# Using SEARCH to Improve TCP Slow Start on Wireless Local Area Networks

Maryam Ataei Kachooei†, Amber Cronin†, Jose Manuel Perez Jimenez†, Katy Stuparu†, Connor Tam†, Jae Chung*, Feng Li*, Benjamin Peters*, Mark Claypool†
†Worcester Polytechnic Institute, Worcester, MA, USA
*Viasat Inc., Marlboro, MA, USA
†{mataeikachooei, acronin, jjimenez1, kastuparu, cmtam, claypool}@wpi.edu
*{jaewon.chung, feng.li, benjamin.peters}@viasat.com

*Abstract*—**TCP slow start faces challenges when running over wireless networks due to varying capacities and round-trip times caused by fluctuating channel conditions and cross traffic from competing flows. The TCP Slow start Exit At Right CHoke point (SEARCH) algorithm has been shown to improve slow start over satellite networks, but has not yet been evaluated over more commonly used wireless local area networks (LANs). This paper presents a performance evaluation of SEARCH on a busy campus Wi-Fi LAN, considering a range of network conditions. Our data shows that SEARCH improves TCP's initial bandwidth-probing behavior in a campus Wi-Fi environment by avoiding premature slow start exits and reducing packet loss compared to traditional TCP and TCP slow start augmented by HyStart.**

*Index Terms*—**Wi-Fi LAN, Round-Trip Time, Congestion**

## I. Introduction

Despite its widespread use as a "last mile" connection to the Internet, Wi-Fi networks present unique challenges that can significantly impact the performance of Transmission Control Protocol (TCP) [1], the de facto Internet communication protocol. In congested environments such as crowded University campus networks, Wi-Fi challenges to TCP are particularly pronounced with multiple devices contending for limited bandwidth, causing congestion. Moreover, interference from neighboring Wi-Fi networks or devices operating on the same frequency as a Wi-Fi access point can degrade signal quality, leading to packet loss and increased latency. These disruptions not only hinder TCP's congestion control mechanisms but also impair overall data transmission efficiency. Additionally, the inherent characteristics of wireless channels, such as fluctuations in signal strength, quality, and propagation, further complicate TCP's ability to determine link capacity and settle transmission rates.

Improving the performance of TCP over Wi-Fi networks has the potential to benefit many Internet users. A key aspect affecting TCP's performance is its strategy for exiting slow start – a phase during which TCP rapidly increases its sending rate until it determines it has reached the available bandwidth. Exiting slow start too early may result in link underutilization, while exiting too late can lead to network congestion and packet loss. Exiting in a timely manner means reaching the available bandwidth but not overshooting enough to cause packet loss and congestion.

To meet this challenge, we developed the Slow start Exit At Right CHoke point (SEARCH) algorithm [1] as an enhancement to TCP [2], [3]. SEARCH identifies the choke point for exiting slow start by analyzing the relationship between delivered bytes and expected delivery bytes within the previous round-trip time (RTT). SEARCH employs smoothing techniques to accommodate variations in link latency, which helps the algorithm be more robust across diverse network conditions. SEARCH has been evaluated over GEO, LEO, and 4G LTE wireless networks.

In this study, we aim to evaluate the effectiveness of SEARCH in an ecologically valid Wi-Fi network setting. Specifically, we assess performance in an active campus Wi-Fi environment, comparing TCP with SEARCH to TCP without (i.e., traditional TCP) and TCP with HyStart [4] (i.e., default Linux TCP). We first broadly assess Wi-Fi performance across campus, ascertaining TCP throughput for Wi-Fi connections with different signal strengths. From this initial set, we select a location with good Wi-Fi performance, one with medium Wi-Fi performance, and one with bad Wi-Fi performance for in-depth performance evaluation.

Our experiments show considerable variation in signal strength and TCP download performance across campus, even while connectivity seems good for users. Moreover, TCP throughput varies considerably over time, even for high signal strength, well-connected areas. TCP SEARCH improves the performance of both TCP with and without HyStart, generally exiting slow start after the congestion point is reached but before inducing congestion and contributing to packet loss.

The rest of this paper is organized as follows: Section II provides a review of related work, Section III outlines the experimental setup and methodology used to evaluate the performance of SEARCH in our campus Wi-Fi environment, Section IV describes experiments and analysis evaluating performance, and Section V summarizes our conclusions and presents possible future work.

## II. Related Work

Commonly deployed TCP variants focus on enhancing TCP's slow start behavior. HyStart [4] tries to avoid overshooting link capacity by monitoring RTT increases and inter-packet

[1]https://search-ss.wpi.edu/

arrival patterns to detect emerging congestion. When either delay inflation or packet clustering is observed, HyStart exits slow start and switches to congestion avoidance, thereby short-circuiting using packet loss to exit slow start. While effective in many wired settings, HyStart may trigger early exits in noisy wireless environments, leading to underutilization [5].

HyStart++ [6], a refinement to HyStart and standardized in the IETF, attempts to reduce the false positives that can plague HyStart. HyStart++ adds a second phase to HyStart where, instead of immediately exiting to congestion avoidance, HyStart++ uses a more conservative rate of congestion window growth while double-checking that the RTT has increased as a measure of "at-capacity". Nevertheless, like its predecessor HyStart, HyStart++ may still struggle in many wireless environments, where rapid RTT fluctuations and cross-traffic make RTT an unreliable signal of congestion.

Other TCP variants aimed at improving performance in general.Kabir et al. [7] evaluate four TCP variants - Cubic, Vegas, YeAH, and Westwood Plus - under varying node densities using NS-3 in wireless networks. Cubic achieves the highest throughput and lowest packet loss, while Vegas maintains stable but limited performance. YeAH and Westwood Plus degrade more under congestion, highlighting the need for congestion control strategies tailored to wireless conditions.

Nguyen et al. [8] evaluate TCP BBR's fairness in a smart device Wi-Fi network compared to TCP Cubic. They find that TCP Cubic often outperforms TCP BBR due to differences in queue management and congestion window sizing. Their discussion highlights the importance of TCP algorithms in alternative scenarios like data uploading (instead of downloading), especially pertinent with the emergence of 5G.

Grazia et al. [9] demonstrate TCP BBR's inefficiency in Wi-Fi bandwidth utilization in IEEE 802.11n and IEEE 802.11ac networks. To overcome this, they propose BBRp, a refined TCP BBR that improves upon the throughput over wireless – BBRp has 4-6 times better throughput over IEEE 802.11n and IEEE 802.11ac, albeit with slightly increased latency compared to traditional TCP.

Aoyagi et al. [10] introduce a Mininet Wi-Fi emulator for evaluating 14 Linux-based TCP variants across IEEE 802.11 standards. Their platform supports RTT, throughput, and fairness analysis, and aligns well with real Wi-Fi results. BBR delivered the best performance under bufferbloat, while Reno showed the highest fairness. The study highlights emulation as a practical tool for analyzing TCP behavior in wireless settings.

Our work complements the above research by providing additional performance evaluation of TCP, but whereas prior works focus on the congestion avoidance phase of TCP or on TCP slow start in general, our work focuses on the slow start phase of TCP over Wi-Fi.

## III. Methodology

This section describes our experimental setup and methodology for evaluating the performance of SEARCH on a campus Wi-Fi network and comparing it with TCP with HyStart enabled (the default on Linux) and TCP with HyStart disabled (traditional TCP). Our goal is to assess the ability of SEARCH to provide a timely exit for TCP slow start – after the congestion point is reached but before inducing packet loss.

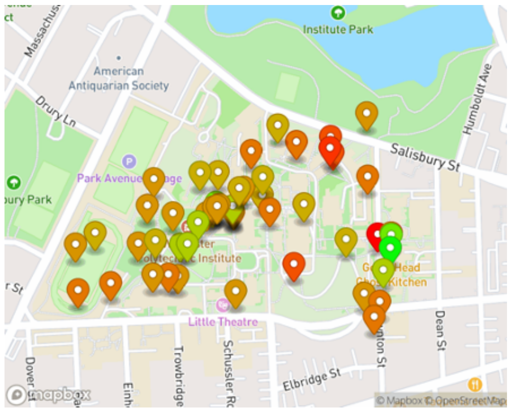

Fig. 1: Wi-Fi RSSI on the WPI campus.

### A. Wi-Fi Connection Strength

For Wi-Fi connections, the Received Signal Strength Indicator (RSSI) represents the strength of the received radio signal from the access point and is a key indicator of the quality of wireless connections. Higher RSSI values typically signify stronger signal reception and better network performance, while lower RSSI values may indicate weaker signal reception and potential connectivity issues.

We assess the wireless signal strength across various locations within our campus Wi-Fi environment. Figure 1 charts the distribution of RSSIs at different points, superimposed on our campus map. Each pin indicates a location where we did a TCP download, but first measured the RSSI as reported by the laptop Wi-Fi driver. Locations with a strong signal strength are shown with green coloring, while locations with weak signal strength are shown with red coloring. By visually depicting the RSSI at each location, the map allowed us to identify areas with strong, moderate, and weak signal reception.

To further characterize the network performance associated with different RSSI levels, we analyzed the relationship between RSSI and data transfer rates. Figure 2 shows the results. The x-axis is the RSSI recorded by the client, and the y-axis is the throughput at that location for a 3-second TCP download using the default Linux TCP settings. From the graph, there is a visual correlation between throughput and RSSI. However, at higher RSSI values (better quality), the throughput gain is not uniform. There is much more variation in the data points (lower correlation). This is likely because of additional factors that affect performance (e.g., connections with lower RSSI sharing the access point with connections with higher RSSI).

Based on the RSSI values and the map, we selected three locations for a detailed examination, shown with red points in Figure 2: one representing strong network conditions (high RSSI), one representing moderate network conditions (medium RSSI), and one representing poor network conditions

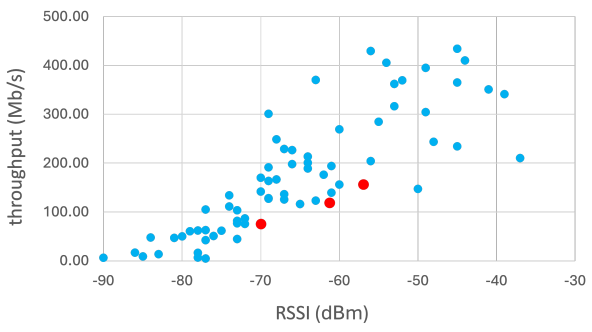

Fig. 2: RSSI versus throughput on the WPI campus.

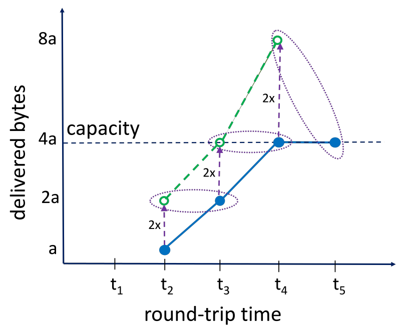

Fig. 3: TCP acknowledged delivery of data over time.

(low RSSI). These cases are used as focal points for evaluating the performance of TCP with SEARCH, comparing it with TCP with HyStart enabled and TCP with HyStart disabled.

## B. SEARCH Overview

Before discussing the experimental methodology, we present a concise overview of the SEARCH algorithm. SEARCH runs during TCP slow start and tries to exit to the congestion avoidance phase after reaching the congestion point (e.g., link capacity) but before inducing packet loss. In TCP slow start, each acknowledgment received incrementally increases the congestion window, effectively doubling the window size – and consequently the sent bytes – each RTT. Hence, under uncongested network conditions, the number of bytes acknowledged as delivered within a given RTT is approximately twice the number of bytes delivered in the preceding RTT [2].

This principle is depicted in Figure 3, where the number of delivered bytes (represented by the blue line) is compared with twice the number of delivered bytes one RTT earlier (indicated by the green line) along the y-axis against the RTTs on the x-axis. For instance, at $t_3$, $2a$ bytes are delivered, indicating that the delivered data doubles compared to the previous RTT ($t_2$). Also, at $t_4$, the number of delivered bytes remains consistent with twice the amount delivered during the preceding RTT ($t_3$), indicating that the network capacity has not been reached. This pattern persists at $t_4$, further indicating uncongested network conditions. However, at $t_5$, the number of delivered bytes remains $4a$, signaling a departure from the previously observed doubling trend – despite attempting to double the congestion window and transmit doubled data, the receiver did not receive these additional bytes, resulting in a stagnant delivery rate. This difference between the delivered bytes and the expected delivered bytes (i.e., delivered bytes not doubling) is the signal to SEARCH that the congestion point has been reached and slow start can be exited before encountering packet loss [3].

To minimize false signals caused by RTT variance unrelated to congestion, SEARCH smooths the delivered-byte measurements across multiple RTTs. This makes SEARCH more resilient to the variability of wireless links, unlike HyStart-type approaches that infer congestion directly from RTT inflation.

SEARCH is open source and has been implemented[2] in Linux, QUIC, and FreeBSD, allowing it to be evaluated and adopted in both research and production transport stacks.

## C. Testbed

Our experimental evaluations were conducted on the WPI campus Wi-Fi environment. WPI has wireless coverage for virtually every building on the 95-acre urban campus. The campus Wireless uses is a controller drop-off that sends all user traffic back to one of 6 campus controllers over an encrypted tunnel. The Access Points (APs) do not connect with each other as would happen in a mesh design – each AP is wired back to the network for its connections. Users associated via guest or open methods of authentication have a hard bitrate limit cap to prevent possible degradation of critical traffic. For bitrate management, Quality of Service (QoS) typically relies on traffic type classification, prioritizing voice, video, or control traffic. For security, there is Distributed Denial of Service (DDoS) protection involved that limits large amounts of packets per second for various protocols over wireless, blocking the session if a flow appears to be sending maliciously. From the wireless signals, generally, there is a limit to the amount of competing wireless noise that causes signal degradation, but because of the urban environment of the campus, sometimes there are not many options due to conflicting signals from other competing wireless initiatives.

For our performance measurements, we used a fixed, dedicated on-campus server and mobile laptop client for bulk TCP downloads, with the different TCP configurations controlled at the server. The server is a PC with an Intel i5-8500 CPU @ 3GHz and 8 GB RAM. It runs Mint-20.3 with Linux kernel version 5.10.79. The client is a MacBook Pro with an M2 Pro chip @ 3.4GHz with a total of 10 cores and 16 GB of memory. The laptop has an OS loader version of 10151.81.1.

## D. Download

During the experiments, we utilized the SEARCH algorithm compiled in the server kernel, comparing its performance with both HyStart [4] enabled (i.e., the Linux default TCP configuration) and HyStart disabled (i.e., the traditional TCP

[2]https://search-ss.wpi.edu/

configuration). To conduct these comparisons, multiple downloads were performed using the iperf3 tool.

The client initiated download sessions where the server was instructed to sequentially use one of: 1) the SEARCH algorithm, 2) HyStart off, or 3) HyStart. Each download session lasted 3 seconds, followed by a 5-second pause. After completing one download with each of the three configurations, the client paused 10 seconds to allow for system stabilization before repeating the process. The process is repeated 100 times, for a total of 300 downloads for a session. Each session was repeated at three selected RSSI locations: high, medium, and low. All datasets were collected on weekdays during the academic year while classes were in session and campus activity was at normal levels. The high RSSI data was captured from 5-6 pm, the medium RSSI data from 10-11 am, and the low RSSI data from 12-1 pm.

## IV. Evaluation

Table I provides a summary of the results derived from the 900 downloads conducted on the WPI campus Wi-Fi network. These downloads compare the performance of SEARCH with HyStart enabled and disabled across varying levels of RSSI strength. The table includes throughput and median throughput (in Mbits per second). The exit bitrate is the TCP congestion window (cwnd) divided by the RTT when slow start exits (in Mbits per second). The packet loss count over the entire 3-second download is reported in the last column. The values are the means of the 100 runs for that configuration, with standard deviations in parentheses.

The rows are in three blocks, one for each RSSI condition: high, medium, and low. Within each block, there is a row for the experimental condition: SEARCH, HyStart on, and HyStart off.

From the table, the low Wi-Fi RSSI connections have about half the throughput as the meium RSSI Wi-Fi connections, with slightly higher standard deviations. The strong RSSI Wi-Fi connections are about 50% higher than the medium RSSI Wi-Fi connections, but with a much higher standard deviation.

Across the three configurations (SEARCH, HyStart on, and HyStart off), throughputs (both mean and median) are similar, especially considering the standard deviations. The exit bitrates (computed via cwnd/RTT) are all above the medians.

For packet loss, HyStart off has the highest, and with all configurations having a large standard deviation.

### *A. TCP Link Capacity over Wi-Fi*

As mentioned earlier, SEARCH aims to exit slow start after the congestion window has sufficiently grown to reach link capacity without inducing packet loss. To assess whether or not SEARCH (and HyStart enabled/disabled) were able to do this, we examined the throughput over time for the TCP flow past the slow start condition.

Figure 4(a) shows an example, depicting throughput over time (in 15 ms intervals) computed from a Wireshark pcap file on the client. The throughput is generally high (around 200 Mb/s), but packet loss does not occur until 0.46 seconds, and it is not visually clear what the capacity is and where TCP should have exited slow start. So, we use the median value over the first second as a measure of the 'at capacity' point.

Figure 4(b) shows the cumulative distribution of the throughputs from Figure 4(a) for the same run, with the time the median throughput was reached indicated by the light blue dashed line. The slow start exit bitrate, calculated by dividing the congestion window by the RTT at the slow start exit time, can be compared to the 'at capacity' point for each download to ascertain whether or not slow start reached the capacity point before exiting. The dashed orange line shows the exit time (bitrate in the key) for this connection, which surpasses the median throughput, indicating that the capacity has been reached. Figures 2(c) and 2(d) illustrate the cwnd and RTT, respectively, for the first 100 milliseconds of the same download. The dashed green lines indicate the slow start exit time, the dashed blue lines show the 'at capacity' time based on the median throughput. Notably, the exit time occurs after the 'at capacity' point, indicating slow start reached capacity. As this is before packet loss (at 0.46 seconds), this is a 'timely' exit.

Table II compares the exit status for each TCP configuration (SEARCH, HyStart On, HyStart Off) at each location (RSSI high, medium, low). The percentage that TCP exited in each category is given for 'Early exit' – exiting slow start before reaching capacity – 'Timely exit' – exiting slow start after reaching capacity but before experiencing packet loss – and 'Late exit' – exiting slow start only upon encountering packet loss. The 'Total' category at the bottom is the average across all Wi-Fi configurations.

For all Wi-Fi conditions, HyStart off (traditional TCP) nearly always exited late, upon encountering packet loss. HyStart on (default Linux TCP), mostly had timely exits but still had a modest (about 28%) percentage of late exits and some (5%) early exits. SEARCH demonstrated the lowest percentage of early exits (about 2%) and the highest percentage of timely exits (76%), with a modest occurrence of late exits.

TABLE I: SEARCH, HyStart enabled, HyStart disabled over Wi-Fi (throughputs measured from the server-side tcpdump)

| Signal Strength | Slow start method | Throughput (Mb/s) | Median Thrpt. (Mb/s) | Exit bitrate (Mb/s) | Loss (pkt) |
|---|---|---|---|---|---|
| High | SEARCH | 152.9 (96.0) | 156.3 (101.7) | 272.0 (141.1) | 65 (159.6) |
| | HyStart on | 146.4 (99.6) | 149.0 (104.8) | 355.4 (296.7) | 37 (57.2) |
| | HyStart off | 136.20 (97.5) | 136.7 (103.4) | 306.9 (176.2) | 470 (495.2) |
| Medium | SEARCH | 113.20 (52.8) | 113.4 (57.0) | 307.9 (212.1) | 63 (137.8) |
| | HyStart on | 115.3 (57.8) | 116.5 (64.3) | 297.69 (152.6) | 84 (127.2) |
| | HyStart off | 114.91 (58.8) | 116.26 (65.5) | 286.5 (142.4) | 416 (378.9) |
| Low | SEARCH | 67.2 (39.8) | 67.9 (42.7) | 161.56 (73.6) | 41 (72.2) |
| | HyStart on | 66.8 (45.5) | 67.3 (49.1) | 134.7 (88.7) | 29 (47.6) |
| | HyStart off | 71.3 (39.1) | 72.2 (42.9) | 136.6 (96.2) | 67 (150.0) |

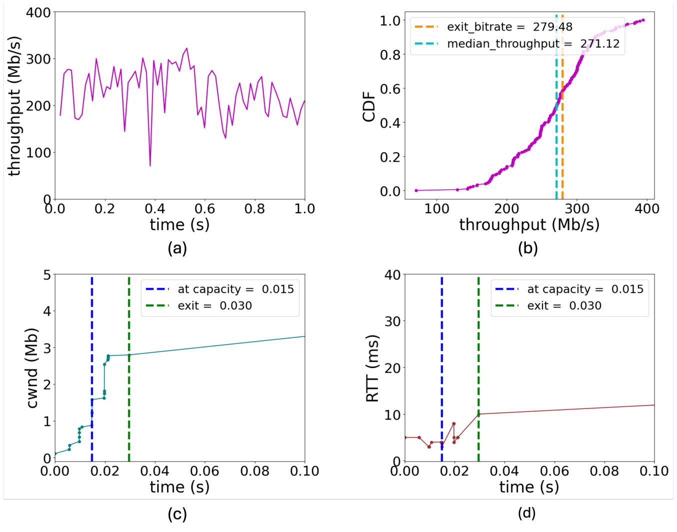

Fig. 4: TCP download performance: (a) throughput over time, (b) throughput distribution, (c) cwnd over time, and (d) RTT over time. The 'at capacity' point is the median of the throughputs over the first second, and the 'exit' is the slow start exit time.

TABLE II: Exit analysis for SEARCH, HyStart enabled, HyStart disabled over Wi-Fi.

| Signal Strength | Slow start configuration | Early exit (%) | Timely exit (%) | Late exit (%) |
|---|---|---|---|---|
| High | SEARCH | 2 | 81 | 17 |
| | HyStart on | 6 | 68 | 26 |
| | HyStart off | 2 | 0 | 98 |
| Medium | SEARCH | 0 | 73 | 27 |
| | HyStart on | 2 | 67 | 31 |
| | HyStart off | 2 | 0 | 98 |
| Low | SEARCH | 3 | 73 | 24 |
| | HyStart on | 6 | 68 | 26 |
| | HyStart off | 6 | 0 | 94 |
| Total | SEARCH | 1.6 | 75.7 | 22.7 |
| | HyStart on | 4.6 | 67.7 | 27.7 |
| | HyStart off | 3.3 | 0 | 96.7 |

## V. Conclusion

Despite their prevalence, Wi-Fi networks present unique challenges to TCP due to their susceptibility to congestion and interference, making it difficult for TCP slow start to determine the right exit point. Exiting slow start too early can mean underutilization as the congestion avoidance phase takes longer to reach the capacity point, while exiting slow start too late can cause unnecessary packet loss and take longer for the transmission rate to stabilize.

The Slow start Exit At Right CHokepoint (SEARCH) algorithm is designed to improve slow start by exiting after capacity is reached, but before packet loss occurs. Over satellite and 4G LTE links, SEARCH has been shown to reduce the number of early exits compared to TCP with HyStart (the default Linux setting) and greatly reduce the late exits compared to TCP without HyStart (traditional TCP) [3], but has not yet been evaluated over Wi-Fi.

This paper presents results characterizing the Wi-Fi signal strengths across a busy campus network, with a correlation between throughput and signal strength for dozens of on-campus locations. An in-depth performance evaluation for three locations with different Wi-Fi RSSI strengths – high, medium and low – shows that SEARCH works better than TCP with HyStart, exiting slow start in a timely fashion more often, and works far better than TCP without HyStart, which nearly always exits TCP late.

Since SEARCH requires only modest modifications to CUBIC slow-start in the Linux TCP stack, SEARCH could potentially be adopted in Wi-Fi router firmware or operating-system TCP implementations to improve start-up flow performance in environments where kernel updates are feasible.

In future work, SEARCH can be assessed in other wireless networks, including 5G, as well as incorporate other wireless factors such as indoor/outdoor and mobility. Other research efforts could consider explicitly assessing the effects of contending and cross traffic on SEARCH. Complementary future work could include performance comparisons between SEARCH and other TCP algorithms, such as BBR [11], HyStart++ [6]. Other future work can include extending experiments with uplink flows, as well as conducting deeper analysis of throughput and loss variability.

## References

[1] W. Eddy, "Transmission Control Protocol (TCP)," RFC 9293, Aug. 2022. [Online]. Available: https://www.rfc-editor.org/info/rfc9293

[2] M. A. Kachooei, J. Chung, F. Li, B. Peters, and M. Claypool, "SEARCH: Robust TCP Slow Start Performance over Satellite Networks," in *IEEE 48th Conference on Local Computer Networks (LCN)*. Daytona Beach, FL, USA: IEEE, 2023, pp. 1–4.

[3] M. A. Kachooei, J. Chung, F. Li, B. Peters, J. Chung, and M. Claypool, "Improving TCP Slow Start Performance in Wireless Networks with SEARCH," in *IEEE WoWMoM*, Perth, Australia, 2024.

[4] S. S. Ha and I. Rhee, "Taming the Elephants: A New Hybrid Slow-start for TCP," *Computer Networks*, vol. 52, no. 13, pp. 2577–2591, 2008.

[5] B. Peters, P. Zhao, J. W. Chung, and M. Claypool, "TCP HyStart Performance over a Satellite Network," in *Proceedings of the 0x15 NetDev Conference*, Virtual Conference, Jul. 2021.

[6] P. Balasubramanian, Y. Huang, and M. Olson, "HyStart++: modified slow start for TCP," *IETF Draft draft-balasubramanian-tcpm-hystartplusplus-03*, 2020.

[7] S. Kabir, A. C. Biswas, R. Shaikh, and A. K. Paul, "A Comprehensive Study on Different Variants of TCP Protocol in Wireless Network," in *5th International Conference on Image Processing and Capsule Networks (ICIPCN)*. Dhulikhel, Nepal: IEEE, 2024, pp. 748–753.

[8] K. Miyazawa, S. Yamaguchi, and A. Kobayashi, "Performance Evaluation of TCP BBR and Cubic TCP in Smart Devices Downloading on Wi-Fi," in *IEEE International Conference on Consumer Electronics-Taiwan (ICCE-Taiwan)*. Taoyuan, Taiwan: IEEE, 2020, pp. 1–2.

[9] C. A. Grazia, M. Klapez, and M. Casoni, "BPRP: Improving TCP BBR Performance over WLAN," *IEEE Access*, vol. 8, 2020.

[10] S. Aoyagi, Y. Horie, D. Thi Thu Hien, T. Duc Ngo, D.-D. Le, K. Nguyen, and H. Sekiya, "An Accurate Platform for Investigating TCP Performance in Wi-Fi Networks," *Future Internet*, vol. 15, no. 7, p. 246, 2023.

[11] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based Congestion Control," *Communications of the ACM*, vol. 60, no. 2, pp. 58–66, 2017.