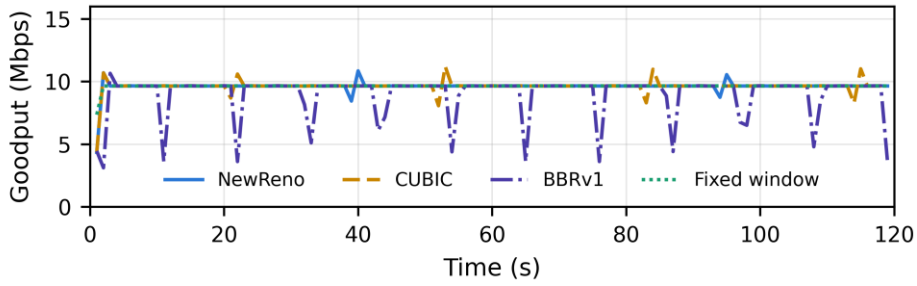
Figure 1 shows goodput samples every 1 second for the duration of the simulation; Figure 2 shows in-flight data over time; and Figure 3 shows bottleneck queue occupancy against the 100-packet queue limit. In each figure, panel (a) corresponds to DelAckCount = 1 (per-segment ACKing) and panel (b) to DelAckCount = 2. Tables 2 and 3 summarize losses, RTOs, spurious retransmissions (Spurious retx), and average goodput for the startup phase (0–20 s) and steady state (20–120 s), respectively. We count a retransmission as spurious when its sequence range overlaps with data already delivered to the receiver at the transport layer.

As seen from the summarized results in Tables 2 and 3, the delayed ACK setting does not affect the fixed-window sender. In fact, both configurations produce identical results, with 0 losses, 0 RTOs, and an average goodput of 9.526 Mbps during startup and 9.640 Mbps during steady state. ACK arrivals do not impact the sending rate, since it is fixed for the duration of the connection. Therefore, they only update the sender's picture of packet delivery status. This confirms that any differences observed in the remaining CCAs (NewReno, CUBIC, and BBRv1) are attributable to how each algorithm reacts to ACK arrivals, not to the underlying ACK mechanism itself disturbing the sender’s sending rate.

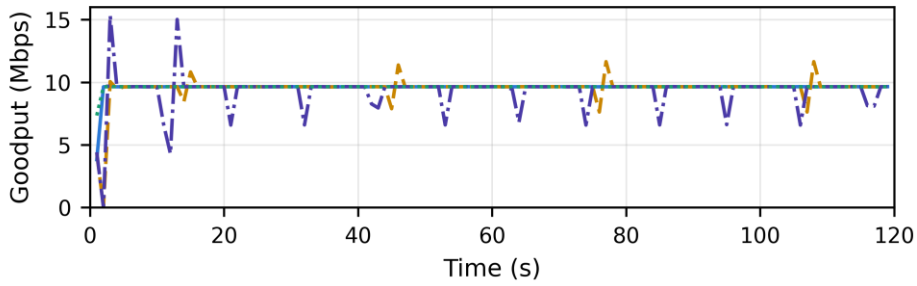
Both NewReno and CUBIC converge to a nearly identical steady-state goodput of around 9.6 Mbps under both ACK configurations, as shown in Figure 1 and Table 3. However, the differences between configurations appear in the startup phase and in the queue dynamics. For NewReno, delayed ACKs slow the AIMD cycle, so the queue fills approximately 3 times over the steady-state interval (roughly once every 36 s), compared to 6 times under DelAckCount = 1, as shown in Figures 2 and 3. This is mirrored exactly in the steady-state loss count of 3 versus 6 packets, that is, one drop per cycle, as seen in Table 3. Although the two configurations achieve nearly identical steady-state goodput (9.640 Mbps versus 9.641 Mbps), the faster window growth under DelAckCount = 1 during slow start delivered slightly more bytes for the 120 s run (144.08 MB versus 143.88 MB). However, this was at the cost of twice the startup losses (194 versus 97), as seen in Table 2. Overall, the conservative behavior of NewReno's window growth and PRR recovery resulted in zero RTO events and zero spurious retransmissions, as seen in Tables 2 and 3.

For CUBIC, the delayed ACK setting mainly impacted the startup phase of the connection. Under DelAckCount = 1, CUBIC startup achieved the same average goodput as NewReno (9.427 Mbps versus

9.428 Mbps) with zero RTOs. However, CUBIC experienced fewer packet drops than NewReno (74 versus 194 losses), as seen in Table 2. This is because HyStart's delay signal exits slow start early, at roughly 250 segments, before the loss feedback reaches the sender. Nonetheless, CUBIC experienced two spurious retransmissions while NewReno experienced none. Under DelAckCount = 2, the halved ACK stream weakens both of HyStart's exit signals, the delay signal and the ACK-train signal. This prevents HyStart's exit from triggering, causing CUBIC to spike beyond 350 segments (Figure 2(b)). The result is a mass drop of packets and the only RTO CUBIC experienced under either DelAckCount setting, as shown in Table 2. This single RTO event caused CUBIC's startup goodput to drop by 5.5% compared to per-segment ACKing (8.906 Mbps versus 9.427 Mbps). Beyond startup, the two configurations are nearly identical, as with NewReno, achieving a steady goodput of 9.639 Mbps, as seen in Table 3. However, CUBIC's window growth function fills and drains the queue more frequently than NewReno's, producing 111 and 119 steady-state losses (DelAckCount = 1 and 2, respectively) across roughly 11 loss episodes over the run as seen in Figure 3, compared to NewReno's 6 and 3 packet drops. Each episode leaves multiple holes in the flight window, and until the sender recovers them, in-order delivery at the receiver stalls for a few RTTs. This results in around 10 spurious retransmissions per configuration, and produces the characteristic freeze-then-burst pattern in delivered goodput, as shown in Figure 1. Overall, CUBIC delivered 144.06 MB under per-segment ACKing and 142.76 MB under delayed ACKs.



(a) No delayed ACK (DelAckCount = 1)



(b) Delayed ACK (DelAckCount = 2)

Figure 1. Goodput over time in the baseline scenario (10 Mbps reverse link). Goodput is sampled over 1 s intervals.

BBRv1 shows the opposite trade-off to the loss-based CCAs as DelAckCount = 2 incurred more losses during startup but improves its steady-state goodput. During startup, DelAckCount = 2 cost BBRv1 an extra RTO (2 versus 1) and 55% more drops (237 versus 153), as seen in Table 2. This is because the full-pipe detector reacts a round later on clumped ACKs. In steady state, however, DelAckCount = 2 achieved an average goodput of 9.335 Mbps, compared to 9.037 Mbps under DelAckCount = 1, around 3.3% higher, and delivered more bytes over the full 120-second run (139.26 MB versus 134.97 MB). Notably, BBRv1 is the only CCA that failed to reach the ~9.6 Mbps plateau of the other three in either configuration, and the only one whose steady-state goodput depends on the ACK configuration. This is because an ACK covering two segments can produce inflated rate samples, with bytes delivered in an adjacent measurement interval. Such inflation would average out under most filters, but the BtlBw max filter latches onto the highest rate sample. This causes BBRv1 to pace slightly above the bottleneck's service rate. The excess bytes accumulate in the queue each round, amounting to around 75 segments per epoch, as seen in Figure 3. This backlog remains below the 100-packet queue limit, which is why the overshoot inflates queueing delay without

causing loss. PROBE_RTT, however, remains uncorrupted by the delayed ACK setting. As seen in Figure 3, the sender shrinks its window to 4 segments and drains the backlog and then returns the in-flight data to slightly above the BDP of the bottleneck (Figure 2). This inflation of rate samples is a known weakness of the BBRv1 estimator [19, 34], although it has been studied mostly in Wi-Fi networks, where ACKs can be compressed or released in batches. Our results show that the same mechanism can be excited by the delayed ACK setting alone, over a clean symmetric path. Overall, BBRv1 recorded zero losses, zero RTOs, and zero spurious retransmissions during steady state in both configurations, as shown in Table 3.

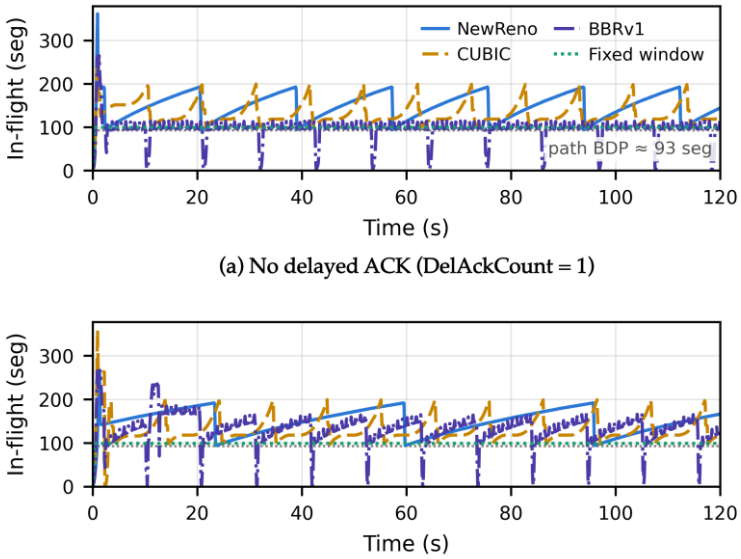


Figure 2. Data in-flight, in segments in the baseline scenario (10 Mbps reverse link). The dotted line marks the path BDP of ≈ 93 segments.

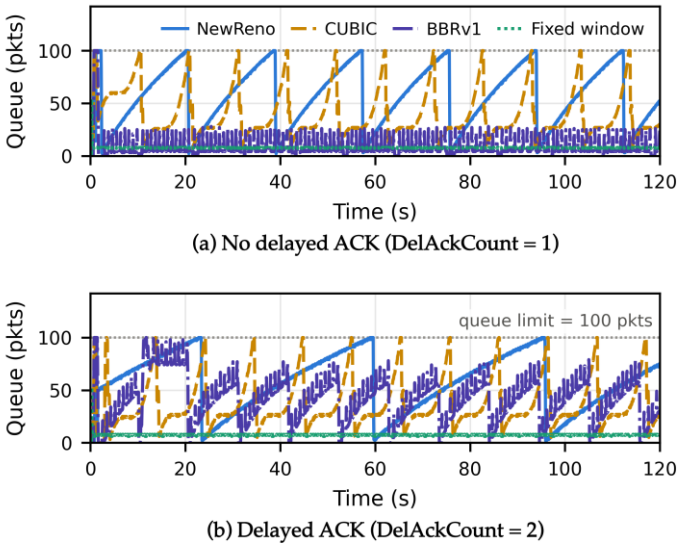


Figure 3. Bottleneck queue occupancy in the baseline scenario (10 Mbps reverse link). The dotted line marks the 100-packet queue limit.

Table 2. Startup-phase metrics (0–20 s) for each CCA and delayed-ACK setting.

CCA	DelAckCount	Losses (segment)	RTOs	Spurious retx	Avg goodput (Mbps)[1]
NewReno	1	194	0	0	9.428
NewReno	2	97	0	0	9.349
CUBIC	1	74	0	2	9.427

CCA	DelAckCount	Losses (segment)	RTOs	Spurious retx	Avg goodput (Mbps)[1]
CUBIC	2	290	1	2	8.906
BBRv1	1	153	1	0	8.803
BBRv1	2	237	2	0	9.030
Fixed window	1	0	0	0	9.526
Fixed window	2	0	0	0	9.526

[1] Goodput is the application bytes delivered divided by the window duration. Windows starting at 0 s include connection setup (≈0.2 s).

Table 3. Steady-state metrics (20–120 s) for each CCA and delayed-ACK setting.

CCA	DelAckCount	Losses (segment)	RTOs	Spurious retx	Avg goodput (Mbps)
NewReno	1	6	0	0	9.640
NewReno	2	3	0	0	9.641
CUBIC	1	111	0	10	9.639
CUBIC	2	119	0	10	9.639
BBRv1	1	0	0	0	9.037
BBRv1	2	0	0	0	9.335
Fixed window	1	0	0	0	9.640
Fixed window	2	0	0	0	9.640

4.2. Bandwidth asymmetry sweep

Having established the baseline behavior under symmetric links, this section sweeps the reverse-path bandwidth from 10 Mbps down to 33 kbps under DelAckCount = 2, the default Linux configuration. Figure 4 shows the average goodput over the steady-state window (20–120 s) for all four configurations, while Table 4 reports whole-run metrics (0–120 s). Under the fixed-window scheme, the average goodput remains unchanged until the reverse path becomes severely constrained (100:1 and 300:1). At a 100 kbps reverse link it achieved an average goodput of 5.36 Mbps, the lowest across all configurations at this rate, and at 33 kbps it achieved 2.51 Mbps. This degradation stems mainly from the ACK clock. The fixed-window scheme triggered no forward-path losses and no RTOs, yet its sending rate is throttled by the ACK arrival rate, as shown in Figures 4 and 5.

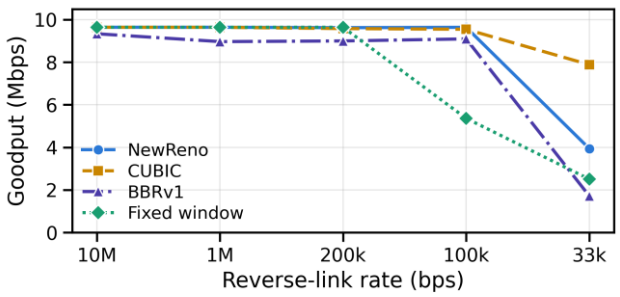


Figure 4. Average goodput (20–120 s) versus reverse-link rate for each congestion control algorithm including the fixed-window baseline.

Once the connection starts, the fixed-window sender transmits its full window of 100 segments. Since the forward path is set to 10 Mbps, it delivers the 100 segments in around 0.12 s, plus the propagation delay of the access links. With DelAckCount = 2, the receiver generates one ACK for every two segments, so the burst results in 50 ACK packets. Under a 100 kbps reverse link and around 54-byte ACK size, the reverse link can carry at most around 231 ACKs/s. With each ACK releasing two segments of 1448 bytes, this caps goodput at around 5.36 Mbps (ACK rate × segments per ACK × segment size), and the measured goodput matches this ceiling to within 1% as shown in Table 4. The ACKs the burst generates beyond this drain rate are buffered at the reverse queue, and these buffered ACKs then drive the sender: each ACK that leaves the queue releases two segments, which generate exactly one new ACK to take its place. This forms a closed loop of ACK-driven transmission at the rate of the reverse link, with a standing reverse queue that persists for the connection's lifetime. However, at 33 kbps this loop no longer holds. The reverse queue cannot absorb the ACKs generated by the window, so ACKs are dropped (Figure 5(b)) and the surviving cumulative ACKs each cover more than two segments, lifting goodput to 2.51 Mbps against the 1.77 Mbps ceiling that two-segment ACKs would impose. This acts as our reference case for the other CCAs as the reverse queue is unsaturated, it isolates the pure ACK-clock effect at a fixed window of 100 segments, so any deviation by a real CCA, above or below, is attributable to its window dynamics rather than to the ACK path itself. Once the reverse queue saturates, ACK losses will thin the acknowledgment stream, and each surviving cumulative ACK acknowledges a larger block of data. Such thinned ACK streams are known to alter CCA dynamics, producing burstier transmission [15, 25], which we observe for the adaptive CCAs next.

Table 4. Whole-run metrics (0–120 s) across the reverse-path sweep.

CCA	Reverse rate	Ratio	Goodput (Mbps)	Losses (segment)	RTOs	Spurious retx
NewReno	10 Mbps	1:1	9.59	100	0	0
	1 Mbps	10:1	9.59	100	0	0
	200 kbps	50:1	9.57	100	0	0
	100 kbps	100:1	9.55	82	0	0
	33 kbps	300:1	4.00	44	1	45
CUBIC	10 Mbps	1:1	9.52	409	1	12
	1 Mbps	10:1	9.52	409	1	12
	200 kbps	50:1	9.02	300	1	1
	100 kbps	100:1	9.51	161	0	0
	33 kbps	300:1	7.75	330	8	742
BBRv1	10 Mbps	1:1	9.28	237	2	0
	1 Mbps	10:1	8.94	158	1	0
	200 kbps	50:1	8.97	242	1	0
	100 kbps	100:1	9.08	0	0	0
	33 kbps	300:1	1.72	0	0	0
Fixed window	10 Mbps	1:1	9.62	0	0	0
	1 Mbps	10:1	9.62	0	0	0
	200 kbps	50:1	9.62	0	0	0
	100 kbps	100:1	5.36	0	0	0

CCA	Reverse rate	Ratio	Goodput (Mbps)	Losses (segment)	RTOs	Spurious retx
	33 kbps	300:1	2.51	0	0	0

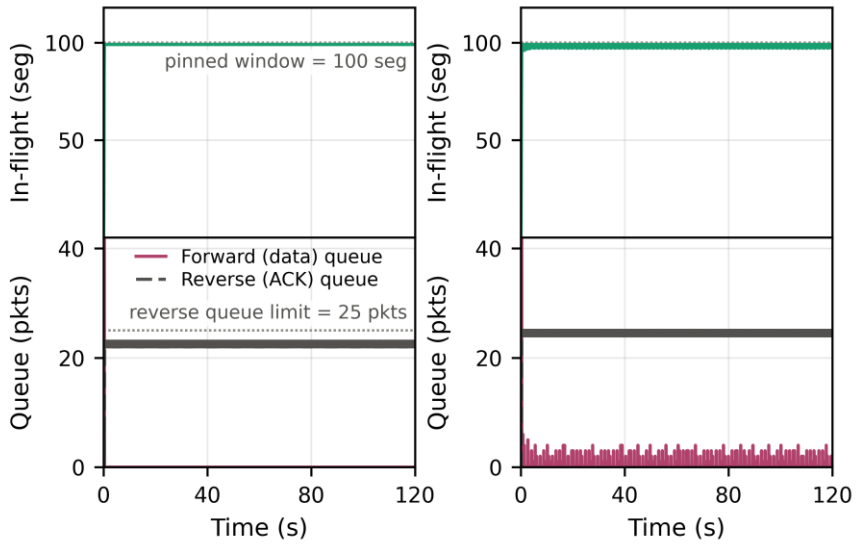


Figure 5. Fixed-window under a (a) 100 kbps and (b) 33 kbps reverse link: (top) segments in-flight, (bottom) forward and reverse queue occupancy.

NewReno achieved nearly identical average goodput, around 9.5 Mbps, across most of the reverse-path sweep, dropping only at 33 kbps to 4 Mbps, still 59% higher than the fixed-window scheme's 2.51 Mbps. Unlike in the fixed-window scheme, each ACK in NewReno increases the congestion window and therefore the sending rate. At 100 kbps, the reverse queue saturates at 25 packets, so ACKs return late and thinned, and the window grows only as fast as the reverse link can deliver acknowledgments. This slows window growth but not the steady-state goodput, as the window is already large enough to keep the forward link saturated. Instead, the thinned ACK stream only prolongs each AIMD cycle, filling the forward queue slowly, as shown in Figure 6(a). The flow, therefore, sustains 9.5 Mbps for the rest of the run. NewReno collapsed only at 33 kbps, achieving 4 Mbps with a single RTO and 45 spurious retransmissions. This is because after slow start overshoots and the sender enters fast recovery, it is unable to complete recovery over the severely constrained reverse link, triggering the RTO and resetting the congestion window (cwnd) to 1. The sender then regrows cwnd through slow start until it reaches the reduced slow-start threshold, at which point it enters congestion avoidance. From there, the window still increases, but only as fast as the thinned ACK stream permits. At the same time, the reverse ACK queue inflates the RTT to around 470 ms, which raises the window required to fill the pipe to roughly 390 segments, while cwnd only reaches around 180 by the end of the run, as shown in Figure 6(b).

CUBIC achieved nearly identical average goodput to NewReno, around 9.5 Mbps, up to ~6% difference in favor of NewReno across the 10 Mbps, 1 Mbps, 200 kbps, and 100 kbps sweep points. However, CUBIC suffered between 161 and 409 losses and at most one RTO per run at these points, with the majority of losses occurring during startup at all but the 100 kbps point (Figure 7(a)). As in the baseline scenario, CUBIC fails to exit slow start through HyStart [13] before overflowing the forward queue. At 33 kbps, CUBIC achieved the highest average goodput among all four configurations, reaching 7.75 Mbps as seen in Figure 7(b). This is a direct result of CUBIC's window growth function, which increases the window as a function of time since the last loss rather than per ACK arrival as in NewReno. The growth schedule is therefore largely insensitive to ACK thinning and to the inflated RTT (up to 480 ms), allowing CUBIC to repeatedly refill the forward queue, roughly every 15 s, where NewReno could not refill it even once after its loss episode. Nevertheless, since the reverse queue is saturated at 25 packets, ACK drops starve the

ACK clock and cripple loss recovery, triggering an RTO on every epoch, 8 in total. The result is 742 spurious retransmissions, attributed primarily to RTO recovery as shown in Table 4.

BBRv1 achieved a lower average goodput than NewReno and CUBIC, ranging from 8.94 to 9.28 Mbps across the 10 Mbps, 1 Mbps, 200 kbps, and 100 kbps sweep points, and lower than the fixed-window across the 10 Mbps, 1 Mbps, and 200 kbps points. Under 100 kbps, BBRv1 achieved an average goodput of 9.08 Mbps. This is, in fact, the result of the inflated rate samples we observed in the baseline scenario. Here, however, the aggregation is compounded by ACK loss as the saturated reverse queue drops many ACKs, each surviving ACK cumulatively acknowledges the segments covered by the lost ones. These aggregated ACKs inflate individual delivery-rate samples, causing the BtlBw max filter to latch onto them. Unlike the baseline scenario, the inflated estimate does not build a queue at the bottleneck. This is because ACKs return slowly, so unacknowledged data quickly reaches the sending limit set by the target cwnd, and the sender can only transmit when an ACK arrives and frees space in the window. Once BBRv1 has injected a full window of data into the network, its sending is no longer driven by the pacer but clocked by returning ACKs, just like the fixed-window behavior.

Unlike the 100 kbps scenario, under 33 kbps the reverse link did not drop any ACKs, and therefore the phenomenon where aggregated ACKs inflate BBRv1's rate samples does not occur. This is a result of BBRv1's own in-flight cap, as the ACK rate falls, so does the BtlBw estimate, along with the bound on data in-flight. The throttled sending rate produces an ACK stream slow enough that the reverse queue never approaches saturation, hovering around 10 of its 25 packets, as seen in Figure 8(b). Since BBRv1's delivery-rate samples are computed from ACK arrivals, and no sample inflation occurs, BBRv1 converges onto the ACK-clock ceiling of the reverse link, reaching 1.72 Mbps against a ceiling of 1.77 Mbps. Overall, at 33 kbps BBRv1 achieved the lowest average goodput among all four configurations, as the only one that never escaped the ceiling through ACK thinning.

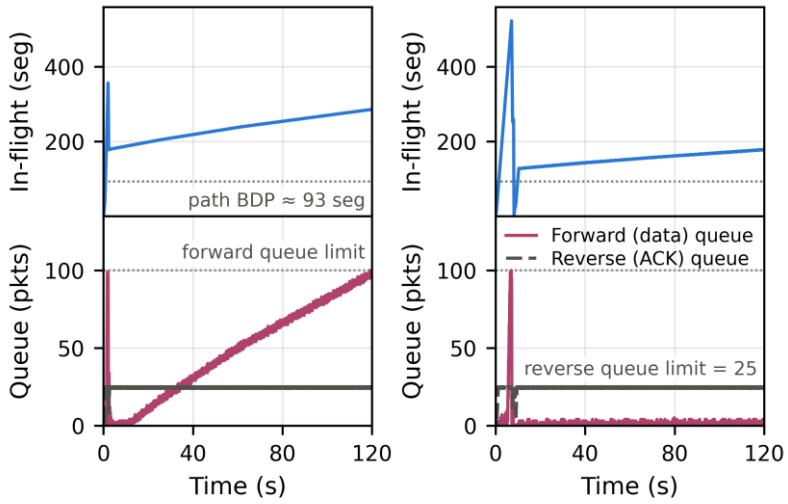


Figure 6. NewReno under a (a) 100 kbps and (b) 33 kbps reverse link: (top) segments in-flight, (bottom) forward and reverse queue occupancy.

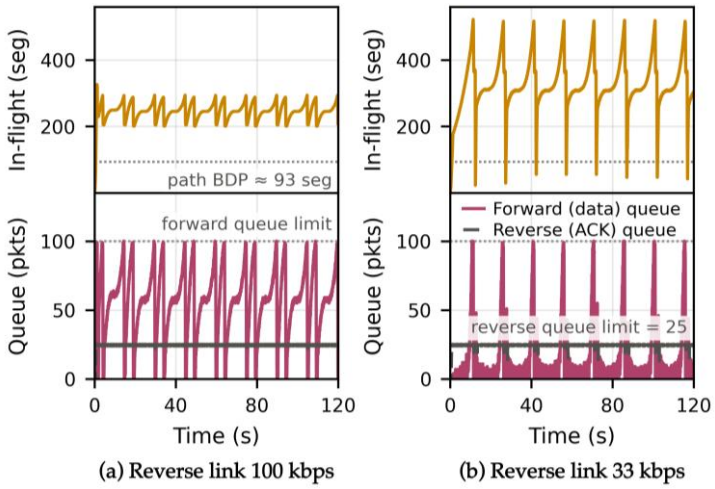


Figure 7. CUBIC under a (a) 100 kbps and (b) 33 kbps reverse link: (top) segments in-flight, (bottom) forward and reverse queue occupancy.

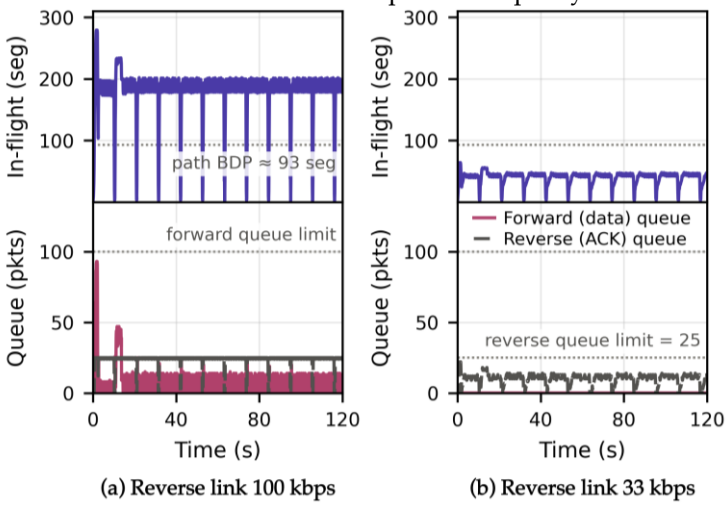


Figure 8. BBRv1 under a (a) 100 kbps and (b) 33 kbps reverse link: (top) segments in-flight, (bottom) forward and reverse queue occupancy.

While a minimum of 4 Mbps is recommended for a solo primary care practice, including standard-definition video consultation, 10 Mbps is the minimum recommendation for rural health clinics [35]. Under mild asymmetry, all three CCAs sustain goodput near this level. However, under severe asymmetry (300:1), only CUBIC approaches the rural clinics' recommended threshold, achieving an average goodput of 7.75 Mbps. CUBIC still exceeded the solo-practice recommendation, while NewReno sat at the threshold with 4.00 Mbps. However, throughput alone does not determine service quality. CUBIC achieved the highest goodput by saturating the forward queue, resulting in repeated RTOs and inflated delays, both of which can impact the quality of interactive consultation services. BBRv1 reduced the forward delay but also reduced throughput, achieving an average goodput of 1.72 Mbps, below either recommendation [35]. Nonetheless, the minimum achieved forward capacity across CCAs far exceeds the rate needed to deliver voice-based consultations. Similarly, vital data monitoring, such as ECG and blood pressure monitoring, requires only up to 20 kbps [36], though such data travel on the constrained reverse path rather than the forward path measured here.

5. Conclusions

Motivated by the limitations of rural access technologies for telemedicine, this paper revisited the bandwidth asymmetry problem and evaluated its impact on three CCAs, NewReno, CUBIC, and BBRv1, using the ns-3.47 simulator. In addition, a fixed-window sender scheme was used to serve as a reference case, isolating the pure ACK-clock effect and matching the derived ceiling to within 1% at 100:1 and exceeding it at 300:1 only through ACK thinning, so any deviation by the CCAs is attributable to their window dynamics rather than the ACK path itself. All three CCAs tolerated low-to-moderate asymmetry (up

to 100:1) with minimal goodput loss. Both NewReno and CUBIC sustained between 9.0 and 9.6 Mbps, while BBRv1 achieved around 8.9 to 9.1 Mbps across the 10:1 to 100:1 sweep points. However, under severe asymmetry (300:1), CUBIC achieved the highest average goodput at 7.75 Mbps. This is credited to its window growth function, which increases the window as a function of time and therefore makes it largely insensitive to ACK thinning and inflated RTT. NewReno, on the other hand, grows its window per ACK arrival through AIMD, which makes it vulnerable to the thinned ACK stream and inflated RTT. As a result, NewReno achieved an average goodput of 4 Mbps. BBRv1 achieved only 1.72 Mbps because its target congestion window throttled its sending rate, leaving it to operate under the reverse link's ACK-clock ceiling. Beyond the asymmetry sweep, the results show that even with symmetric links, BBRv1 is affected by delayed ACKs, which inflate its delivery-rate samples, resulting in a steady-state goodput of 9.335 Mbps versus 9.037 Mbps under per-segment ACKing. In telemedicine terms, none of the CCAs sustained the 10 Mbps recommended for rural health clinics under severe asymmetry, and only CUBIC approached it. Both CUBIC and NewReno remained at or above the 4 Mbps solo-practice recommendation, while BBRv1 fell below either recommendation. Nonetheless, even the lowest achieved goodput far exceeds the rate needed for voice-based consultations.

The evaluation in this paper used a single flow over one dumbbell topology, with the sweep conducted under the default delayed ACK setting. Therefore, the findings should be read within the limits of our experimental design. Future work will address these limits by extending the evaluation to more network settings and to newer BBR versions such as BBRv2 and BBRv3. In addition, future work will investigate modern transport protocols such as QUIC, which incorporates different reliability and acknowledgment mechanisms from TCP. Finally, these results motivate receiver-side acknowledgment strategies that could mitigate some of the asymmetry effects observed in TCP.

Supplementary Materials: Not applicable.

Funding: This research received no external funding.

Acknowledgments: During the preparation of this manuscript, the authors used Claude Opus 4.8 (Anthropic) to improve the language and clarity of the text. The authors reviewed and edited all output and take full responsibility for the content of the publication.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sri-Ganeshan, M.; Cameron, P. Remote Monitoring in Telehealth: Advancements, Feasibility and Implications. In A Comprehensive Overview of Telemedicine; IntechOpen: London, UK, 2024. https://doi.org/10.5772/intechopen.1004661.
2. Meda, G.V.; Brennan-Davies, A.H. Remote Patient Monitoring and the Need for a New Care Model: A Narrative Review of Implementation Challenges. Cureus 2026, 18, e102803. https://doi.org/10.7759/cureus.102803.
3. Sornlertlamvanich, P.; Phakaket, C.; Hantula, P.; Kanjanawattana, S.; Kaoungku, N.; Srivisut, K. Integration of Cloud-Based Central Telemedicine System and IoT Device Networks. Computers 2025, 14, 357. https://doi.org/10.3390/computers14090357.
4. Rao, S.P.; Jayant, N.S.; Stachura, M.E.; Astapova, E.; Pearson-Shaver, A. Delivering Diagnostic Quality Video over Mobile Wireless Networks for Telemedicine. Int. J. Telemed. Appl. 2009, 2009, 406753. https://doi.org/10.1155/2009/406753.
5. Totten, A.M.; Womack, D.M.; Griffin, J.C.; McDonagh, M.S.; Davis-O'Reilly, C.; Blazina, I.; Grusing, S.; Elder, N. Telehealth-Guided Provider-to-Provider Communication to Improve Rural Health: A Systematic Review. J. Telemed. Telecare 2024, 30, 1209–1229. https://doi.org/10.1177/1357633X221139892.
6. Cummins, M.R.; Wong, B.; Wan, N.; Han, J.; Shishupal, S.D.; Gouripeddi, R.; Ivanova, J.; Franklin, A.; Johnny, J.; Ong, T.; et al. Social Vulnerability, Lower Broadband Internet Access, and Rurality Associated with Lower Telemedicine Use in U.S. Counties. JAMIA Open 2025, 8, ooaf056. https://doi.org/10.1093/jamiaopen/ooaf056.
7. Chiha, A.; Van der Wee, M.; Colle, D.; Verbrugge, S. Techno-Economic Viability of Integrating Satellite Communication in 4G Networks to Bridge the Broadband Digital Divide. Telecommun. Policy 2020, 44, 101874. https://doi.org/10.1016/j.telpol.2019.101874.
8. Mohan, N.; Ferguson, A.E.; Cech, H.; Renatin, P.R.; Bose, R.; Marina, M.K.; Ott, J. A Multifaceted Look at Starlink Performance. In Proceedings of the ACM Web Conference 2024 (WWW '24), Singapore, 13–17 May 2024; pp. 2723–2734. https://doi.org/10.1145/3589334.3645328.
9. Garcia, J.; Sundberg, S.; Caso, G.; Brunstrom, A. Multi-Timescale Evaluation of Starlink Throughput. In Proceedings of the 1st ACM Workshop on LEO Networking and Communication (LEO-NET '23), Madrid, Spain, 6 October 2023; pp. 31–36.
10. Lai, W.-P.; Chen, W.-R.; Lai, H.-L.; Li, H.-Y. Impacts of 5G-TDD Time Slot Configurations on the Downlink and Uplink Data Rates. In Proceedings of the 2023 Asia Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC), Taipei, Taiwan, 31 October–3 November 2023; pp. 2149–2154. https://doi.org/10.1109/APSIPAASC58517.2023.10317479.
11. International Telecommunication Union. Asymmetric Digital Subscriber Line (ADSL) Transceivers; ITU-T Recommendation G.992.1; ITU: Geneva, Switzerland, 1999.
12. Henderson, T.; Floyd, S.; Gurtov, A.; Nishida, Y. The NewReno Modification to TCP's Fast Recovery Algorithm; RFC 6582; Internet Engineering Task Force: Fremont, CA, USA, 2012. https://doi.org/10.17487/RFC6582.
13. Xu, L.; Ha, S.; Rhee, I.; Goel, V.; Eggert, L., Ed. CUBIC for Fast and Long-Distance Networks; RFC 9438; Internet Engineering Task Force: Fremont, CA, USA, 2023. https://doi.org/10.17487/RFC9438.
14. Cardwell, N.; Cheng, Y.; Gunn, C.S.; Yeganeh, S.H.; Jacobson, V. BBR: Congestion-Based Congestion Control. Commun. ACM 2017, 60, 58–66. https://doi.org/10.1145/3009824.
15. Balakrishnan, H.; Padmanabhan, V.N.; Katz, R.H. The Effects of Asymmetry on TCP Performance. Mob. Netw. Appl. 1999, 4, 219–241. https://doi.org/10.1023/A:1019155000496.
16. Lakshman, T.V.; Madhow, U.; Suter, B. Window-Based Error Recovery and Flow Control with a Slow Acknowledgment Channel: A Study of TCP/IP Performance. In Proceedings of the IEEE INFOCOM '97, Kobe, Japan, 7–11 April 1997; Volume 3, pp. 1199–1209. https://doi.org/10.1109/INFCOM.1997.631144.
17. Kalampoukas, L.; Varma, A.; Ramakrishnan, K.K. Improving TCP Throughput over Two-Way Asymmetric Links: Analysis and Solutions. In Proceedings of the 1998 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, Madison, WI, USA, 22–26 June 1998; pp. 78–89. https://doi.org/10.1145/277851.277877.
18. Fairhurst, G.; Samaraweera, N.K.G.; Sooriyabandara, M.; Harun, H.; Hodson, K.; Donadio, R. Performance Issues in Asymmetric TCP Service Provision Using Broadband Satellite. IEE Proc. Commun. 2001, 148, 95–99. https://doi.org/10.1049/ip-com:20010130.

19. Grazia, C.A.; Patriciello, N.; Klapez, M.; Casoni, M. BBR+: Improving TCP BBR Performance over WLAN. In Proceedings of the IEEE International Conference on Communications (ICC 2020), Dublin, Ireland, 7–11 June 2020; pp. 1–6. https://doi.org/10.1109/ICC40277.2020.9149220.
20. Garcia, J.; Sundberg, S.; Brunstrom, A. TCP Congestion Control Performance over Starlink. In Proceedings of the 2025 Applied Networking Research Workshop (ANRW '25), Madrid, Spain, July 2025. https://doi.org/10.1145/3744200.3744760.
21. Lin, J.; Cui, L.; Zhang, Y.; Tso, F.P.; Guan, Q. Extensive Evaluation on the Performance and Behaviour of TCP Congestion Control Protocols under Varied Network Scenarios. Comput. Netw. 2019, 163, 106872. https://doi.org/10.1016/j.comnet.2019.106872.
22. nsnam. ns-3 Network Simulator, Release 3.47; 2026. Available online: https://www.nsnam.org/releases/ns-3-47/ (accessed on 6 July 2026).
23. Albalawi, A. An Efficient Receiver-Driven Automatic Repeat Request (RDQ) for Transport Protocols on the Internet. Electronics 2026, 15, 802. https://doi.org/10.3390/electronics15040802.
24. Bacco, M.; De Cola, T.; Giambene, G.; Gotta, A. M2M Traffic via Random Access Satellite Links: Interactions between Transport and MAC Layers. IEEE Trans. Aerosp. Electron. Syst. 2019, 55, 846–863. https://doi.org/10.1109/TAES.2018.2865150.
25. Balakrishnan, H.; Padmanabhan, V.N.; Fairhurst, G.; Sooriyabandara, M. TCP Performance Implications of Network Path Asymmetry; RFC 3449 (BCP 69); Internet Engineering Task Force: Fremont, CA, USA, 2002. https://doi.org/10.17487/RFC3449.
26. Louati, F.; Barakat, C.; Dabbous, W. Handling Two-Way TCP Traffic in Asymmetric Networks. In High Speed Networks and Multimedia Communications (HSNMC 2004); Lecture Notes in Computer Science, Vol. 3079; Springer: Berlin/Heidelberg, Germany, 2004; pp. 233–243. https://doi.org/10.1007/978-3-540-25969-5_21.
27. Park, J.; Park, D.; Hong, S.; Park, J. Preventing TCP Performance Interference on Asymmetric Links Using ACKs-First Variable-Size Queuing. Comput. Commun. 2011, 34, 730–742. https://doi.org/10.1016/j.comcom.2010.09.010.
28. Li, F.; Chung, J.W.; Jiang, X.; Claypool, M. TCP CUBIC versus BBR on the Highway. In Passive and Active Measurement, Proceedings of the 19th International Conference, PAM 2018, Berlin, Germany, 26–27 March 2018; Beverly, R., Smaragdakis, G., Feldmann, A., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2018; Volume 10771, pp. 269–280. https://doi.org/10.1007/978-3-319-76481-8_20.
29. Bless, R.; Lihotzki, L.; Zitterbart, M. Insights into BBRv3's Performance and Behavior by Experimental Evaluation. In Proceedings of the 2025 IFIP Networking Conference, Limassol, Cyprus, 26–29 May 2025; IEEE. Available online: https://publikationen.bibliothek.kit.edu/1000182022.
30. Claypool, M.; Chung, J.W.; Li, F. BBR': An Implementation of Bottleneck Bandwidth and Round-Trip Time Congestion Control for ns-3. In Proceedings of the 10th Workshop on ns-3 (WNS3 2018), Surathkal, India, 13–14 June 2018. https://doi.org/10.1145/3199902.3199903.
31. Jain, V.; Mittal, V.; Tahiliani, M.P. Design and Implementation of TCP BBR in ns-3. In Proceedings of the 10th Workshop on ns-3 (WNS3 2018), Surathkal, India, 13–14 June 2018; pp. 16–22. https://doi.org/10.1145/3199902.3199911.
32. Piotrowska, A. Performance Evaluation of TCP BBRv3 in Networks with Multiple Round Trip Times. Appl. Sci. 2024, 14, 5053. https://doi.org/10.3390/app14125053.
33. Abrol, A.; Murali Mohan, P.; Truong-Huu, T.; Gurusamy, M. BBR Congestion Control Algorithms: Evolution, Challenges and Future Directions. ACM Comput. Surv. 2026, 58, Article 235, 1–36. https://doi.org/10.1145/3793537.
34. Su, B.; Jiang, X.; Jin, G.; Chen, H. Rethinking the Rate Estimation of BBR Congestion Control. Electron. Lett. 2020, 56, 239-241. https://doi.org/10.1049/el.2018.6440.
35. Federal Communications Commission. Health Care Broadband in America: Early Analysis and a Path Forward; OBI Technical Paper No. 5; FCC: Washington, DC, USA, 2010. Available online: https://transition.fcc.gov/national-broadband-plan/health-care-broadband-in-america-paper.pdf (accessed on 20 July 2026).
36. Aleksic, S.; Atanasov, M.; Agius, J.C.; Camilleri, K.; Čartolovni, A.; Climent-Pérez, P.; Colantonio, S.; Cristina, S.; Despotović, V.; Ekenel, H.K.; et al. State of the Art of Audio- and Video-Based Solutions for AAL. arXiv 2022, arXiv:2207.01487. https://doi.org/10.48550/arXiv.2207.01487.