Investigating the Impact of ACK Aggregation on TCP Performance using ns-3

Evaluation of Transport and MAC-Layer Aggregation Techniques

Hanna Heinczinger[1]

Supervisor(s): Fernando Kuipers[1]**, Adrian Zapletal**[1]

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 15, 2025

Name of the student: Hanna Heinczinger
Final project course: CSE3000 Research Project
Thesis committee: Fernando Kuipers, Adrian Zapletal, Asterios Katsifodimos

Abstract

Modern TCP congestion control algorithms rely on timely ACK feedback to adjust their parameters. However, some networks deliberately suppress ACKs. This study uses the ns-3 simulator to experiment with the impact of suppressing ACKs on the reverse path on four TCP variants (BBRv3, Cubic, NewReno, Vegas) in both wired and wireless dumbbell topologies. In wired experiments, we implement a custom queue that allows us to test fixed aggregation ratios at the transport-layer. In wireless scenarios, we utilize the IEEE 802.11n standard's MAC-layer aggregation schemes (A-MSDU, A-MPDU), both individually and in combination, and also evaluate varying maximum A-MPDU aggregation sizes. Our results show that while transport-layer aggregation can degrade performance and fairness, especially for BBRv3 and Cubic, MAC-layer aggregation consistently improves throughput without destabilizing TCP behaviour. NewReno demonstrates strong resilience across both setups, while Vegas exhibits highly inconsistent performance. These findings highlight the importance of aligning aggregation mechanisms with congestion control strategies carefully.

1 Introduction

The Transmission Control Protocol [7] is the main transport protocol used in the Internet, ensuring reliable and ordered delivery of data across various network environments. TCP's reliability and congestion control mechanisms depend on acknowledgements travelling across the reverse path. When ACK feedback is suppressed, senders may misinterpret network conditions, confusing increased delays for losses or missing timely congestion signals. This can lead to suboptimal congestion window evolution, increased timeouts, and ultimately throughput degradation or collapse.

With the rise of mobile and wireless networks, deliberate ACK aggregation has become an important research topic. In particular, the IEEE 802.11n MAC sublayer supports two forms of frame aggregation [8], A-MSDU (Aggregated MAC Service Data Unit) and A-MPDU (Aggregated MAC Protocol Data Unit), to increase throughput by reducing per-frame overhead. These MAC-layer mechanisms affect the timing and delivery of TCP ACKs, altering the sender's feedback loop.

This study systematically analyses the behaviour of four distinct TCP algorithms under ACK suppression in two different setups. The selection of BBRv3, Cubic, Vegas and NewReno is deliberate, given the diverse approaches they use to manage network congestion [18]. In the wired setup, a custom queue is employed to precisely adjust transport-layer ACK aggregation rates, allowing us to assess its impact on TCP performance. In the wireless setup, we compare the effects of MAC-layer aggregation with various A-MPDU and A-MSDU configurations.

Our experiments show that while TCP performance degrades in transport-layer aggregation in wired settings, most TCP variants remain relatively robust under moderate aggregation levels. In contrast, A-MSDU and A-MPDU, which are MAC-layer aggregation mechanisms, significantly improve performance in wireless environments, achieving a throughput of more than three times higher. We also evaluated different maximum A-MPDU sizes to represent more controlled aggregation levels; however, this operates differently compared to transport-layer packet aggregation. Across both scenarios, NewReno exhibits stable and resilient behaviour, while Vegas shows highly inconsistent performance. These results highlight the importance of aligning aggregation strategies with the design expectations of congestion control algorithms and the characteristics of the underlying network.

By comparing the transport-layer and MAC-layer aggregation effects side by side, this work provides a comprehensive and quantitative assessment of algorithmic resilience when the TCP feedback loop is intentionally disrupted.

2 Background and Related Work

Over the years, several congestion control algorithms have been developed to improve TCP performance under diverse network conditions. Early designs such as Reno and its improved version, NewReno [10], follow a loss-based approach, where congestion is inferred from packet loss events. NewReno enhanced Reno's recovery behaviour, particularly during partial losses. In contrast, Vegas [5] uses a delay-based algorithm that aims to detect and avoid congestion by monitoring variations in round-trip delay and adjusting the sending rate before losses occur.

One commonly used modern algorithm is Cubic [15], which is now the default in all major operating systems. Cubic remains fundamentally loss-based but modifies the window growth function to handle high-speed, long-delay paths better. Another modern variant is BBR [11], introduced by Google, which builds a model of the network path by estimating bottleneck bandwidth and round-trip time.

TCP congestion control mechanisms are typically evaluated using metrics such as throughput, fairness, and convergence time. Comparative studies have analyzed the behaviour of delay-based and loss-based algorithms like Vegas and NewReno [6], while evaluations of Cubic provided insights into its performance across diverse network scenarios [13]. The behaviour of BBR has also been extensively studied, including analyses of its first version [11], comparisons between BBRv1 and BBRv2 [17], and evaluations of its latest revisions [19]. Such studies help to establish the standard testing environments and key metrics for analyzing congestion control performance.

However, real-world networks often differ from idealized, controlled environments. In particular, wireless and asymmetric networks often disrupt the feedback loop of TCP. A key phenomenon that occurs is ACK suppression, which is a reduction in the number or frequency of acknowledgement packets sent back to the sender. ACK aggregation is a related mechanism in which multiple ACKs are intentionally

(a) Wired Setup

(b) Wireless Setup

Figure 1: Dumbbell topology of wired and wireless simulation.

delayed and then transmitted together as a single burst. Ultimately, sending only the last ACK in a series produces the same effect as ACK aggregation. In Wi-Fi, the logic of ACK aggregation happens in the MAC-layer scheduling mechanisms. Such changes in receiving ACKs have an effect on the behaviour of TCP, where ACKs are expected to be returned promptly for each received segment or after every few segments. When ACKs are aggregated, the sender may receive bursty or delayed feedback, potentially leading to congestion window mismanagement, increased queueing, and low throughput.

The disadvantageous impact of such conditions was recognized early by Balakrishnan et al. [4], who examined TCP's performance over asymmetric links and noted the vulnerability of congestion control to ACK-path limitations. More recently, Grazia et al. [9] studied TCP in IEEE 802.11 Wi-Fi environments and demonstrated that MAC-layer ACK aggregation can delay feedback enough to cause unnecessary retransmissions and overinflated congestion windows. Their findings highlight the importance of cross-layer awareness in congestion control design.

As mentioned, IEEE 802.11n introduces two essential MAC-layer aggregation mechanisms [12] [8]. A-MSDU works by grouping multiple MSDUs into a single one under one MAC header. It offers minimal overhead but requires retransmitting the entire aggregate if any subframe is corrupted. A-MPDU bundles multiple MPDUs, with each having its own MAC header, into a single PHY transmission opportunity, separated by delimiters. It has a slightly higher header overhead than A-MSDU, but it is more resilient in lossy channels. These mechanisms can be combined in two-level aggregation by embedding A-MSDUs inside A-MPDUs to optimize both efficiency and resilience. Although studies show that two-level aggregation often delivers the best throughput and delay performance [2], its interaction with modern congestion control algorithms such as BBRv3 and CUBIC has not yet been systematically evaluated.

DOCSIS is another environment where ACK suppression is common due to link-level asymmetries and scheduling delays. This can lead to irregular ACK patterns that mislead the congestion control algorithms. Abrahamsson [1] raised practical concerns on the IETF AQM mailing list about whether ACK suppression strategies could be beneficial in lossy uplinks, noting the potential for increased unfairness and misinterpretation of loss signals.

In formalizing these issues, Arun et al. [3] applied verification techniques to analyze the sensitivity of modern algorithms like BBR to feedback timing. Their work revealed that small changes in ACK timing, such as those introduced by suppression or aggregation, can trigger significantly different congestion responses. This highlights the need to understand these feedback anomalies across diverse algorithmic designs.

While there is literature about how ACK suppression and aggregation can degrade TCP performance, particularly in Wi-Fi and asymmetric links, there is a lack of systematic, comparative studies that examine how modern congestion control algorithms respond to these phenomena. Our research addresses that gap by using ns-3 simulations to emulate programmable ACK suppression strategies in both wired and wireless dumbbell topologies.

3 Methodology

This study uses the ns-3 network simulator to investigate the impact of ACK suppression on the performance of various TCP congestion control algorithms. The simulations are structured to compare TCP performance in two distinct network environments: a wired dumbbell topology (Figure 1a) and a wireless dumbbell topology (Figure 1b), both designed to include a central bottleneck link where congestion would typically occur.

In the wired configuration (Figure 1a), we use a conventional dumbbell topology, where two sets of leaf nodes communicate through a pair of routers connected by a shared bottleneck link. Each access link, which connects leaf nodes to the routers, operates at 100 Mbps with a 2 ms delay, while the bottleneck link itself is limited to 10 Mbps with a higher latency of 20 ms. This setup enables precise observation of congestion dynamics and feedback mechanisms.

To study the effects of delayed feedback, a custom ACK suppression queue is implemented and attached to the router on the reverse path. This queue intercepts all ACK packets and applies aggregation by delaying and batching them. The aggregation strategy buffers ACKs and releases them in bursts either after a fixed delay of 5 ms or once a specified number of packets have been accumulated. This custom queue directly controls the frequency of TCP feedback, allowing us to test different degrees of transport-layer suppression.

In parallel, wireless experiments are conducted using a modified dumbbell topology with IEEE 802.11n MAC-layer behaviour (Figure 1b). The left side of the network connects two sender nodes via point-to-point links to a station node (STA). The STA transmits data wirelessly to an access point (AP) using IEEE 802.11n in the 5 GHz band with high throughput (HT) mode enabled. Once the AP receives the data, it forwards it through a wired link to the right-side receivers. The wired links are configured identically to the access links in the wired configuration, meaning 100 Mbps with a 2 ms delay. Both access points and stations are assigned static positions using a grid layout, and all nodes are configured with the ns-3 Internet stack. IP addresses are assigned using distinct subnets for each network segment, and global routing tables are automatically populated to ensure proper packet forwarding.

MAC-layer aggregation is selectively enabled or disabled through ns-3 configuration parameters. The testbed configurations include enabling only A-MPDU, only A-MSDU, both or neither. Furthermore, we experiment with different values for the maximum aggregation size of A-MPDU.

Although aggregation occurs at different layers, both topologies explore the same fundamentals of how TCP congestion control algorithms react to suppressed or delayed feedback caused by packet or frame aggregation.

We record and collect the key performance metrics, such as throughput, congestion window size, retransmissions, and fairness. Plots are created of the results using additional Python scripts.

4 Evaluation

All experiments were conducted in ns-3 using a dumbbell topology where leaf nodes on the left send data to the receiver nodes on the right. To evaluate the impact of ACK suppression, we tested four TCP variants: BBRv3, Cubic, NewReno, and Vegas. Each simulation ran for 60 seconds of emulated time, starting from 2 seconds to 62 seconds, to ensure the topology was fully initialized before measurements began. A baseline without suppression is conducted for comparison in both setups, enabling a clear assessment of how each TCP variant responds to varying degrees of ACK aggregation in the network.

4.1 Transport-layer Aggregation

In the wired dumbbell setup, the custom ACK aggregation queue was installed as the root queue on the right router's interface facing the bottleneck link. The queue size was configured to have approximately one bandwidth-delay product (BDP) worth of ACK traffic, ensuring it was large enough to avoid premature packet drops while remaining sensitive to congestion. Incoming ACKs were intercepted, and depending on the set aggregation ratio, they were either dropped or forwarded.

To support correct behaviour under suppression, a 5-millisecond timer was added. This timer ensures that an ACK is eventually forwarded even if the specified ratio has not been reached yet. Without this mechanism, we observed that many of the algorithms failed to make progress at higher suppression ratios. Feedback is too sparse in this case, so a sender may stall entirely, waiting indefinitely for acknowledgements that never come.

In each run, a single TCP flow is started, sending 100,000 packets of size 1448 B. Using aggregation ratios of 1:1, 2:1, 4:1, 8:1, and 16:1, this range allows us to quantify how each congestion control algorithm adapts to progressively sparser acknowledgement feedback in a controlled, wired environment.

Throughput

Table 1 shows the average throughput in Mbps achieved by each TCP variant under different ACK aggregation ratios, and Figure 2 shows how the throughput changes over time.

Table 1: Average throughput of TCP variants under varying aggregation levels (in Mbps).

Algorithm	1:1	2:1	4:1	8:1	16:1
BBRv3	8.84	6.67	6.57	6.50	6.50
CUBIC	9.00	8.53	8.53	8.50	1.96
NewReno	9.03	8.55	8.50	8.38	7.50
Vegas	1.76	8.56	8.52	2.62	5.56

For BBRv3, average throughput drops immediately when aggregation is introduced, but interestingly, it stabilizes afterwards and maintains a consistent average of 6.5 Mbps in higher ratios. This behaviour is consistent with BBR's model-driven control logic, which relies less on individual ACKs and more on measured bottleneck bandwidth and round-trip time. However, even BBR needs timely feedback to maintain high throughput, so the initial drop suggests that its bandwidth probing is disrupted by the reduced feedback frequency.

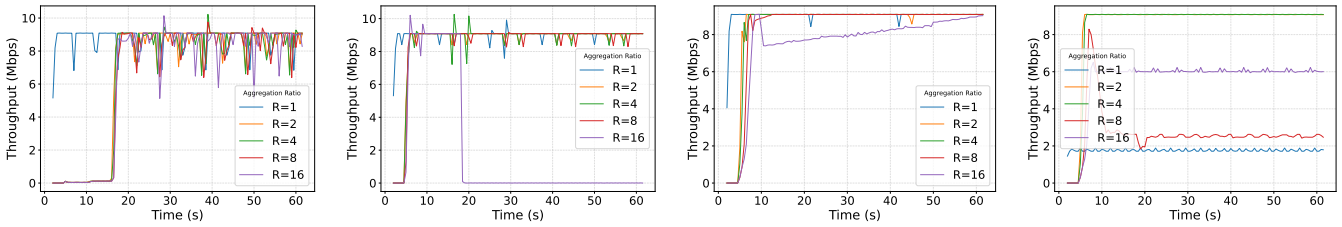
Figure 2a shows that all aggregation ratios result in a slow start, but then they all oscillate around a common steady-state rate of about 9 Mbps. Temporary drops occur, but recovery is consistent. As aggregation increases, the difference in drops becomes higher, reflecting possible misestimations of bottleneck bandwidth, but BBRv3 quickly converges back due to its active probing mechanism.

Cubic shows minimal sensitivity to moderate aggregation. The average throughput remains close to the baseline for ratios up to 8:1, indicating Cubic's robustness to delayed ACKs. Only under extreme aggregation does performance degrade significantly, dropping to 1.96 Mbps. This degradation likely results from delayed or insufficient feedback, causing longer retransmission timeouts, which lead to the collapse of the congestion window.

Similarly to BBRv3, Cubic reaches its optimal throughput of 9 Mbps quickly across most ratios, showing minimal variability (Figure 2b). However, at the extreme 16:1 ratio, throughput collapses to near zero, suggesting retransmission timeouts or severe window reductions caused by insufficient ACK feedback.

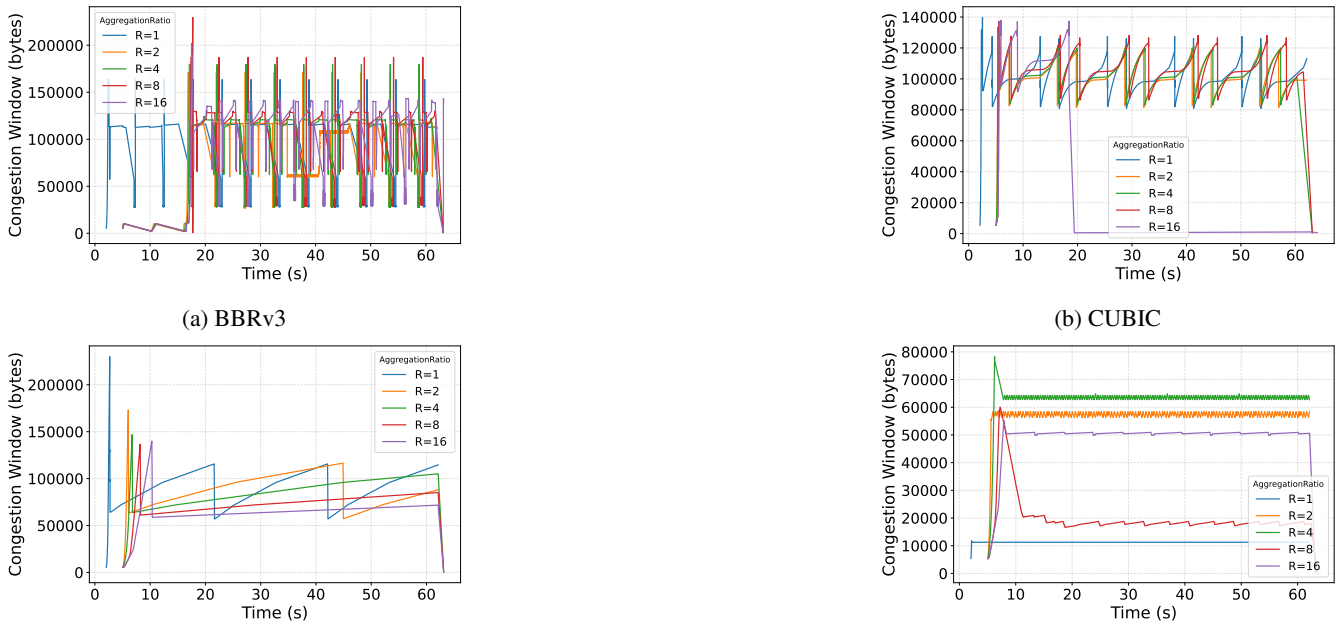
NewReno's average throughput gradually decreases; there is a more noticeable drop from the baseline to 2:1 aggregation and another more pronounced reduction between 8:1 and 16:1 aggregation ratios.

According to Figure 2c, NewReno starts ramping up around 5 seconds, with the convergence speed inversely re-



(a) BBRv3 (b) CUBIC (c) NewReno (d) Vegas

Figure 2: Throughput over time across TCP variants at varying aggregation ratios.



(a) BBRv3 (b) CUBIC

(c) NewReno (d) Vegas

Figure 3: Evolution of congestion window sizes across TCP variants at varying aggregation ratios.

lated to the aggregation ratio. At higher aggregation levels, it takes longer to reach 9 Mbps, and in the 16:1 scenario, there is a pronounced drop of around 10 seconds. However, it recovers slowly and approaches the steady rate of 9 Mbps by the end of the experiment. This illustrates NewReno's steady but ACK-paced growth and its resilience in the long run.

Vegas presents a unique pattern. Under no aggregation, it performs poorly, reaching only about 1.8 Mbps. This reflects on its conservative, delay-based congestion avoidance in a high-bandwidth, low-loss wired environment. Moderate aggregation reduces ACK frequency and the perceived queuing delay, allowing Vegas to send data more aggressively, which leads to higher bandwidth utilization. Surprisingly, at 16:1, Vegas reaches 6 Mbps, possibly because the delay signals become so imprecise that Vegas increases its window more than usual despite potential instability.

Without aggregation, Vegas reaches only a minimal throughput early on and maintains it throughout the simulation (Figure 2d). At 2:1 and 4:1 ratios, Vegas climbs to higher throughput levels and keeps them stable. At 8:1, it increases quickly but then settles at a lower level (2.7 Mbps). The most aggressive case shows a late surge that eventually reaches 6 Mbps. This delayed rise could be due to Vegas misinterpreting the sparsity of ACKs as reduced congestion, thereby expanding its window more than it normally would.

Overall, NewReno shows the most consistent performance across all aggregation levels. Cubic and BBRv3 both maintain high throughput under moderate aggregation, with Cubic experiencing less change. Vegas, while conservative, actually benefits from moderate aggregation, but its behaviour becomes less predictable as the ACK feedback becomes sparser. These results highlight the different sensitivities of loss-based, delay-based, and hybrid congestion control strategies under reduced feedback conditions.

Congestion-Window Changes

To understand how each TCP variant's sender adjusts its window when ACKs are suppressed, the time series traces of the congestion window were collected over the 60s simulation. Figure 3 shows the congestion window over time for the dif-

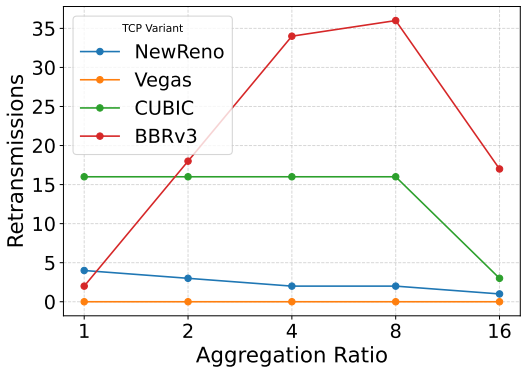


Figure 4: Number of retransmissions across TCP variants at different aggregation ratios.

ferent aggregation ratios, all plotted on the same axes for each TCP variant.

In the case of BBRv3, we observe significant fluctuations in the congestion window across all aggregation ratios. These fluctuations are expected due to BBR's periodic RTT probing, which temporarily reduces the congestion window, and its bandwidth probing, which causes growth phases. For Cubic, the frequency of growth-collapse cycles appears to increase slightly with higher aggregation ratios. NewReno exhibits a more predictable and gradual "sawtooth" pattern. After the initial start and stabilization, the cwnd ramps up and drops sharply once loss is detected. As aggregation increases, it takes longer for the cwnd to grow due to the reduced rate of ACK arrivals. Vegas, being delay-based, shows a drastically different cwnd behaviour. Without aggregation, Vegas quickly increases its window but then clamps it, and there is no further change, which results in a lower throughput. After the start-up, the congestion window settles, and there is barely any change in it for the rest of the simulation.

Retransmissions

Another important indicator of TCP performance under ACK suppression is the number of retransmissions each sender performs. Figure 4 plots, for each TCP variant, the total count of retransmitted segments observed over the 60 s wired run as a function of the aggregation ratio.

Vegas stands out with zero retransmissions across all aggregation levels. This reflects the nature of its delay-based design, which avoids filling queues and backs off before loss occurs. Even under extreme suppression, Vegas maintains enough responsiveness to avoid causing or reacting to loss events.

NewReno has the second lowest retransmissions. As aggregation increases, the number of duplicate ACKs received declines, making fast retransmit less likely to trigger. Due to its conservative growth and fallback mechanisms, NewReno continues to operate reliably without increasing loss.

Cubic, being more aggressive in probing the available bandwidth, results in a higher but relatively stable number of retransmissions. The retransmission count remains roughly consistent across most aggregation ratios, suggesting that its probing strategy remains unchanged regardless of ACK suppression. The exception is at 16:1 aggregation, where retransmissions drop sharply. This drop is likely due not to fewer losses but to insufficient ACK feedback to detect and react to them within the simulation period.

Initially, retransmissions are minimal, as BBRv3 quickly converges to the estimated bottleneck bandwidth and avoids unnecessary loss. However, as ACK feedback becomes sparser with increasing aggregation ratios, BBRv3's bandwidth and RTT estimation become noisy. The algorithm may mistakenly interpret a lack of feedback as an opportunity to send more aggressively, which leads to bursts of congestion and packet loss. Yet, at 16:1, BBRv3's retransmissions decrease, likely for the same reason as Cubic's.

In summary, retransmission behaviour under ACK suppression reveals important differences in algorithm design. Vegas, being delay-based, completely avoids loss. NewReno is robust and conservative, incurring minimal retransmissions. Cubic and BBRv3, while typically high-throughput performers, both become more vulnerable to excessive sending or misinterpretation of feedback.

Fairness

To evaluate fairness among different TCP congestion control algorithms, a set of experiments was conducted in which two concurrent TCP flows compete over the same bottleneck link. The key focus is on how these flows share available bandwidth under varying levels of ACK suppression. Each pairwise combination of TCP variants is tested under aggregation ratios of 1:1, 2:1, 4:1, 8:1, and 16:1.

Intra-Algorithm Fairness

We first evaluate fairness when both flows run the same congestion control algorithm. Ideally, in a fair system, each flow should obtain approximately 50% of the total throughput.

Table 2 shows that both BBRv3 and CUBIC maintain excellent fairness at low to moderate aggregation levels. However, they both drop to 0.5 at an 8:1 aggregation ratio, which indicates that one flow dominates the bandwidth while the other starves. Interestingly, Cubic recovers, suggesting that under very sparse ACK feedback, Cubic's congestion control mechanism stabilises in a way that rebalances the flow shares. BBRv3, on the other hand, does not recover, maintaining poor fairness at the highest aggregation levels. This may be due to BBR's model-based approach being more sensitive to feed-

Table 2: Intra-algorithm fairness: Jain's Fairness Index between two identical flows under varying aggregation levels.

Algorithm	1:1	2:1	4:1	8:1	16:1
BBRv3	0.999	0.978	0.924	0.502	0.500
CUBIC	0.999	0.998	0.977	0.500	0.942
NewReno	1.000	0.994	0.500	0.500	0.501
Vegas	1.000	0.627	0.975	0.502	0.503

Algorithms	1:1	2:1	4:1	8:1	16:1	Generally Favoured Variant
BBRv3 vs. CUBIC	0.998	0.936	0.500	0.500	0.501	CUBIC
BBRv3 vs. NewReno	0.997	0.972	0.500	0.500	0.507	NewReno
BBRv3 vs. Vegas	0.557	0.559	0.500	0.653	0.846	BBRv3
CUBIC vs. NewReno	0.995	0.979	0,500	0.500	0.849	NewReno
CUBIC vs. Vegas	0.535	0.516	0.505	0.594	0.828	CUBIC
NewReno vs. Vegas	0.545	0.539	0.968	0.501	0.503	Vegas

Table 3: Inter-algorithm fairness: Jain's Fairness Index between two distinct TCP variants under varying aggregation ratios.

back scarcity, causing unfair resource allocation. NewReno exhibits near-perfect fairness at 1:1 and 2:1 aggregation levels but quickly degrades to 0.5 from 4:1 onward, indicating persistent unfairness under higher ACK suppression. Vegas shows the most unpredictable behaviour. While perfectly fair at 1:1, it drops sharply at 2:1 and fluctuates thereafter. This irregularity could stem from the delay-based congestion control reacting inconsistently to ACK suppression, which affects how it shares bandwidth with its identical flows.

Inter-Algorithm Fairness

Next, we examine fairness across different congestion control algorithms when two flows use different TCP variants. The same aggregation ratios are applied to assess whether any algorithm systematically dominates others under transport-layer aggregation. The "Generally Favoured Variant" column in Table 3 reflects the overall trend across all tested aggregation ratios while acknowledging that dominance may shift at specific levels of aggregation.

At low aggregation ratios (1:1 and 2:1), fairness between pairs is generally high, indicating balanced resource sharing. As aggregation increases, differences in algorithm behaviour emerge. Loss-based variants, such as CUBIC and NewReno, often secure a larger share of throughput at higher aggregation levels due to their more aggressive congestion window growth and burst accommodation. In contrast, delay- and model-based variants (Vegas, BBRv3) can be less favoured at these higher aggregation levels, although they occasionally gain an advantage.

4.2 MAC-Layer Aggregation

In the wireless setup, MAC-layer aggregation was evaluated by varying the A-MPDU and A-MSDU parameters. The wireless link consisted of a fixed HT AP and a fixed HT STA, both operating on a 20-MHz channel and using the 64-QAM 3/4 modulation coding scheme (MCS), resulting in a PHY data rate of 65 Mbps. To ensure long flows and full utilisation of the link during the 60-second experiment duration, the total number of packets was increased to 500,000.

The maximum A-MSDU frame size is 7935 bytes, which is 256 bytes shorter than the maximum physical layer PSDU size (8191 bytes) since reserved space is allocated for future control information [16]. The maximum length of the PSDU that may be received is 65,535 bytes. At the MAC layer, frame aggregation reduces protocol overhead by combining multiple data frames into a single transmission unit.

Throughput

Both schemes were either disabled or set to their maximum aggregation levels, allowing an investigation of four configurations: (i) no aggregation, (ii) A-MSDU only, (iii) A-MPDU only, and (iv) both schemes enabled simultaneously.

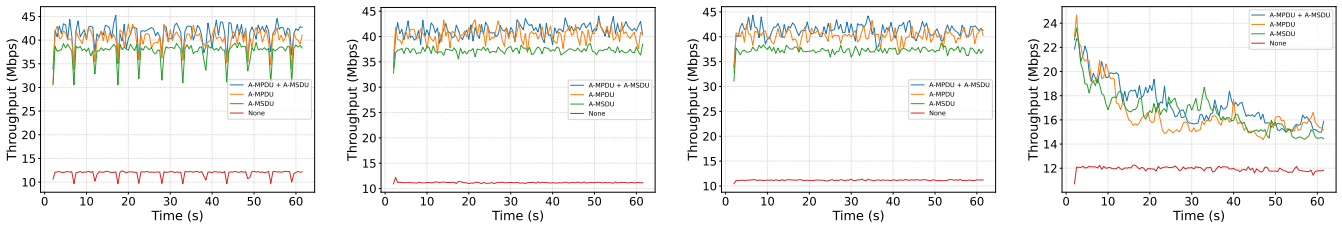
The average throughput results are presented in Table 4. Across all TCP algorithms, the baseline throughput without MAC-layer aggregation remains low and nearly identical due to substantial protocol overheads at the MAC layer. Enabling A-MSDU aggregation alone increases throughput three times compared to the baseline. This gain arises because A-MSDU aggregates multiple MAC Service Data Units into a single MAC Protocol Data Unit (MPDU), reducing header and contention overhead.

A-MPDU aggregation proves even more effective, as it operates at the MPDU level after MAC header encapsulation. Unlike A-MSDU, each subframe within an A-MPDU can be independently acknowledged and retransmitted, providing both efficiency and resilience against wireless errors. When both aggregation schemes are combined (two-level aggregation), throughput reaches approximately 41 Mbps, close to the maximum achievable given the PHY data rate and protocol overheads.

The delay-based Vegas algorithm follows a similar trend but consistently achieves lower throughput than the other variants under all aggregation schemes. Interestingly, Vegas benefits slightly more from A-MSDU than A-MPDU. This may stem from Vegas's sensitivity to RTT variance and its conservative congestion window adaptation: A-MSDU's reduced per-frame delays may provide a more stable RTT estimate, whereas A-MPDU's burstier transmissions introduce

Algorithm	baseline	A-MPDU + A-MSDU	A-MPDU	A-MSDU
BBRv3	11.90	41.43	40.24	37.57
CUBIC	11.17	41.19	40.33	37.23
NewReno	11.15	41.32	40.19	37.26
Vegas	11.94	17.34	16.52	16.68

Table 4: Average throughput for each TCP variant under varying aggregation schemes.



(a) BBRv3 (b) CUBIC (c) NewReno (d) Vegas

Figure 5: Throughput over time across TCP variants at varying aggregation schemes.

more variability into Vegas's RTT measurements.

Figure 5 further illustrates these trends. BBRv3 displays greater throughput fluctuations as expected due to its probing cycles. In contrast, both Cubic and NewReno, as loss-based algorithms, exhibit stable throughput once aggregation is enabled, regardless of aggregation type. Vegas initially overshoots throughput at the start of each flow but eventually stabilizes at lower throughput levels across all aggregation configurations. This initial overshoot is typical of Vegas's startup behaviour, where the congestion window briefly grows aggressively before Vegas detects a queuing delay and reduces its sending rate.

Overall, these results confirm that enabling both A-MSDU and A-MPDU aggregation yields the highest throughput efficiency. A-MPDU alone offers nearly equivalent gains while disabling both schemes results in substantial throughput degradation. Without aggregation, each frame incurs full MAC overhead (headers, inter-frame spaces, backoff delays), which severely limits efficiency at the wireless medium compared to the PHY rate.

Congestion-Window Changes

Figure 6 illustrates the congestion window evolution for each algorithm under the various aggregation configurations. As seen previously, BBRv3 exhibits frequent congestion window adjustments, consistent with its bandwidth and RTT-probing behaviour. Cubic rapidly stabilizes at a fixed congestion window size across all aggregation schemes. Interestingly, its steady-state congestion window is slightly larger when aggregation is enabled, likely due to reduced loss rates and more efficient data delivery per RTT cycle.

NewReno demonstrates a continuous and linear congestion window growth throughout the experiments. When either A-MPDU or A-MSDU is enabled, NewReno's window growth becomes steeper than in the baseline, reflecting the benefits of reduced packet loss and improved transmission efficiency on its additive-increase behaviour.

Vegas shows a different pattern: with aggregation enabled, it initially builds up the congestion window rapidly but quickly retreats to a lower, stable window size. This behaviour reflects Vegas's early detection of queuing delay induced by aggregated bursts, which triggers its congestion avoidance mechanism and leads to conservative window size.

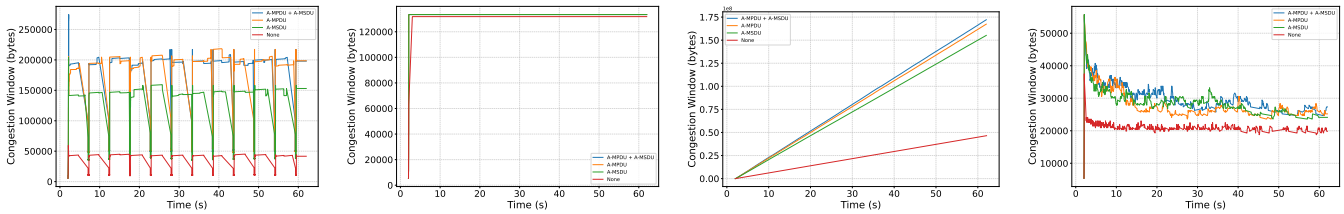
Varying A-MPDU

To further investigate A-MPDU's impact, we varied its aggregation size while keeping A-MSDU disabled. The maximum A-MPDU size was configured to 4 KB, 8 KB, 16 KB, 32 KB, and 64 KB. These values reflect increasing numbers of aggregated MPDUs while staying within standard PHY layer limits. The packet payload size remains fixed at 1448 bytes; therefore, the effective number of aggregated frames per A-MPDU depends on the A-MPDU limit, MAC header overhead, and padding requirements [14]. Since A-MPDU operates after encapsulation, it can aggregate multiple complete MPDUs while preserving independent retransmission for each subframe, reducing retransmission costs compared to A-MSDU.

Table 5: Average throughput for each TCP variant under different maximum aggregation size of A-MPDU.

Algorithm	64KB	32KB	16KB	8KB	4KB
BBRv3	40.24	40.17	38.71	34.53	28.20
CUBIC	40.33	40.33	38.48	34.28	27.42
NewReno	40.19	40.11	38.46	38.46	27.22
Vegas	16.52	16.52	15.58	16.20	17.50

As Table 5 shows, increasing the A-MPDU size steadily improves throughput for BBRv3, Cubic, and NewReno, though the gains diminish at higher aggregation sizes due to saturation of the available PHY bandwidth. NewReno exhibits slightly stronger throughput gains at 8 KB aggregation



(a) BBRv3 (b) CUBIC (c) NewReno (d) Vegas

Figure 6: Evolution of congestion window sizes across TCP variants at varying aggregation schemes.

compared to the others, suggesting that even moderate aggregation significantly benefits its loss-recovery behaviour.

Vegas again displays atypical behaviour. Its throughput decreases as aggregation increases up to 16 KB, after which it stabilizes. This may be attributed to Vegas's heightened sensitivity to aggregation-induced queuing delays, which causes premature window reductions. The burstiness introduced by larger A-MPDUs may briefly overload buffers and inflate RTT estimates, triggering Vegas's conservative congestion response.

In summary, larger A-MPDU aggregation reduces MAC overhead by amortizing headers, contention backoffs, and inter-frame gaps across multiple frames, thereby improving medium utilization. However, the extent of the benefit depends on the congestion control algorithm's sensitivity to burstiness and RTT variability introduced by the aggregation process.

5 Discussion

Our evaluation demonstrates clear distinctions between transport-layer and MAC-layer aggregation mechanisms in terms of their compatibility with TCP congestion control and overall impact on throughput.

Under the wireless conditions, transport-layer aggregation via application-layer queuing was found to negatively interact with TCP congestion control and 802.11 retransmission mechanisms, resulting in severe throughput degradation and retransmissions. In contrast, MAC-layer aggregation mechanisms, such as A-MPDU and A-MSDU, are specifically designed to optimise wireless medium access by reducing protocol overhead while remaining fully compatible with IEEE 802.11 retransmission and acknowledgement mechanisms. MAC-layer aggregation achieves higher throughput and better channel utilization without sacrificing packet-level retransmission granularity or congestion responsiveness.

As expected, two-level aggregation (both A-MSDU and A-MPDU enabled) offers the best performance across all TCP variants [16], with throughput gains approaching the theoretical maximum of the 65 Mbps PHY rate.

5.1 TCP Variant Behaviour

BBRv3 and Cubic are sensitive to aggregation at the transport layer. With increasing aggregation ratios, both exhibit degradation in fairness and high retransmission counts. This can be attributed to the fact that these algorithms rely on timely feedback from either loss or RTT measurements, both of which are distorted when ACK suppression is applied. Nonetheless, BBRv3 and Cubic benefit substantially from MAC-layer aggregation, where timely ACK feedback remains intact and overhead reduction directly improves throughput.

NewReno, while traditionally seen as more conservative and legacy, demonstrated surprisingly robust and stable behaviour across both transport-layer and MAC-layer aggregation scenarios. It showed fewer retransmissions, better fairness retention, and a strong response to increasing A-MPDU sizes, suggesting that its simple congestion control is more tolerant to moderate levels of aggregation-induced feedback distortion.

Vegas, on the other hand, consistently showed unpredictable behaviour under both aggregation types. As a delay-based algorithm, Vegas relies on accurate RTT estimation. Aggregation, particularly bursty transmission patterns introduced by queuing and MAC-level bundling, introduces RTT variability that Vegas is not equipped to handle, leading to either premature window reductions or overly aggressive expansions. This behaviour confirms earlier findings that delay-based algorithms often struggle in highly variable or bursty environments [5].

5.2 Additional overhead of the queue

The custom ACK suppression queue itself introduces only minimal processing overhead, as each packet requires only simple parsing to inspect TCP flags. Our simulation results confirm that enabling the queuing mechanism at an aggregation ratio of 1:1 yields identical results to the baseline configuration, indicating that any overhead from queue processing does not affect flow performance.

5.3 Future work

As with any simulation-based study, certain limitations apply. Our wireless topology uses idealized channel conditions without mobility, interference, or jitter, which may underestimate the variability present in real-world WLAN deployments. Similarly, while the custom queuing implementation accurately models aggregation effects at the transport layer, real-world deployments may exhibit additional complexities.

Furthermore, fairness in MAC-layer aggregation scenarios has not been systematically evaluated, representing an area for future work. Particularly for multi-flow and multi-user wireless scenarios. Additionally, incorporating Block ACK [2] mechanisms would more accurately model real-world IEEE 802.11 behaviour.

6 Responsible Research

The study involves only software simulation in ns-3 [1] and does not handle any private data or interact with human subjects. All code modifications and experiment scripts are well documented and made publicly available. [2]

The experiments are reproducible since all topology configurations and setups are shared in a public repository. Each simulation can be recreated using the provided code and the study's parameters.

During the writing process, a Large Language Model was used as a support tool to improve the clarity and structure of the text. The LLM generated no scientific ideas or interpretations. Additionally, Python scripts for the data visualization were generated using an LLM but thoroughly reviewed and validated before being included in the study. The ns-3 simulation models were implemented based on official documentation and tutorials, with custom extensions explicitly developed for the purposes of this research.

7 Conclusions

This study systematically evaluated the interaction between TCP congestion control algorithms and aggregation mechanisms at both the transport and MAC layers. Our results demonstrate that aggregation can have both beneficial and detrimental effects on TCP performance, depending critically on where and how it is applied.

Transport-layer aggregation via delayed ACKs presents significant challenges for many TCP variants, resulting in throughput degradation, increased retransmissions, and fairness loss, particularly for algorithms that rely heavily on timely feedback. NewReno, despite its simplicity, showed robust performance and stability under moderate aggregation levels, demonstrating its resilience to delayed acknowledgements.

In contrast, MAC-layer aggregation mechanisms (A-MSDU, A-MPDU) are highly effective in wireless networks, providing substantial throughput gains by reducing protocol overhead while preserving compatibility with TCP control loops and retransmission mechanisms. The combination of A-MSDU and A-MPDU aggregation consistently delivered the highest throughput, approaching the theoretical capacity of the wireless link.

Across both wired and wireless scenarios, our findings emphasize the importance of aligning aggregation strategies with the congestion control algorithm in use and the characteristics of the network environment. While MAC-layer aggregation offers clear benefits for wireless networks, transport-layer aggregation must be applied cautiously to avoid destabilizing TCP's feedback-driven behaviour.

These insights contribute to a deeper understanding of cross-layer interactions in modern networks and may inform future protocol designs that coexist with aggregation and congestion control.

[1] https://www.nsnam.org/

[2] The complete source code, simulation scripts, and plotting tools are available at: https://github.com/hheinczinger/ack-aggregation.

References

[1] Mikael Abrahamsson. Tcp ack suppression. IETF AQM mailing list, 2015.

[2] Mustafa Al-Anbagi and Emad Al-Hemiary. Impact of tcp congestion control algorithms on ieee802.11n mac frame aggregation. *International Journal of Computer Science Engineering and Technology (IJCSET)*, Vol 2:1410–1414, 09 2012.

[3] Venkat Arun, Mina Tahmasbi Arashloo, Ahmed Saeed, Mohammad Alizadeh, and Hari Balakrishnan. Toward formally verifying congestion control behavior. In *SIGCOMM*, 2022.

[4] Hari Balakrishnan, Venkata Padmanabhan, and Randy Katz. The effects of asymmetry on tcp performance. *Mobile Networks and Applications*, 4, 1999.

[5] Lawrence S. Brakmo, Sean W. O'Malley, and Larry L. Peterson. Tcp vegas: new techniques for congestion detection and avoidance. *SIGCOMM Comput. Commun. Rev.*, 24(4):24–35, October 1994.

[6] Tanjia Chowdhury and Mohammad Alam. Performance evaluation of tcp vegas over tcp reno and tcp newreno over tcp reno. *JOIV : International Journal on Informatics Visualization*, 3, 08 2019.

[7] Wesley Eddy. Transmission Control Protocol (TCP). RFC 9293, August 2022.

[8] Boris Ginzburg and Alex Kesselman. Performance analysis of a-mpdu and a-msdu aggregation in ieee 802.11n. In *2007 IEEE Sarnoff Symposium*, pages 1–5, 2007.

[9] Carlo Augusto Grazia, Natale Patriciello, Toke Hoiland-Jorgensen, Martin Klapez, and Maurizio Casoni. Aggregating without bloating: Hard times for tcp on wi-fi. *IEEE/ACM Transactions on Networking*, 30(5), 2022.

[10] Andrei Gurtov, Tom Henderson, Sally Floyd, and Yoshifumi Nishida. The NewReno Modification to TCP's Fast Recovery Algorithm. RFC 6582, April 2012.

[11] Mario Hock, Roland Bless, and Martina Zitterbart. Experimental evaluation of bbr congestion control. pages 1–10, 10 2017.

[12] Raja Karmakar, Samiran Chattopadhyay, and Sandip Chakraborty. Impact of ieee 802.11n/ac phy/mac high throughput enhancements on transport and application protocols—a survey. *IEEE Communications Surveys Tutorials*, 19(4):2050–2091, 2017.

[13] Douglas Leith, Robert Shorten, and G. McCullagh. Experimental evaluation of cubic-tcp. In *PFLDnet*, 2007.

[14] Bakeel Maqhat, Mohd Baba, Ruhani ab rahman, and Anwar Saif. Scheduler algorithm for ieee802.11n wireless lans. *International Journal of Future Computer and Communication*, 3:222–226, 01 2014.

[15] Ayush Mishra, Lakshay Rastogi, Raj Joshi, and Ben Leong. Keeping an eye on congestion control in the wild with nebby. In *Proceedings of the ACM SIGCOMM 2024 Conference*, SIGCOMM '24, pages 1–15. ACM, August 2024.

[16] Dionysios Skordoulis, Qiang Ni, Hsiao-hwa Chen, Adrian P. Stephens, Changwen Liu, and Abbas Jamalipour. Ieee 802.11n mac frame aggregation mechanisms for next-generation high-throughput wlans. *IEEE Wireless Communications*, 15(1):40–47, 2008.

[17] Yeong-Jun Song, Geon-Hwan Kim, Imtiaz Mahmud, Won-Kyeong Seo, and You-Ze Cho. Understanding of bbrv2: Evaluation and comparison with bbrv1 congestion control algorithm. *IEEE Access*, PP:1–1, 02 2021.

[18] Belma Turkovic, Fernando A. Kuipers, and Steve Uhlig. Interactions between congestion control algorithms. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*, pages 161–168, 2019.

[19] Danesh Zeynali, Emilia Weyulu, Seifeddine Fathalli, Balakrishnan Chandrasekaran, and Anja Feldmann. Promises and potential of bbrv3. In *PAM 2024*, 2024.