Flex-Cubic: A Runtime-Adaptive Loss-Tolerant TCP Cubic

Pedro P. P. Filho[1,2], Moises R. N. Ribeiro[1], Magnos Martinello[1], Guilherme F. Miguel[1]

[1]Universidade Federal do Espirito Santo (UFES)
Av. Fernando Ferrari, 514 – 29.075-910 – Vitória – ES – Brazil

[2]Instituto Federal do Espirito Santo (Ifes), ES – Brazil

pedro.pecolo@ifes.edu.br
{moises.ribeiro, magnos.martinello,guilherme.miguel}@ufes.br

Abstract. *Traditional TCP still lacks the ability to differentiate between losses primarily caused by congestion from those caused by physical layer errors. This may severely impair performance, especially in data-intensive science applications over high-capacity and long-distance dynamically reconfigurable transparent optical networks. This work proposes and experimentally implements a variant of TCP Cubic designed for reconfigurable networks to turn transport layer tolerant to non-congestion losses and variable RTT exploiting available bandwidth more efficiently. This is done by designing a congestion window (cwnd) reduction mechanism that conditions loss reactions on evidence of congestion, given by RTT measurements. In addition, Flex-Cubic aims to supports dynamic parameter tuning and higher-precision timing, resulting in greater stability and improved bandwidth utilization. Thus, eBPF has been used as a platform for TCP congestion control algorithm (CCA) implementation, enabling new algorithms to be loaded into the kernel via JIT at runtime, without recompilation. Through struct_ops and maps, eBPF allows per-flow instrumentation and dynamically CCA adaptation.*

Resumo. *O TCP tradicional ainda carece da capacidade de diferenciar entre perdas causadas principalmente por congestionamento daquelas causadas por erros na camada física. Isso pode prejudicar severamente o desempenho, especialmente em aplicações científicas com uso intensivo de dados em redes ópticas transparentes, dinamicamente reconfiguráveis, de alta capacidade e longa distância. Este trabalho propõe e implementa experimentalmente uma variante do TCP Cubic projetada para redes reconfiguráveis, tornando a camada de transporte tolerante a perdas não relacionadas a congestionamento e a RTTs variáveis, explorando a largura de banda disponível de forma mais eficiente. Isso é feito através do projeto de um mecanismo de redução da janela de congestionamento (cwnd) que condiciona as reações de perda à evidência de congestionamento, fornecida por medições de RTT. Além disso, o Flex-Cubic visa suportar o ajuste dinâmico de parâmetros e temporização de maior precisão, resultando em maior estabilidade e melhor utilização da largura de banda. Assim, o eBPF foi utilizado como plataforma para a implementação do algoritmo de controle de congestionamento (CCA) do TCP, permitindo que novos algoritmos sejam carregados no kernel via JIT em tempo de execução, sem recompilação. Através de struct_ops e mapas, o eBPF permite a instrumentação por fluxo e a adaptação dinâmica do CCA.*

1. Introduction

The Transmission Control Protocol (TCP) remains the dominant transport protocol for reliable data delivery, and its congestion control mechanisms are a fundamental pillar of Internet stability. Since their introduction in the late 1980s and early 1990s, classic loss-based algorithms such as Tahoe and Reno have defined the core principles of congestion control, including slow start, congestion avoidance, and the Additive Increase/Multiplicative Decrease (AIMD) paradigm, all driven by the evolution of the congestion window (cwnd) [Mathis et al. 1997].

The congestion control mechanisms are grounded in the assumption that packet loss is a reliable indicator of network congestion, an hypothesis that has shaped subsequent designs and remains embedded in widely deployed variants such as Cubic [Ha et al. 2008]. Although delay-based approaches, such as Vegas, explored latency as an early congestion signal, their adoption was limited by practical deployment challenges, leading recent research to favor hybrid models that combine multiple signals, as exemplified by BBR (Bottleneck Bandwidth and Round-trip propagation time) [Yang et al. 2023].

Despite their success, traditional TCP congestion control algorithms still lack the ability to differentiate between packet losses caused by congestion and those arising from physical-layer impairments. This limitation is particularly severe in high-capacity, long-distance, and dynamically reconfigurable networks. In transparent optical networks, non-congestion losses, due to increased residual bit error rate (BER) by optical noise accumulation along transparent hops, are an important issue in traffic engineering (TE) context [Borges et al. 2024], [Pedro Filho et al. 2025]. Low-orbit satellite link dynamics pose challenges to current congestion control algorithms (CCA) [Lai et al. 2025]. In these environments, loss-based reactions that indiscriminately interpret every loss as congestion result in unnecessary reductions of the cwnd, leading to inefficient use of available bandwidth and substantial overall performance degradation.

For the case of scientific applications, and large-scale distributed machine learning, there are transfer of massive data volumes across long delay [Zanotelli et al. 2025]. Timing constraints and highly synchronized communication patterns, making them extremely sensitive to micro-congestion events and transient throughput fluctuations [Chou and Chung 2024], [Antelmi and Carlini 2026]. This means that the TCP congestion control must evolve from simple reactive mechanisms to more adaptive and informed systems, as modern network challenges the assumptions of traditional CCAs.

There is a pressing need for transport mechanisms capable of distinguishing between different loss causes, adapting to RTT (Round-Trip Time) dynamics, and also exploiting fine-grained kernel-level instrumentation. In this context, technologies such as eBPF (Extended Berkeley Packet Filter) have emerged as key enablers of real-time observability [Song and Li 2024] and more informed congestion control and transport decisions [Zadeh et al. 2023]. Beyond observability, the programmability offered by eBPF has opened the door to novel transport-layer innovations, allowing the implementation of entirely new high-performance transport protocols directly within the secure kernel environment, while avoiding the complexity and limitations of traditional kernel development or user-space bypass [Chen et al. 2025].

In contrast to early solutions constrained by static processing pipelines and param-

eters rigidly embedded in the TCP stack [Pecolo F et al. 2025], [Zanotelli et al. 2025], this work proposes and experimentally evaluates Flex-Cubic, a variant of TCP Cubic specifically designed for reconfigurable network environments. Focusing on loss-based algorithms, the central idea is to make the transport layer tolerant to non-congestion-related losses and variable RTTs by conditioning loss reactions on explicit evidence of congestion derived from RTT measurements. This enables loss-based algorithms to distinguish between sporadic losses and congestion-induced losses. Flex-Cubic introduces a refined congestion window reduction mechanism, supports dynamic parameter tuning, and leverages higher-precision timing to improve stability and bandwidth utilization. In order to illustrate its potential even using an emulated environment such as Mininet, a simple and yet effective algorithm is implemented using eBPF, which enables TCP congestion control algorithms to be safely loaded into the kernel at runtime via JIT (Just In Time) compilation, without recompilation. By exploiting eBPF struct ops and maps, the solution allows per-flow instrumentation and dynamic adaptation of congestion control behavior, making it well suited for modern, programmable, and high-performance network infrastructures.

The remainder of this paper is organized as follows: Section 2 provides a concise overview of related works pertinent to research theme. Section 3 then presents the behavioral model of the standard TCP Cubic compared to TCP Flex-Cubic, along with the application of the eBPF tool and its interaction with the TCP framework. Finally, Sections 4 and 5 present a description of the setup used and results of the study, followed by the conclusion Section.

2. Background and Related Work

A structural limitation remains in most CCAs in the differentiation between losses caused by congestion events, illustrated in Figure 1 (a), and those caused by corrupted bits at physical layer by noise and interference, which is depicted in Figure 1 (b). In long-distance optical fibers, free-space optical (FSO) links subject to atmospheric turbulence [Borges et al. 2024], and satellite communications links under noise and interference [Lai et al. 2025], a non-negligible fraction of packet losses stems from residual bit errors rate (BER), and not from buffer overload.

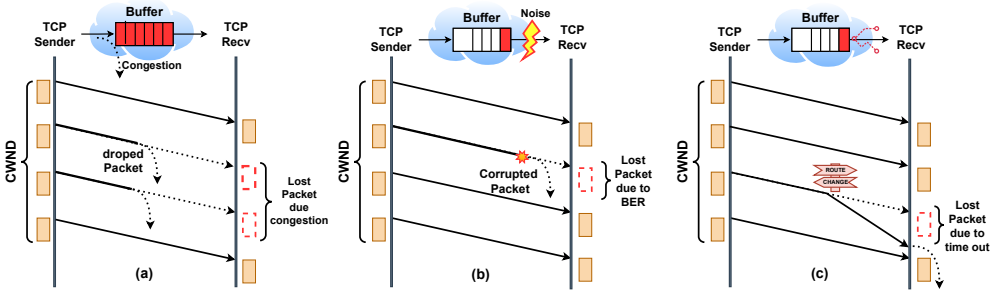

Figure 1. Discarding of packets due to: (a) buffer overload, (b) packet corruption, and (c) packet timeout.

As a result, losses caused by corrupted packets are interpreted as congestion signals, causing cwnd reductions, unwarranted rate oscillations, and systematic under-utilization of available bandwidth. In this work, we focus on networks composed of high Bandwidth-Delay Product (BDP) links, where this effect is particularly severe, as window

recovery after a reduction may require tens or hundreds of RTTs, significantly degrading average throughput, while also highlighting the behavior of loss-based algorithms.

In addition, there are strong dependence of congestion control on RTT. Route reconfigurations in adaptable networks may impose different propagation delays to in-flight packets of a given flow, as illustrated in 1 (c) triggering a loss event. Classic performance models show that TCP throughput is proportional to cwnd/RTT and inversely proportional to $RTT \cdot \sqrt{p}$, where p denotes the packet loss probability. Thus, links with high RTT require proportionally larger windows to fully exploit the available capacity. However, in modern networks, particularly in optical networks [Borges et al. 2024] and Low Earth Orbit (LEO) satellite networks [Lai et al. 2025], this assumption is no longer valid. In these systems, RTT becomes a dynamic variable, subject to frequent sudden changes; and modern CCAs must be equipped with tools for facing such challenges.

Extended Berkeley Packet Filter (eBPF) is established as a mature infrastructure for real-time, in-kernel monitoring and control [Miano et al. 2018] and [Song and Li 2024] suited for tackling the issues discussed above. In particular, for extending and observing the TCP/IP stack. It can safely influence TCP behavior at runtime, enhancing visibility without modifying the core kernel code [Zadeh et al. 2023]. This programmability enables transport-layer swift innovations. For instance, eTran [Chen et al. 2025] uses eBPF to implement entirely new, high-performance transport protocols directly within the secure kernel environment, avoiding the pitfalls of traditional development or user-space bypass. Furthermore, eBPF allows TCP to become path-aware, with mechanisms like the TCP Path Changer (TPC) dynamically rerouting active connections in response to failures or latency increases, leveraging IPv6 Segment Routing [Jadin et al. 2022].

A key challenge in leveraging eBPF is designing efficient control and monitoring architectures. While eBPF enables highly efficient packet processing, early approaches suffered from static pipelines and high overhead. Research has addressed this by proposing fully adaptive control planes that manage eBPF programs and metrics dynamically, moving beyond simple feature toggles to achieve true flexibility and performance [Magnani et al. 2022].

The promise of eBPF also extends to optimizing core network functions like congestion control. The TD-BBR [Pan et al. 2024] algorithm exemplifies this, using eBPF to implement a BBR variant that actively meets application delay targets while maintaining high throughput, showcasing eBPF's role in creating more responsive and application-aware control loops [Pan et al. 2024]. However, the performance benefits of eBPF are not universal. Analyses demystify its impact, showing that significant performance gains are primarily realized for simple applications placed optimally in the datapath (e.g., using eXpress Data Path - XDP).

3. Flex-Cubic: Runtime-Adaptive Loss-Tolerant TCP Cubic

This Section presents the basic operating logic of CCA in Flex-Cubic, A proposal to improve TCP Cubic based on [Wibowo E.and Bi 2025]. It is a variation of TCP Cubic aiming at including mechanisms that allow, for a certain degree, the differentiation between packet dropped due to buffer overflow from the ones discarded because of corrupted bits. In addition, by exploiting functionalities provided by eBPF, the goal is to provide runtime changes to transmission window growth control parameters. Here, TCP Cubic was cho-

sen as the base algorithm because it demonstrated positive results in situations involving queue bottlenecks and insertion losses, as presented in a previous study, [Pedro Filho et al. 2025].

3.1. TCP Cubic standard vs Flex-Cubic: basic concepts

The TCP Cubic modeling the behavior of cwnd using a cubic growth function over time, instead of the linear growth used by its predecessors. The algorithm operates through the following key phases: I - *Cubic Growth Function*; II - *Fast Recovery after Packet Loss*; III - *Slow and Fast Growth Phases*.

Therefore, the central concept of TCP Cubic lies in the evolution of cwnd following a cubic function of time, providing an increase in cwnd in high-latency networks, while allowing for smooth and stable adjustment compared to previous algorithms. In case of loss, Cubic acts by reducing cwnd by a multiplicative factor β (typically 0.7). After that, grows as a function of time t:

$$\text{CWND}(t) = C(t-K)^3 + W_{\max} \quad (1) \qquad K = \sqrt[3]{\frac{W_{\max} \cdot \beta}{C}} \quad (2)$$

Where C is a scaling constant that determines the growth rate (typically $C = 0.4$), t is the time elapsed since the last loss event, $W_{\max}$ is the congestion window value just before the last loss, K is the inflection point, and β is the multiplicative decrease factor. The value K defines the time at which the congestion window reaches $W_{\max}$ again: for $t < K$, the growth is slower to ensure stability, while for $t > K$, it becomes faster to enable more aggressive bandwidth utilization. An important property of TCP Cubic is its RTT-independence, as the congestion window growth depends on elapsed time rather than directly on RTT, improving fairness among flows with different delays.

Flex-Cubic extends TCP Cubic by preserving its loss-based behavior while incorporating RTT-based refinement to avoid unnecessary congestion window reductions when losses are not caused by congestion, i.e., when no RTT increase is observed. Its main goal is to address a key limitation of Cubic by preventing unwarranted cwnd reductions in the absence of queue buildup.

Unlike hybrid approaches such as TCP BBR, which rely on bandwidth estimation rather than packet loss as the primary control signal, Flex-Cubic remains fundamentally loss-driven, enhanced with RTT-based decision logic.

3.2. Providing a runtime-adaptive CCA for TCP: An eBPF-based approach

eBPF provides a flexible mechanism for capturing metrics and interacting with TCP congestion control, enabling experimentation and modification of CCA behavior directly in the kernel without recompilation or system restart. In Flex-Cubic, eBPF is used to safely inject custom logic into the TCP processing path, leveraging its verifiable execution model and kernel-enforced safety constraints.

Integration with the TCP subsystem is achieved through the *BPF_PROG_TYPE_STRUCT_OPS* program type, which allows extending or overriding kernel structures such as *tcp_congestion_ops* [Hinz et al. 2023]. This structure defines the set of callbacks invoked during the TCP connection lifecycle.

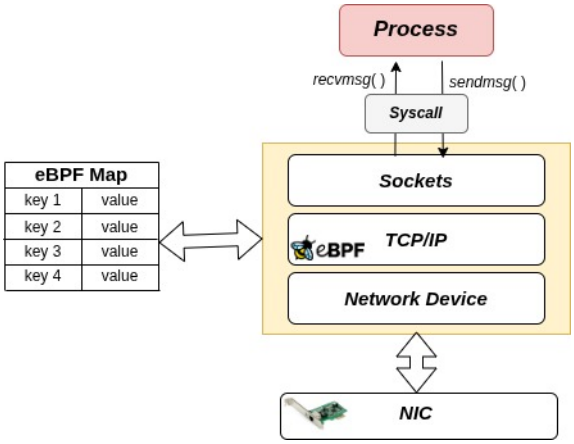

Figure 2. eBPF with maps running in kernel space.

The algorithm state is maintained per TCP socket using the *icsk_ca_priv* structure, which stores key Cubic variables such as the previous maximum cwnd, inflection point, ACK counters, timestamps, and phase-related parameters. In Flex-Cubic, these variables are managed within eBPF programs under BPF constraints (e.g., no unbounded loops and controlled memory access), using helper functions when needed for timing and safe operations.

The implementation follows standard TCP Cubic hooks. The *init* hook initializes the congestion window and internal state, the ACK handler updates cwnd based on the cubic function and elapsed time since the last loss, and the loss handler adjusts cwnd and updates the cubic reference point.

As illustrated in Fig. 2, *eBPF maps* enable dynamic parameter control at runtime, allowing modifications of growth strategies and algorithm coefficients without changing the source code. This facilitates experimentation, reproducibility, and adaptive behavior.

Overall, this approach demonstrates how eBPF can implement a full TCP congestion control algorithm integrated with the kernel via *struct_ops*. Rather than replacing the TCP stack, it extends it safely, preserving compatibility with network analysis tools and leveraging the kernel verifier to ensure secure and controlled execution. This enables efficient evaluation of new congestion control strategies without requiring kernel modifications.

3.3. Selective window reduction

Traditional TCP Cubic treats packet loss as a direct indication of congestion, promptly reducing the congestion window based on assumptions rooted in wired networks, where losses are primarily caused by buffer overflows. In modern networks, however, packet losses may also result from random bit errors, transient effects, or link impairments unrelated to persistent congestion, making immediate window reduction inefficient and potentially harmful to throughput.

To address this limitation, using a temporal threshold based on a multiple of the RTT before reducing the congestion window in the *LOSS* state serves as a filtering mechanism that distinguishes persistent congestion from transient or noise-induced losses. By requiring loss signals to persist over multiple feedback cycles, the algorithm improves the

reliability of congestion inference, reduces unnecessary cwnd oscillations, and preserves Cubic's growth dynamics. In Flex-Cubic, this limit is defined by looking at the RTT behavior, providing a balanced trade-off between responsiveness and stability, particularly in high-granularity eBPF-based implementations.

3.4. TCP Cubic association with eBPF

A basic TCP Cubic code implements the congestion control algorithm using eBPF through *struct_ops* can be found in [Wibowo E.and Bi 2025]. The core logic of the algorithm remains aligned with traditional TCP Cubic, including *slow start*, *HyStart*, cubic growth of cwnd, TCP *friendliness*, *fast convergence*, and loss handling, but the implementation becomes programmable and extensible at runtime.

In the TCP Flex-Cubic, the main structural change is in the fundamental Cubic parameters, such as the reduction factor beta (β), the cubic scaling constant (C), and an RTT-related multiplier, are no longer compile-time constants embedded in the kernel. Instead, they are retrieved dynamically from an eBPF map. This design allows the behavior of Flex-Cubic to be tuned at runtime without recompiling or rebooting the kernel, making it well suited for experimentation and dynamic adaptation controlled from user space.

The Flex-Cubic also introduces a set of auxiliary functions dedicated to compute and studies RTT metrics, including the historical minimum RTT, absolute and relative RTT variation, and detection of significant RTT inflation. These functions do not directly modify the congestion window, but they provide additional instrumentation that is not present in standard Cubic, enabling future hybrid policies that combine loss-based and delay-based signals.

The key difference lies in how loss events are handled. In TCP Cubic, any loss resets the Cubic state, whereas in Flex-Cubic this occurs only if the loss is accompanied by significant RTT inflation relative to the minimum RTT (minRTT) maintained by the TCP kernel. Using minRTT as a reference provides a stable baseline and reduces sensitivity to transient fluctuations, making losses not clearly associated with persistent congestion, such as random losses or microbursts, less likely to trigger aggressive window reductions.

The RTT inflation threshold (e.g., 2×minRTT) was defined based on preliminary experiments evaluating factors from 2× to 5×, with no performance gains beyond 2×. Higher thresholds delayed congestion detection, increasing retransmissions, buffer buildup, and packet drops, and reducing the ability to distinguish between sporadic losses and actual congestion. Thus, 2×minRTT was selected as a trade-off between responsiveness and stability.

In burst loss scenarios, however, only the first loss may exhibit RTT inflation, while subsequent losses may not, leading to partial misclassification of losses as non-congestion-related. This highlights a trade-off of RTT-based filtering: improved robustness to random losses at the cost of reduced sensitivity to correlated loss bursts. Further analysis is left for future work.

In summary, Flex-Cubic does a differentiated congestion control algorithm on TCP Cubic using eBPF. It preserves the original semantics while adding dynamic configurability, improved timing resolution, and a loss reaction conditioned on clear RTT-based evidence of congestion. This makes Flex-Cubic a flexible foundation for research, exper-

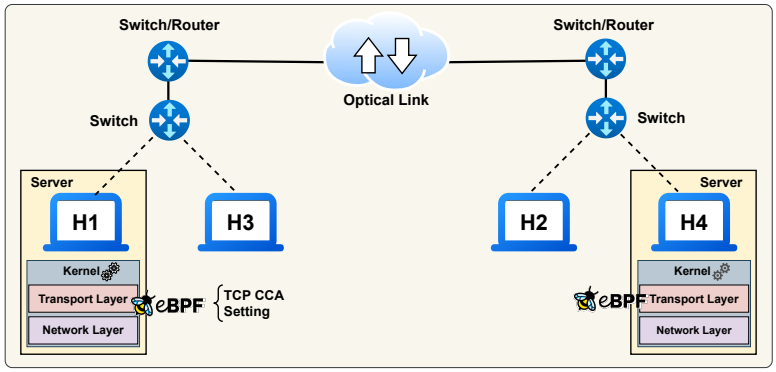

Figure 3. Topology adopted and results of competition between two (partially overlapping) flows using standard TCP Cubic vs. Flex-Cubic with losses and queue bottlenecks.

imentation, and fine-grained tuning in dynamic network environments.

4. Validation Scenarios

For simplicity, a Dumbbell topology was adopted, with four hosts (H1-H2 and H3-H4 pairs), access and core switches Fig. 3. This configuration supports to evaluate the behavior of the TCP Cubic and Flex-Cubic algorithms when competing for bandwidth resources under network links with different one-way delay (OWD) values (10, 25, and 50 ms), packet loss rates ranging from 0% to 2% (introduced via *tc*—traffic control) to emulate losses caused by corrupted packets at physical layer, and throttling of intermediate node buffer queues to values ranging from 100% to 25% of the bandwidth delay product (BDP) in order to intensify the response to packet overload. Here, the values of delays, and consequently values of minimum RTT (20, 50, and 100 ms) were selected based on [Chen et al. 2020]. Each experiment involves two concurrent long-lived TCP flows sharing the bottleneck link, allowing controlled observation of bandwidth competition under identical configurations, and no background traffic was added to focus on TCP behavior under bit-error-induced losses. The experiment focused on the direct comparison between the standard TCP Cubic from the Linux kernel and the Flex-Cubic algorithm proposed in this work. Comparisons involving standard Cubic with BBR, and other classic algorithms can be found in [Pedro Filho et al. 2025].

The network emulation was conducted on a server running 64-bit Linux Ubuntu 22.03 (*kernel 6.8.1*), *bpftool v7.4.0*, with 48 GB RAM, a 12-core processor, and Mininet 2.3.1b4, highlighted in Fig 3. An important aspect concerns the potential overhead introduced by the eBPF-based implementation. Although eBPF programs are executed in-kernel with JIT compilation, additional costs may arise from map accesses and the indirection introduced by *struct_ops*. Throughout the experiments, average CPU usage remained limited to between 5 and 10%, suggesting that the overhead is limited in the evaluated scenarios. However, a detailed analysis of execution cost, including per-packet processing overhead and eBPF map access latency, is outside the scope of this study. A systematic evaluation comparing native kernel implementations and eBPF-based CCAs in terms of CPU usage, throughput, and latency could be part of future work.

The behavior of TCP regarding the congestion window dynamics is strongly influenced by the parameters β and C of Equations (1) and (2), which jointly govern the

response to loss events and the evolution of the window during the congestion avoidance phase. The parameter β defines the multiplicative decrease factor applied to the cwnd after a congestion event (Equations 1 and 2). In standard Cubic, $\beta \approx 0.7$ results in a moderate reduction of the congestion window, favoring fast throughput recovery in links with a high bandwidth–delay product (BDP). Values lower than the standard lead to an excessively aggressive reaction to losses, causing abrupt reductions in the cwnd and penalizing the average throughput, particularly in scenarios with non-congestive losses. Conversely, for higher values ($\beta > 0.7$), the reduction becomes less severe, resulting in a more aggressive sending behavior.

The parameter *C* controls the constant of the cubic function is responsible for the growth of the cwnd over time. Higher values of *C* produce a faster and more aggressive increase of the congestion window, allowing the flow to return more quickly to its previous maximum value. In contrast, reducing *C* smooths the cubic growth curve, delaying the recovery of the cwnd.

The joint tuning of these parameters enables TCP Cubic to be adjusted toward more or less aggressive behavior, according to the constraints and operating conditions imposed by the network. Scripts are available on github [1].

5. Results and Discussions

Figures 4 to 6 summarize the averaged results of 150 `iperf3` runs, with metrics statistically aggregated. The reported values correspond to mean throughput, with variability indicated by error bars, reflecting the dispersion observed in the time-series data. The results provide a comprehensive comparison between standard TCP Cubic and the proposed Flex-Cubic under varying network conditions, including packet loss rates (0–2%), one-way delay (OWD = 10, 25, and 50 ms), and buffer sizes ranging from 25% to 100% of the BDP, for different values of the multiplicative decrease factor β.

In the presented results, β values both lower ($\beta = 0.3$) and higher ($\beta = 0.9$) than the standard Cubic ($\beta = 0.7$) were evaluated to explore the impact of aggressiveness in cwnd adjustment. The parameter C was kept constant ($C = 41$), as preliminary observations indicated limited sensitivity to its variation within the evaluated scenarios. Although the parameter C theoretically controls the growth rate of the congestion window, experimental results indicated limited sensitivity to its variation under the evaluated conditions, suggesting that other factors (e.g., β and RTT variation) play a more dominant role.

Overall, Flex-Cubic generally achieves higher average throughput than standard TCP Cubic across most combinations of OWD, loss rate, and queue size. This trend is particularly evident under moderate loss conditions, where Cubic exhibits a pronounced degradation in throughput even at relatively low loss rates (e.g., 0.5%). In contrast, Flex-Cubic maintains higher and more stable throughput levels, suggesting that the RTT-conditioned loss reaction mechanism effectively avoids unnecessary congestion window reductions in the presence of non-congestion-related losses. However, in some scenarios, particularly under low-loss or low-delay conditions, the overlap between confidence intervals indicates that the observed differences may not always be statistically significant.

The impact of RTT is particularly evident in the results. As the OWD increases

[1] https://github.com/nerds-ufes/Flex-Cubic.git

from 10 ms to 50 ms, the performance of TCP Cubic degrades substantially, especially under moderate to high loss rates. This behavior is consistent with the known sensitivity of loss-based algorithms to the bandwidth-delay product (BDP), where recovery from congestion window reductions becomes increasingly slow. In contrast, Flex-Cubic demonstrates strong resilience to increasing RTT, sustaining higher throughput even in high-delay scenarios. For instance, at OWD = 25 ms and 0.5% loss, Flex-Cubic achieves substantially higher throughput than Cubic, in some cases by several multiples, highlighting the impact of misinterpreting non-congestion losses as congestion signals in high-BDP environments.

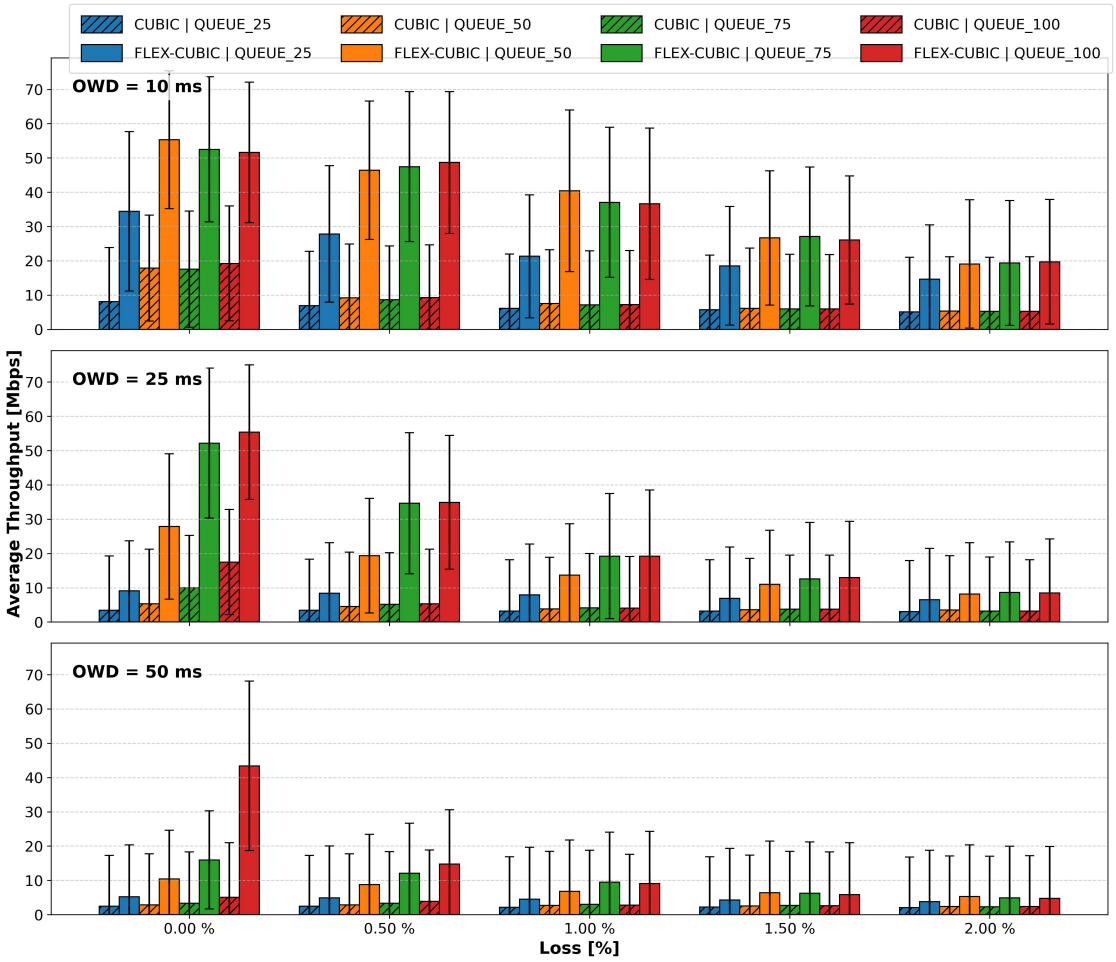

Figure 4. Throughput comparison between Cubic and Flex-Cubic ($\beta = 0.3$,$C = 41$)

Buffer size also plays a relevant role in performance. Larger buffers (75% and 100% of BDP) generally enable higher throughput by allowing greater congestion window growth. However, while TCP Cubic often fails to effectively utilize the available buffer due to frequent and premature window reductions, Flex-Cubic is able to better exploit the available capacity, maintaining higher sending rates even under adverse conditions.

The evaluation of different β values reveals important trade-offs between aggressiveness and stability. The configuration with $\beta = 0.7$ consistently provides the best over-

all performance, achieving higher average throughput while maintaining relatively stable behavior. Lower values, such as $\beta = 0.3$, result in a more conservative response due to stronger multiplicative decrease, leading to slightly reduced throughput but improved stability in some scenarios. Conversely, higher values such as $\beta = 0.9$ tend to increase aggressiveness, which may introduce higher variability, particularly under elevated loss conditions. These results indicate that β plays a critical role in balancing responsiveness and robustness in Flex-Cubic.

As packet loss increases, the conservative nature of TCP Cubic becomes more evident. Even under relatively low loss rates, Cubic exhibits a pronounced reduction in throughput due to its strictly loss-driven congestion response, often leading to underutilization of available capacity. In contrast, Flex-Cubic shows a more gradual degradation, maintaining higher average throughput across most scenarios. Nevertheless, the increased variability observed in some configurations suggests the presence of transient effects, such as retransmission bursts or dynamic adaptation of the congestion window.

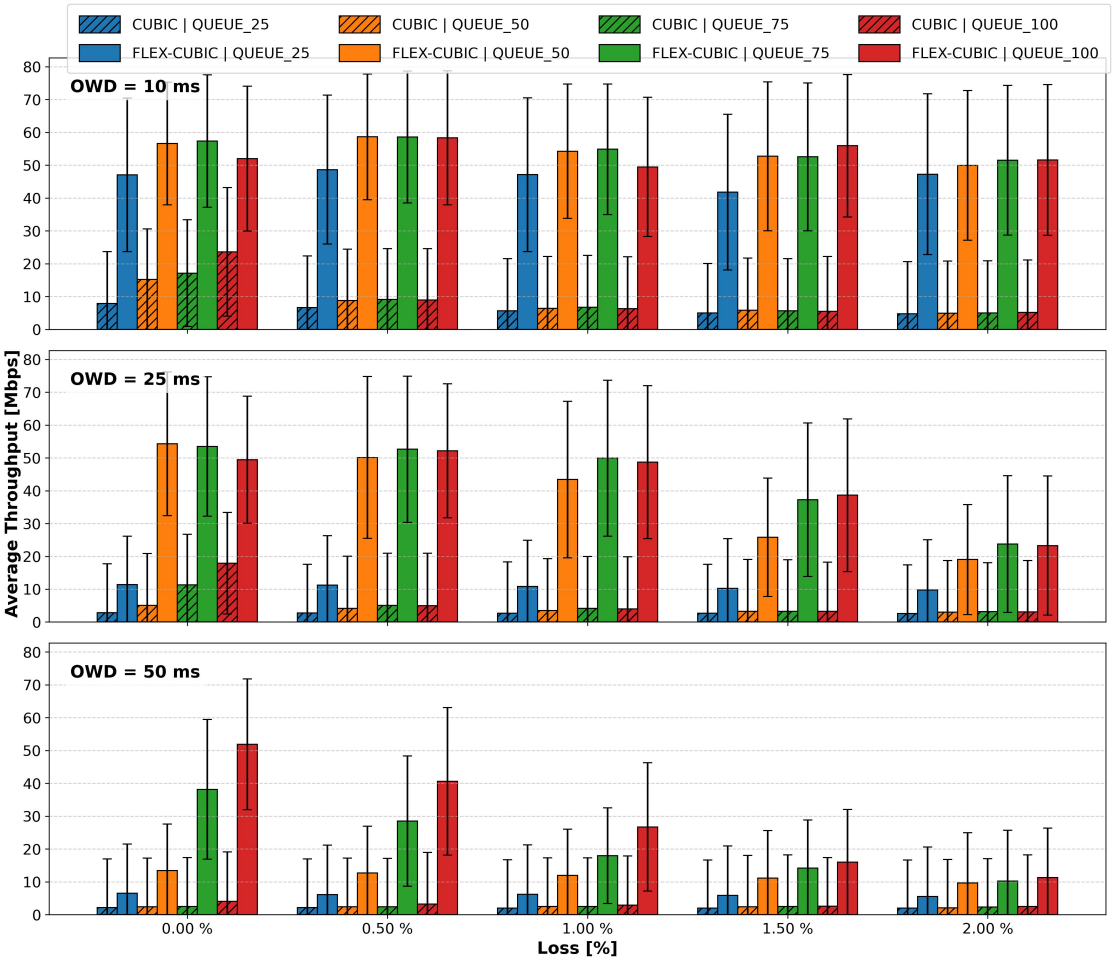

Figure 5. Throughput comparison between Cubic and Flex-Cubic ($\beta = 0.7$,$C = 41$)

The throughput gains observed with Flex-Cubic are achieved through an RTT-conditioned congestion control strategy, which delays congestion window reduction until persistent congestion is detected. While this improves bandwidth utilization, it also

increases retransmissions, suggesting a trade-off between efficiency and loss tolerance. Although retransmission counts are not explicitly reported, preliminary observations indicate a consistent increase compared to TCP Cubic, motivating future evaluation using goodput as a complementary metric. Overall, these results indicate that Flex-Cubic mitigates key limitations of TCP Cubic by incorporating RTT-based decision mechanisms, enabling improved performance in high BDP and loss-prone environments while providing a tunable balance between aggressiveness, stability, and retransmission overhead.

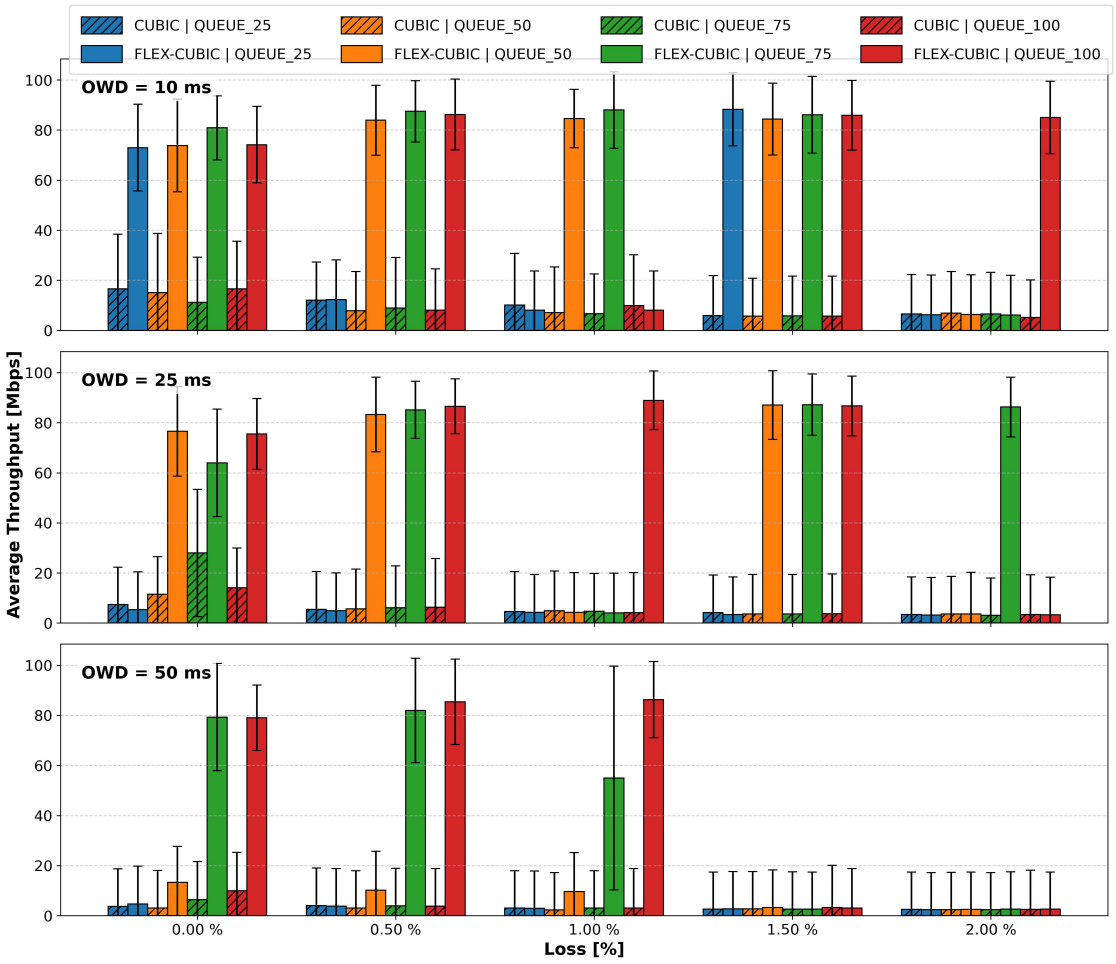

Figure 6. Throughput comparison between Cubic and Flex-Cubic ($\beta = 0.9$,$C = 41$)

A limitation of this study is the adopted loss model, where packet losses are injected randomly using the Linux tc/netem module. While this enables controlled and reproducible experiments, it does not capture the bursty and correlated loss patterns typical of optical, FSO, or satellite networks, which may affect congestion control behavior under consecutive loss events. However, random losses allow isolating the algorithm's ability to distinguish between congestion-induced and non-congestion-induced losses, which is the main focus of this work. Thus, the results should be interpreted as a baseline, and future work will consider more realistic bursty loss models.

Another limitation is the absence of a fairness analysis under heterogeneous congestion control algorithms. Although Flex-Cubic and TCP Cubic are evaluated under

identical conditions, scenarios with mixed CCAs were not explored. Since fairness is critical in shared networks, especially when different control strategies compete for bandwidth, a more comprehensive evaluation including mixed flows, larger-scale scenarios, and fairness metrics is left for future work.

6. Conclusion

This work presented TCP Flex-Cubic, a CCA derived from TCP Cubic and enhanced with eBPF features support toward runtime adaptability. TCP Flex-Cubic may dynamically follow communication link conditions and, when compared to traditional TCP Cubic, offers the additional capability to differentiate between packet losses caused by queue overload at intermediate nodes and losses induced by physical layer impairments effects. This work extends our previous studies on the dynamic adaptability of TCP CCAs to physical-layer constraints and the requirements of loss-sensitive applications, such as Data-Intesive Science (DiS), leveraging additional metrics toward cross-layer approaches.

The integration of the maps eBPF tool with the TCP congestion control algorithm (CCAs) was discussed in details, offering an alternative to the static algorithms traditionally implemented in the Linux kernel. This integration allows for dynamic interaction between runtime metrics and CCA control variables, representing a substantial evolution of TCP by enabling adaptation to networks with non-static characteristics, including satellite, FSO, and optical networks with disjoint paths and heterogeneous properties.

Future works will progress toward larger scale tests for Flex-Cubic with multi-hop flows competing with background traffic and under physical layer reconfiguration toward TE studies, which will include Cubic parameter optimization according to current perceived network states.

References

Antelmi, A. and Carlini, E. (2026). Large-scale hpc approaches and applications on highly distributed platforms.

Borges, E. S., Ribeiro, M. R., Guimaraes, R. S., Xavier, B. M., Pecolo, P., Dominicini, C. K., Martinello, M., and Rufini, M. (2024). Fso-based reconfigurable optical networks: Source routing for decoupling multilayer te.

Chen, X. et al. (2020). Measuring tcp round-trip time in the data plane. In *Proceedings of the Workshop on Secure Programmable Network Infrastructure*, SPIN '20, page 35–41, New York, NY, USA. Association for Computing Machinery.

Chen, Z., Meng, Q., Lao, C., Liu, Y., Ren, F., Yu, M., and Zhou, Y. (2025). {eTran}: Extensible kernel transport with {eBPF}. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pages 407–425.

Chou, J. and Chung, W.-C. (2024). Cloud computing and high performance computing (hpc) advances for next generation internet.

Ha, S., Rhee, I., and Xu, L. (2008). Cubic: A new tcp-friendly high-speed tcp variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74.

Hinz, J.-T., Addanki, V., Györgyi, C., Jepsen, T., and Schmid, S. (2023). Tcp's third eye: Leveraging ebpf for telemetry-powered congestion control. In *Proceedings of the 1st Workshop on eBPF and Kernel Extensions*, pages 1–7.

Jadin, M. et al. (2022). Leveraging ebpf to make tcp path-aware. *IEEE Transactions on Network and Service Management*, 19(3):2827–2838.

Lai, Z. et al. (2025). Leocc: Making internet congestion control robust to leo satellite dynamics. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 129–146.

Magnani, S., Risso, F., and Siracusa, D. (2022). A control plane enabling automated and fully adaptive network traffic monitoring with ebpf. *IEEE Access*, 10:90778–90791.

Mathis, M., Semke, J., Mahdavi, J., and Ott, T. (1997). The macroscopic behavior of the tcp congestion avoidance algorithm. *SIGCOMM Comput. Commun. Rev.*, 27(3):67–82.

Miano, S., Bertrone, M., Risso, F., Tumolo, M., and Bernal, M. V. (2018). Creating complex network services with ebpf: Experience and lessons learned. In *2018 IEEE 19th International Conference on High Performance Switching and Routing (HPSR)*, pages 1–8. IEEE.

Pan, W., Xu, Y., Wang, C., and Wu, J. (2024). An ebpf-empowered congestion control system with delay requirements. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3875–3880. IEEE.

Pecolo F, P. P., Ribeiro, M. R. N., Borges, E. S., and Martinello, M. (2025). On the interplay of congestion and osnr degradation for tcp over optical networks. In *2025 SBMO/IEEE MTT-S International Microwave and Optoelectronics Conference (IMOC)*, pages 650–654.

Pedro Filho, P., Togneri, A. P., Ribeiro, M. R., Segatto, M. E., Borges, E. S., and Martinello, M. (2025). Tcp/ipodwdm: A case for adaptive congestion control in optical networks under physical layer impairments. In *Workshop de Pesquisa Experimental da Internet do Futuro (WPEIF)*, pages 25–32. SBC.

Song, L. and Li, J. (2024). ebpf: Pioneering kernel programmability and system observability-past, present, and future insights. In *2024 3rd International Conference on Artificial Intelligence and Computer Information Technology (AICIT)*, pages 1–10. IEEE.

Wibowo E.and Bi, B. (2025). Evaluating ebpf as a platform for congestion control algorithm implementation. https://github.com/ebpfcca/ebpfcca.

Yang, S., Tang, Y., Pan, W., Wang, H., Rong, D., and Zhang, Z. (2023). Optimization of bbr congestion control algorithm based on pacing gain model. *Sensors*, 23(9):4431.

Zadeh, S. A. et al. (2023). On augmenting tcp/ip stack via ebpf. In *Proceedings of the 1st Workshop on eBPF and Kernel Extensions*, pages 15–20.

Zanotelli, V. F., Pontes, E. C., Martinello, M., Ros-Giralt, J., Borges, E. S., Comarela, G., Ribeiro, M. R. N., and Newman, H. (2025). Transport efficiency for data-intensive science: Deployment experiences and bottleneck analysis. *Annals of Telecommunications*, 80(9):793–805.