

SpectraCCC: Passive Classification of TCP Congestion-Control Algorithms

Jonas Krause*, Max Franke*, Oliver Hohlfeld[†] and Stefan Schmid*

*TU Berlin, [†]Uni Kassel

Abstract—The identification of TCP Congestion Control Algorithms can provide valuable insights for optimizing network performance, fairness, and stability. However, little is currently known about the extent to which TCP algorithms can be identified through passive observations alone. Working purely passively on time-series data captured through the Berkeley Extensible Software Switch, we first show that Dynamic Time Warping with a k-Nearest-Neighbor classifier can already distinguish 14 Congestion Control Algorithms without inducing packet loss with up to 59% accuracy. We then introduce *SpectraCCC* which performs a lightweight hierarchical cascade: it first groups Congestion Control Algorithms by packet transmission behavior and then refines classification through acknowledgment dynamics, raising overall accuracy from 59 % to 66 %. Notably, it achieves 90%–100% accuracy for Congestion Control Algorithms that earlier passive studies ignored, including HTCP, CDG, and Westwood. Beyond *SpectraCCC*, our study provides deeper insights into congestion control behavior by visualizing distinct transmission patterns and highlighting the efficacy of BESS monitoring features.

I. INTRODUCTION

Identifying TCP Congestion Control Algorithms (CCAs) is a relevant network observability problem with direct implications for performance, fairness, and stability. CCAs govern sending rates to balance throughput and congestion, shaping end-to-end behavior. As CDNs and network providers increasingly deploy custom TCP stacks, visibility into deployed CCAs is essential for assessing fairness across competing flows [1], [2]. Yet, fairness evaluations often assume homogeneous CCAs, making it hard to attribute performance differences to either network conditions or protocol behavior. Passive CCA inference thus plays a vital role—not only for troubleshooting but also for enabling Internet-scale studies of CCA deployment and evolution.

Having a mix of different CCAs present also causes several problems for network operators. For one, unfair bandwidth sharing among different algorithms is becoming an issue, especially with the rise of BBR, which has been shown to dominate loss-based CCAs and inflate latency for competing flows [3], [4]. Beyond fairness, heterogeneous CCAs complicate troubleshooting and performance attribution: when flows experience degraded throughput or increased delay, operators cannot easily distinguish whether the root cause lies in network conditions or in aggressive or non-standard congestion control behavior [1]. In practice, operators often lack reliable visibility into which CCAs are deployed. Configuration-based approaches require access to end hosts, which is typically unavailable in ISP, IXP, or multi-tenant data center environ-

ments, and even when accessible, CCAs may differ per socket or changed dynamically at runtime. Additionally, the growing deployment of custom TCP stacks and vendor-specific variants further limits the usefulness of host-side logs for network-wide monitoring [5]. These challenges motivate on-path CCA identification methods that infer congestion control behavior directly from observed traffic, without requiring host cooperation.

Existing work on CCA identification falls into two broad categories: *active* and *passive* approaches. The bulk of existing methods are *active*, meaning they deliberately manipulate network parameters—such as by injecting delays or losses—to provoke characteristic reactions from CCAs [6]. While these methods can achieve high classification accuracy [7], they inherently disrupt normal traffic and are unsuitable for studying productive traffic at an ISP or data center. *Passive* approaches, in contrast, rely solely on observing naturally occurring traffic patterns without modifying the flow. This makes them well-suited for transparent, scalable deployment. However, passive identification methods are much less explored and face significant limitations. Existing passive techniques typically support only a small number of CCAs—often limited to three—and some rely on non-explainable deep learning models, such as in [8], which hampers interpretability and trust in their results. As a consequence, there remains a substantial gap in our ability to passively and reliably infer the wide range of modern congestion control algorithms deployed in the wild.

This paper introduces a passive method with explainable feature set for classifying 14 CCAs. Our approach leverages time-series data passively collected through the Berkeley Extensible Software Switch (BESS) [9], applying Dynamic Time Warping (DTW) and 1-Nearest Neighbor (1-NN) classification. From 28 candidate features analyzed, we identify packet- and acknowledgment (ACK)-count series as most discriminative (up to 59 % accuracy). Building on this insight, we design *SpectraCCC*, a hierarchical two-stage classifier that first groups traffic series by packet transmission patterns and then refines the decision with ACK-dynamics, raising the overall accuracy to 66 % and reaching 90–100 % for previously unclassified variants such as HTCP, CDG, and Westwood. Widely used CCAs such as BBR and Reno are identified with 100 % accuracy by *SpectraCCC*.

Our contributions are twofold.

SpectraCCC: We present *SpectraCCC*, a fully passive method for inferring TCP congestion control algorithms (CCAs) based on a hierarchical classifier using Dynamic Time

Warping and k-NN. Unlike prior passive methods, which distinguish at most 3 CCAs, *SpectraCCC* accurately identifies 14 different TCP CCAs without inducing packet loss. This enables *SpectraCCC* to monitor operational traffic at ISPs or data centers. We will also release the source code of *SpectraCCC* to ensure reproducibility and as a contribution to the research community.

Explainable Features: Through a systematic sensitivity analysis of 28 switch-level features, we identify 5 key features most relevant for CCA classification. These features are interpretable and tied to core CCA dynamics, setting our approach apart from black-box deep learning models.

Our results demonstrate that passive monitoring—previously limited to only a few CCAs—can reliably recognize a far wider range of CCAs in a non-intrusive manner.

II. RELATED WORK

Prior work on CCA identification falls into two methodological categories: active and passive approaches. Active methods deliberately manipulate network conditions—e.g., by injecting delays or losses—to trigger distinctive CCA behavior. In contrast, passive methods rely solely on observing naturally occurring traffic, making them suitable for unobtrusive monitoring at ISPs or data centers.

Active approaches. Yang et al. [6] emulate delay and loss against public web servers, identifying 14 CCAs with 97% accuracy. Their method is inherently active, as it manipulates live flows to elicit reactions. Mishra et al.’s Great Internet TCP Congestion Control Census [5] scales this idea to 20 000 Alexa sites, identifying vendor-specific variants like Google’s BBR G1.1 by injecting loss and shaping bandwidth in controlled bottlenecks. Similarly, Sonoda et al. [10] detect Reno and Vegas at a Wi-Fi access point by actively altering MAC-layer parameters at a Wi-Fi access point to observe flow behavior under induced differentiation. Inspector Gadget [11] crafts bespoke connections where ACKs are withheld and RTT is varied to reveal *cwnd* dynamics, classifying eight CCAs with high accuracy through targeted perturbations. CCAnalyzer [12] inserts a controlled bottleneck queue and matches occupancy traces via Dynamic Time Warping, achieving 100 % accuracy on 15 CCAs, leading to artificial packet loss. Nebby [13] targets TCP and QUIC flows by actively delaying ACKs and imposing bottlenecks, identifying 13 CCAs across Alexa top sites with 96.7% accuracy.

Passive approaches. In contrast, passive methods observe traffic without injecting artifacts, making them suitable for production monitoring. Oshio et al. [7], [14] use *cwnd* dynamics inferred from packet timing to distinguish between Reno, Highspeed, and similar CCAs, using clustering on features extracted at a passive relay. Nogiwa and Hirata [15] transform inter-packet jitter into images and apply CNNs to distinguish Cubic and Compound TCP with over 95% accuracy. DeepCCI [8] combines CNN and Long Short-Term Memory layers on packet-arrival histograms, identifying Reno, Cubic, and BBR with F1 scores exceeding 90 %. The

DeepCCI’s approach is purely passive, as it relies only on timestamping packets already traversing the link. However, its deep learning architecture is non-explainable, making the basis of classification difficult to assess. Kfoury et al. [16] use a P4 switch to embed per-flow passive metrics like bytes-in-flight and queueing delay, distinguishing only two broad classes: loss- vs. model-based. Dragonfly [17] uses eBPF to monitor router queue occupancy and classifies three CCAs with an F1 score of up to 85%, based on ingress throughput and queue length.

Research Gap. Active approaches dominate the field and offer high accuracy, but are unsuitable for passive monitoring of production traffic. Passive techniques remain scarce, typically limited to identifying two or three CCAs, and in some cases rely on non-transparent models. With *SpectraCCC*, we aim at closing this gap.

III. SPECTRACCC SYSTEM DESIGN

In this section, we outline our overall system design by first presenting the approach (Section III-A) and then describing its implementation (Section III-B).

A. Approach

In this work, we propose a passive approach to classify the CCAs used by a TCP flow entirely on data passively captured at a switch (in our case, BESS [9]) as vantage point. We chose a switch as a convenient way to passively obtain TCP-level statistics at the data plane (Figure 1) without requiring host-support, which is relevant for operational monitoring at ISPs or data centers.

Although many congestion control algorithms only expose their most distinctive behavior under packet loss, *SpectraCCC* does not rely on inducing congestion or observing loss events. It instead leverages differences in dynamics that remain observable under passive, loss-free operation. These include startup behavior, bandwidth probing patterns, ACK pacing, and delay sensitivity, all of which influence packet and acknowledgment rate time series even when the network is lightly loaded. For example, model-based algorithms such as BBR exhibit characteristic rate plateaus and periodic probing, while hybrid or delay-sensitive CCAs (e.g., Illinois or YeAH) show stronger short-term variability in response to inferred delay. We note that in fully underutilized paths, multiple CCAs may converge to similar behavior, which fundamentally limits passive identifiability. Many operational networks however experience contention, startup phases, or mild queueing, during which these algorithmic differences become visible without active perturbations. *SpectraCCC* is therefore best viewed as a complementary observability tool that infers CCA behavior when natural traffic dynamics provide sufficient signal, rather than as a replacement for active classification methods.

Measurements and Datasets. We begin by recording all features that we can obtain from an open-source BESS [9] switch as an initial feature set, which will then be reduced to only relevant features in our sensitivity study. In this regard, we capture 28 features including packet/ACK counts,



Fig. 1. Measurement Setup. The client sends TCP data to the server through BESS. The server sends ACK packets to the client through BESS. The switch exports 28 metrics as time-series.

queue statistics and packet latencies. For each feature, we obtain 8 time series per CCA. Each time series contains 40 measurements, with each measurement being recorded once every 500 ms. We take three time series per CCA for the training dataset and five for testing. This granularity produces a behaviorally rich yet easily traceable data volume, giving enough resolution to expose the distinctive dynamics of each CCA while keeping the dataset compact.

Baseline classification. In the next step, we need to assess the similarity of each time-series data set. A high similarity indicates that the same CCA variant is being used, while a low similarity suggests a different variant. We measure similarity between series with DTW [18], [12]. For every feature, we compute the DTW distance from a test series to all training series and assign the label of the 1-nearest neighbor.

To clarify why we chose DTW, Figure 2 contrasts it with the simpler Euclidean distance (ED). ED enforces a rigid one-to-one alignment.

$$ED(X, Y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2} \quad (1)$$

Here, $X = (x_1, x_2, \dots, x_n)$ and $Y = (y_1, y_2, \dots, y_n)$ are two time series of equal length n . This method assumes a strict one-to-one mapping of points, making it sensitive to temporal shifts. In contrast, DTW finds an optimal alignment by allowing flexible pointwise comparisons. The DTW distance is computed recursively as:

$$DTW(i, j) = d(x_i, y_j) + \min \begin{cases} DTW(i-1, j) \\ DTW(i, j-1) \\ DTW(i-1, j-1) \end{cases} \quad (2)$$

Here, $d(x_i, y_j) = |x_i - y_j|$ represents the absolute difference between two points.

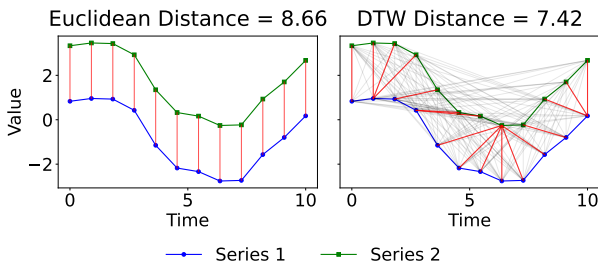


Fig. 2. ED pairs points at identical times; DTW warps the timeline and yields a shorter cumulative distance.

We then infer CCA variants by clustering the DTW distances with a 1-Nearest Neighbor (1-NN) classifier.

SpectraCCC: Hierarchical classification. Based on this, we introduce *SpectraCCC*, a two-stage DTW cascade. The two-stage approach enables us to exceed the $\leq 59\%$ accuracy of single features (see § V-C). In § V-E we show that this hierarchical approach is the key insight to enable *SpectraCCC* to achieve high classification accuracy.

- 1) **Stage I: Broad Classification.** We compare a *packet-transmission* time-series against the training set with DTW. The 1-NN label partitions series into groups that share similar sending patterns (shown in Figure 3).
- 2) **Stage II: Fine-Grained Classification.** Inside the groups we repeat DTW, now on *ACK-dynamics* series. This pass distinguishes the remaining candidates. ACK dynamics reflect how the sender reacts to network feedback, enabling Stage II to distinguish CCAs with similar packet-rate behavior.

The lightweight hierarchy in *SpectraCCC* provides a refined decision process that improves classification precision while remaining computationally efficient and fully passive.

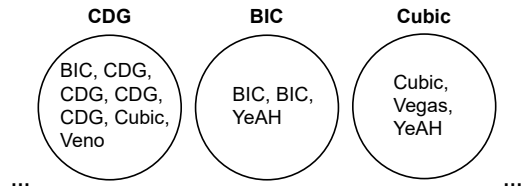


Fig. 3. Broad classification. Each test instance is assigned to a group based on minimal DTW distance. A group is defined by the predicted CCA label assigned to the test instances within this group. The second-stage refines the classification within these groups.

B. Implementation

All evaluation measurements run inside one virtual machine. Two Docker containers—sender and receiver—are attached to a BESS [9] switch configured in user space with DPDK support. The sender acts as an *iperf3* client and cycles through the 14 Linux CCAs, whereas the receiver runs a persistent *iperf3* server. The BESS queue is configured with a depth of 1024 packets, large enough to avoid overflow during our experiments. This design guarantees passivity: the switch never delays, duplicates, or discards traffic. Our testbed is dimensioned such that no queue overflows occur. This is confirmed by `DROPPEDPKTS = 0` for all measurements (see § IV), ensuring strictly loss-free conditions.

IV. SENSITIVITY STUDY

To identify the relevant features for CCA classification, we conduct a sensitivity study across the 28 time-series features exported by BESS. The goal of this study is to reduce this set to the most informative features without sacrificing accuracy.

For clarity, we group the 28 raw features into four functional categories based on their originating BESS module (Table I). This abstraction helps to summarize the feature

space succinctly, avoiding unnecessary low-level detail. The *Show Module* and *Show Port* count packets and ACKs at different positions in the BESS pipeline. *Queue Status* exposes the numbers of enqueued, dequeued, and dropped packets. Finally, *Measure* derives delay-domain features that capture latency and jitter.

We then empirically evaluate each feature’s relevance and identify the 5 most important for CCA identification:

- **VPORT0PKTS, PORTOUTACKS, PORTINACKS**: three packet/ACK-rate features that reached the highest single-feature accuracies out of 28 (59 %, 57 %, 54 %). They stem from the *Show Port* and *Show Module* functions.
- **MAXLATENCY**: the strongest feature measured with the *Measure* module (21 % accuracy). Although it is far below the top performers, it injects information about path delay that rate counters do not provide.
- **DROPPEDPKTS**: the queue-drop counter (7.1 %) is deliberately included as a sanity baseline: its constant value 0 confirms that the experiment remains fully passive and loss-free.

Together, these five features provide a compact yet expressive view of congestion-control behavior and form the feature set analyzed in depth in our evaluation.

V. EVALUATION

We evaluate CCA inference across five aspects. The first four analyses provide empirical insights that inform our method: § V-A packet transmission behavior, § V-B ACK transmission behavior, § V-C feature-level classification accuracy, and § V-D the relationship between CCAs and specific features. These findings guide the design of *SpectraCCC*, whose effectiveness we assess in § V-E by measuring the performance gain from our hierarchical classification strategy.

A. Packet Transmission Behavior

Figure 4 shows the average packets seen at the virtual BESS ingress port (VPORT0PKTS).

Illinois is by far the most aggressive—peaking at ≈ 4000 packets by measurement 32 before collapsing. This can be observed due to the hybrid nature of this CCA, which uses loss and delay-based methods [19]. BBR remains almost quiescent (around 0 – 2 packets per measurement). It controls the transmission power based on estimates of the available bandwidth and the minimum RTT, which can lead to a constantly low number of transmitted packets [20]. We also see that Westwood falls relatively sharply until measurement 3 (from 1900 to 1100 packets) and then levels off between

BESS function	# Features	Semantic focus
Show Module	16	Packet/ACK counts
Queue Status	3	Queue packet counts
Show Port	1	Port packet counts
Measure	8	Latency/jitter statistics

TABLE I

BESS FUNCTIONS AND EXPORTED FEATURES.

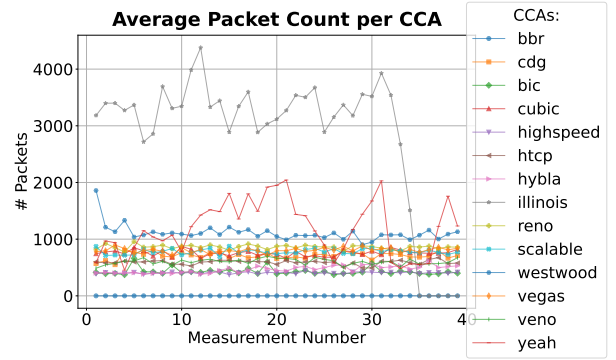


Fig. 4. Average packet count per CCA. The number of packets passing through the virtual BESS ingress port (VPORT0PKTS). 14 CCAs and their average time series measurements are shown here.

1000 and 1300 packets. It adjusts transmission power based on estimates of available bandwidth, which can lead to a rapid drop and subsequent stabilization of transmitted packets [21]. YeAH shows very high fluctuations between 600 and over 2000 packets. It combines loss and delay metrics, which can lead to variable transmission powers and the observed fluctuations [22]. The time series of the remaining CCAs appear relatively constant and behave very similarly at first glance.

These distinctive rate profiles are a first hint at separability and we use this feature in the first stage of our hierarchical approach in Section V-E.

B. Acknowledgment Transmission Behavior

Figure 5 shows the average number of ACKs observed at egress of the PortOut() ¹ module (PORTOUTACKS). The ranking largely mirrors the packet-rate plot (Figure 4). Because ACK cadence reflects how quickly a sender responds to feedback, it offers an orthogonal perspective.

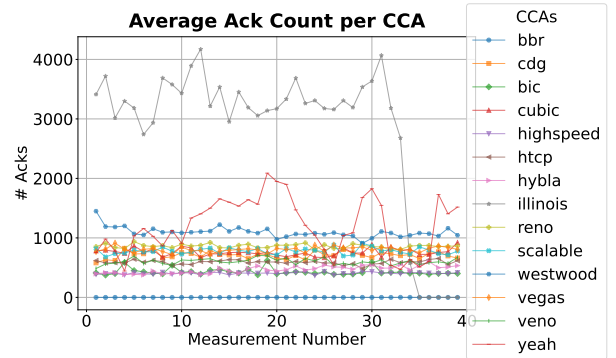


Fig. 5. Average ACK count per CCA. The number of ACKs processed in the PortOut module (PORTOUTACKS). 14 CCAs and their average time series measurements are shown here.

Most CCAs keep the same relative order—Illinois again dominates, BBR stays near zero—but subtle differences

¹<https://github.com/NetSys/bess/wiki/Built-In-Modules-and-Ports>

emerge. For instance, *Westwood*'s early rate drop now appears as a small matching ACK dip, confirming its rapid bandwidth re-estimation, while *YeAH*'s oscillations translate into equally erratic ACK bursts. We use this feature as the second-stage discriminator in our hierarchical classifier (Section V-E).

C. Feature Accuracies

Applying DTW-1NN to each feature in isolation yields the accuracies shown in Figure 6.

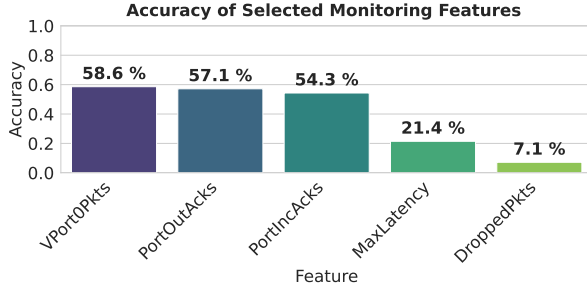


Fig. 6. Classification accuracy per feature. Packet- and ACK-count metrics (VPORT0PKTS, PORTOUTACKS, PORTINACKS) clearly outperform delay (MAXLATENCY) and loss (DROPPEDPKTS) counters.

VPORT0PKTS achieves the highest accuracy (58.6 %), followed by PORTOUTACKS (57.1 %) and PORTINACKS (54.3 %). MAXLATENCY reaches only 21.4 %, and DROPPEDPKTS matches the random baseline of 7.1 %: the queue never overflows in our loss-free testbed, hence the value is always zero for each time series measurement.

D. CCA-Feature Interplay

Figure 7 shows per-CCA accuracy for each feature.

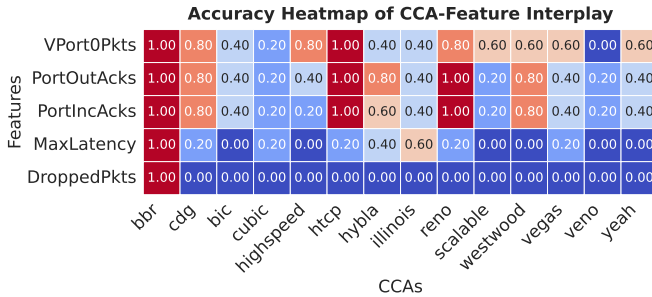


Fig. 7. Per-CCA accuracy across the five features.

PORTINACKS and PORTOUTACKS identify BBR, HTCP and Reno perfectly. VPORT0PKTS achieves 100 % accuracy only for BBR and HTCP. MAXLATENCY identifies only Illinois better than all other features (60 % accuracy). DROPPEDPKTS is uninformative (always zero). For DROPPEDPKTS the accuracy regarding BBR is 100 % and for all other CCAs 0 %, as each test instance is classified as BBR here. BBR can be recognized in all features with an accuracy of 100 %. Veno is generally difficult to recognize (maximum accuracy at 20 %). We also find that features can be

complementary: VPORT0PKTS scores 80 % on Highspeed vs. 40 % for PORTOUTACKS, whereas the roles flip for Hybla (40 % vs. 80 %). Such diversity motivates feature combinations.

Correlation analysis. A Pearson analysis (matrix omitted for space) shows that VPORT0PKTS and PORTOUTACKS correlate almost perfectly ($r \approx 1$) across all traces. Despite that correlation, their *in-flow* timing offsets carry complementary information—an insight exploited by *SpectraCCC* (Section V-E).

E. Hierarchical Classification

Guided by the analyses in Sections V-C and V-D, we combine the two most informative—yet complementary—features, VPORT0PKTS and PORTOUTACKS, in a two-stage cascade. In *SpectraCCC*, we use VPORT0PKTS in the first stage for broad classification, and PORTOUTACKS in the second stage for fine-grained refinement.

Overall gain. Relative to VPORT0PKTS (the best single feature), the two-stage cascade lifts *accuracy* from 59 % to 66 %, *precision* from 59 % to 72 %, *recall* from 59 % to 71 %, and the *F1* score from 56 % to 68 %.

Per-CCA view. Figure 8 details the outcome. Three CCAs – BBR, HTCP, and Reno – are recognized perfectly (100 %), whereas the weakest are Veno (33 %) and Highspeed (40 %). Compared to the strongest single-feature (VPORT0PKTS), we observe the following gains for nine CCAs: CDG (10 %), BIC (20 %), Cubic (26.7 %), Hybla (40 %), Illinois (30 %), Reno (20 %), Westwood (30 %), Veno (33.3 %) and YeAH (13.3 %). Only Highspeed (40 %) and Scalable (13.3 %) deteriorate, while the remaining three CCAs (BBR, HTCP, and Vegas) keep their single-feature peak.

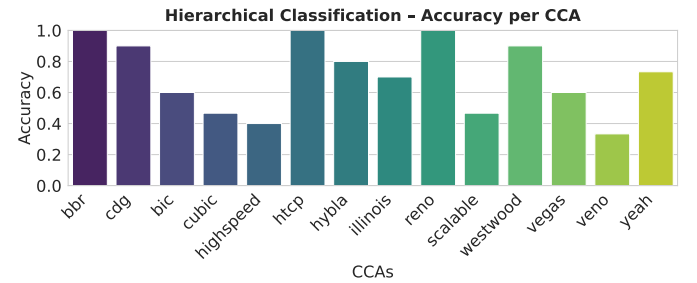


Fig. 8. Per-CCA accuracy of *SpectraCCC*. Relative to the strongest single feature, accuracy improves for nine CCAs, matches for three, and drops only for two.

Overall, the hierarchical approach increases the total accuracy to 66 % and delivers significant gains at CCA level for multiple algorithms. This two-stage classifier forms the core of our *SpectraCCC* framework.

F. Limitations

Our evaluation is conducted in a controlled testbed and should be understood as a proof-of-concept for passive CCA identification under ideal conditions. This setup allows us

to isolate congestion control behavior and study the limits of purely passive inference. It however does not capture the full complexity of operational networks, where traffic is heterogeneous and conditions can vary wildly over time. In real deployments, factors such as cross-traffic, variable RTTs, transient queueing, and random loss may partially obscure CCA-specific dynamics. That said, SpectraCCC relies primarily on rate-based features and short-term temporal patterns, which are known to be more robust to noise than delay- or loss-only signals, and Dynamic Time Warping explicitly tolerates temporal distortions in time-series data. As a result, we expect the approach to degrade gracefully rather than fail catastrophically as network conditions become more complex.

VI. CONCLUSION

This paper demonstrates that comprehensive CCA identification is feasible with fully passive observations. Using BESS, we analyze raw traffic to create time-series that a DTW-1NN baseline already classifies with up to 59 % accuracy. A central insight is that the two strongest features, packet rate (VPORTPKTS) and ACK cadence (PORTOUTACKS), though correlated in the aggregate, encode different short-term dynamics within a time series. Exploiting this complementarity in SpectraCCC, a two-stage hierarchical cascade, increases overall accuracy to 66 %, boosts the F1 score from 56 to 68 %, and yields 100 % accuracy for the widely used CCAs BBR, HTCP, and Reno. Additionally, we reach 90 %–100 % accuracy for algorithms that previous passive studies ignored, such as HTCP, CDG, and Westwood. Side-by-side visualizations of the two features used in SpectraCCC reveal these differences clearly and allow human operators to interpret flow behavior.

Future work could validate SpectraCCC in production network environments and extend it to other protocols like QUIC with minimal changes to the current setup. Additional features—such as CPU utilization or scheduling metrics—could be collected via existing BESS modules and integrated into the classification pipeline for richer, multi-dimensional analysis. Another promising direction is to identify optimal CCA subsets that maximize diversity while maintaining high classification accuracy. Finally, enhancing the current two-stage hierarchy or replacing DTW-1NN with lightweight learning models may further improve performance.

ACKNOWLEDGMENT

The authors acknowledge the financial support by the German Research Foundation (DFG) project, Schwerpunktprogramm SPP 2378, ReNO-2, 2025-2029.

REFERENCES

- [1] I. Kunze, J. R  th, and O. Hohlfeld, "Congestion control in the wild - investigating content provider fairness," *IEEE Transactions on Network and Service Management*, vol. 17, pp. 1224–1238, Jun. 2020.
- [2] J. R  th, I. Kunze, and O. Hohlfeld, "An empirical view on content provider fairness," in *Network Traffic Measurement and Analysis Conference*, 2019.
- [3] R. Drucker, G. Baraskar, A. Balasubramanian, and A. Gandhi, "Bbr vs. bbrv2: A performance evaluation," in *2024 16th International Conference on COMMunication Systems & NETWORKS (COMSNETS)*. IEEE, 2024, pp. 379–387.
- [4] A. Mishra, W. H. Tiu, and B. Leong, "Are we heading towards a bbr-dominant internet?" in *Proceedings of the 22nd ACM Internet Measurement Conference*, 2022, pp. 538–550.
- [5] A. Mishra, X. Sun, A. Jain, S. Pande, R. Joshi, and B. Leong, "The great internet tcp congestion control census," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 3, no. 3, pp. 1–24, 2019.
- [6] P. Yang, J. Shao, W. Luo, L. Xu, J. Deogun, and Y. Lu, "Tcp congestion avoidance algorithm identification," *IEEE/ACM Transactions On Networking*, vol. 22, no. 4, pp. 1311–1324, 2013.
- [7] J. Oshio, S. Ata, and I. Oka, "Real-time identification of different tcp versions," in *Managing Next Generation Networks and Services: 10th Asia-Pacific Network Operations and Management Symposium, APNOMS 2007, Sapporo, Japan, October 10-12, 2007. Proceedings 10*. Springer, 2007, pp. 215–224.
- [8] C. Sander, J. R  th, O. Hohlfeld, and K. Wehrle, "Deepcci: Deep learning-based passive congestion control identification," in *Proceedings of the 2019 workshop on network meets AI & ML*, 2019, pp. 37–43.
- [9] [Online]. Available: <https://github.com/NetSys/bess/wiki/BESS-Overview>
- [10] K. Sonoda, K. Ogura, and J. Katto, "Tcp differentiation using version identification and edca for low-delay multimedia streaming," in *2013 IEEE 10th Consumer Communications and Networking Conference (CCNC)*. IEEE, 2013, pp. 753–756.
- [11] S. Gong, U. Naseer, and T. A. Benson, "Inspector gadget: A framework for inferring tcp congestion control algorithms and protocol configurations," in *Network Traffic Measurement and Analysis Conference*, 2020.
- [12] R. Ware, A. A. Philip, N. Hungria, Y. Kothari, J. Sherry, and S. Seshan, "Ccanalyzer: An efficient and nearly-passive congestion control classifier," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 181–196.
- [13] A. Mishra, L. Rastogi, R. Joshi, and B. Leong, "Keeping an eye on congestion control in the wild with nebbby," in *Proceedings of the ACM SIGCOMM 2024 Conference*, 2024, pp. 136–150.
- [14] J. Oshio, S. Ata, and I. Oka, "Identification of different tcp versions based on cluster analysis," in *2009 Proceedings of 18th International Conference on Computer Communications and Networks*. IEEE, 2009, pp. 1–6.
- [15] T. Nogiwa and K. Hirata, "Identification of tcp congestion control algorithms with convolution neural networks," in *2018 International Conference on Information Networking (ICOIN)*. IEEE, 2018, pp. 663–665.
- [16] E. Kfoury, J. Crichigno, and E. Bou-Harb, "P4cci: P4-based online tcp congestion control algorithm identification for traffic separation," in *ICC 2023-IEEE International Conference on Communications*. IEEE, 2023, pp. 4007–4012.
- [17] D. Carmel and I. Keslassy, "Dragonfly: In-flight cca identification," *IEEE Transactions on Network and Service Management*, 2024.
- [18] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *Proceedings of the 3rd international conference on knowledge discovery and data mining*, 1994, pp. 359–370.
- [19] S. Liu, T. Ba  sar, and R. Srikant, "Tcp-illinois: A loss and delay-based congestion control algorithm for high-speed networks," in *Proceedings of the 1st international conference on Performance evaluation methodologies and tools*, 2006, pp. 55–es.
- [20] N. Cardwell, Y. Cheng, S. H. Yeganeh, and V. Jacobson, "Bbr congestion control," *IETF Draft draft-cardwell-icrgg-bbr-congestion-control-00*, 2017.
- [21] S. Patel, Y. Shukla, N. Kumar, T. Sharma, and K. Singh, "A comparative performance analysis of tcp congestion control algorithms: Newreno, westwood, veno, bic, and cubic," in *2020 6th International Conference on Signal Processing and Communication (ICSC)*. IEEE, 2020, pp. 23–28.
- [22] N. Adel, G. Khoriba, and R. Sadek, "An efficient hybrid approach of yeah with multipath routing (mp-yeah) for congestion control in high-speed networks," in *Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2019*. Springer, 2020, pp. 809–820.