Fast and Effective Bandwidth Estimation Using TCP Slow Start

Maryam Ataei Kachooei[†], Jae Chung[*], Feng Li[*], Benjamin Peters[*], Mark Claypool[†]
[†]Worcester Polytechnic Institute, Worcester, MA, USA
[*]Viasat Inc., Marlboro, MA, USA
[†]{mataeikachooei, claypool}@wpi.edu [*]{jaewon.chung, feng.li, benjamin.peters}@viasat.com

***Abstract*—Accurately estimating available bandwidth helps understand network performance and identify target bitrates for rate-based applications (e.g., video streaming). Traditional "speed test" bandwidth estimation tools often rely on prolonged data transfers, resulting in significant overhead in both time and data transferred. As an alternative, we evaluate using TCP slow start as a lightweight bandwidth probing tool, enhanced to detect the slow start exit point before inducing packet loss. We evaluate this TCP slow start 'plus' an enhancement approach (SS+) against three widely used speed test tools over various access links. Results show that the SS+ achieves accuracy comparable to traditional methods, while significantly reducing measurement time and data transmitted.**

***Index Terms*—available bandwidth, slow start, speed test**

I. Introduction

Accurate estimation of available bandwidth is important for optimizing the performance of rate-based applications such as video streaming, cloud gaming, and adaptive data transfer. These applications depend on timely knowledge of network capacity to make effective bitrate and congestion control decisions. Bandwidth measurement is also helpful for network diagnostics, capacity planning, and service quality monitoring [1], [2].

Several tools are widely used for bandwidth estimation. Ookla Speedtest [3] is a popular commercial solution, performing multi-threaded TCP downloads over several seconds and reporting average throughput. M-Lab's NDT7 (NDT7) [4] is an open-source platform that runs single-stream TCP tests and emphasizes transparency and research use. LibreSpeed [5] is a self-hosted alternative that supports browser-based testing using customizable HTTP or WebSocket-based downloads. While effective, these tools all rely on sustained data transfers to saturate the link, resulting in considerable measurement time and data usage, often tens of megabytes per test [6].

We propose a lightweight alternative that leverages TCP slow start as a natural bandwidth probing mechanism. During TCP slow start, the congestion window (cwnd) increases exponentially with each round-trip time (RTT) until the path capacity is exceeded. When this overshoot triggers packet loss (or ECN marking), TCP reduces its congestion window toward the available bandwidth. However, determining the capacity upon overshoot (i.e., packet loss) is challenging – current heuristics scale the congestion window to a hard-coded fraction of its current value. Fortunately, heuristics have been proposed for traditional TCP slow start that aim to avoid overshoot, exiting after reaching capacity but before packet loss occurs. One such technique is the Slow start Exit at CHoke point (SEARCH) algorithm [7] that detects when the delivery rate stops growing and then exits slow start. Unlike traditional tools that require prolonged transfers to infer capacity, traditional slow start enhanced with an algorithm like SEARCH (SS+) can be used to report the delivery rate as an estimate of bandwidth when the network chokepoint has first been reached, dramatically reducing test duration and data volume over traditional speed tests.

We evaluate SS+ against Ookla Speedtest, M-Lab NDT7, and LibreSpeed across three access networks: 4G LTE (Verizon), residential cable (Spectrum), and a geostationary (GEO) satellite network (Viasat). Our tests use private servers for NDT7 and LibreSpeed to ensure fair, controlled comparisons with SS+. Results show that SS+ achieves similar bandwidth estimates compared to the other tools, while reducing test time and bytes sent by up to an order of magnitude, making SS+ a practical alternative for fast, low-overhead bandwidth estimation.

The remainder of this paper is organized as follows: Section II reviews related work on bandwidth estimation tools and techniques; Section III describes the SS+ approach and outlines our experimental methodology; Section IV analyzes the results comparing SS+ against Ookla Speedtest, M-Lab NDT7, and LibreSpeed over different networks; Section V discusses the implications of our findings; and Section VI summarizes our conclusions and possible future work.

II. Related Work

Accurate bandwidth estimation has been researched as a tool for improving network performance and application Quality of Experience (QoE). Research approaches typically rely on either active probing or passive monitoring.

Yang et al. [6] introduced FastBTS, a lightweight bandwidth test that replaces flooding-based probing with a fuzzy rejection sampling framework. By leveraging Elastic Bandwidth Probing (inspired by BBR [8]), FastBTS reduced test duration and data usage while maintaining accuracy for 5G and metered networks.

Building on this, Li et al. [9] proposed FastUpBTS for efficient uplink testing. FastUpBTS combined memorization-

reinforced sampling with informed multi-homing to reduce overhead and stop early when throughput stabilizes. FastUpBTS reached about 90% accuracy, ran faster, and used less data than Speedtest.net, making it better-suited for mobile and high-speed access.

Song et al. [10] presented CUP, a lightweight bandwidth estimator for cellular and Wi-Fi networks that minimized probe traffic by using intelligently timed TCP bursts. CUP leveraged low-level wireless properties – like frame aggregation and transport block size fluctuation – alongside a Bayesian change-point detection algorithm to estimate available bandwidth with high accuracy and low overhead. Designed for efficiency, CUP enabled longitudinal and large-scale measurements even in mobile or resource-constrained settings.

Ciaccia et al. [11] proposed SABES, a passive bandwidth estimator that inferred available bandwidth by analyzing TCP acknowledgment (ACK) inter-arrival times during slow start. By statistically modeling inter-ACK gaps and applying a neural network filter, SABES estimated bottleneck capacity and available bandwidth without injecting active probes, making it suitable for lightweight, real-time monitoring.

MacMillan et al. [12] conducted a large-scale, side-by-side evaluation of Ookla Speedtest and M-Lab NDT7, highlighting how factors like server selection and connection count impacted measurement accuracy. They found that Speedtest's multi-stream design tended to report higher throughput due to aggressive parallelism, whereas NDT7, using a single TCP stream, offered lower overhead but often underestimated capacity, especially on fast links.

Mangla et al. [13] examined how bandwidth estimation results from platforms like Speedtest, NDT7, and Fast.com can be misinterpreted due to methodological differences – e.g., number of flows, server proximity, and test duration. Their large-scale deployment across urban neighborhoods showed that these tests often failed to account for infrastructure-driven disparities. They argued for more context-aware and equity-focused interpretations, especially when using test results for policy or digital inclusion decisions.

Despite advances in bandwidth estimation, existing tools often involve trade-offs between accuracy, efficiency, and practical deployment. Active methods typically require long tests or multiple flows, while passive tools may depend on kernel access or specific assumptions. Our work takes a different direction by leveraging the inherent probing behavior of TCP slow start, where the growth of the congestion window naturally reveals the bottleneck capacity. To improve the precision of slow start, we advocate enhancing slow start to detect capacity before loss – we employ the previously-developed SEARCH algorithm in this paper – yielding an SS+ approach that combines the practicality of slow start with a refined exit condition.

III. Methodology

This section describes our approach for evaluating TCP slow start + SEARCH (SS+) as a lightweight bandwidth probing method, comparing performance to state-of-the-art speed test tools.

A. TCP Slow Start for Bandwidth Probing

TCP slow start provides a natural mechanism for bandwidth probing. During slow start, the sending rate approximately doubles each RTT until the path's bottleneck capacity is reached. At that point, the sending rate continues to grow, but the delivery rate no longer does as the bottleneck bandwidth limits any increase – additional packets accumulate in the bottleneck queue, leading to queuing delay without further throughput gains. As the sending rate continues to grow, the bottleneck queue overflows, and packets are lost. Upon detecting such loss, TCP knows it has surpassed the bottleneck bandwidth, but determining by how much, exactly, is a challenge. Current TCP implementations reduce the sending rate by a fixed fraction as a coarse estimate of the bottleneck bandwidth.

To improve the accuracy of TCP slow start bottleneck estimates, we incorporate the previously developed SEARCH algorithm [7]. SEARCH augments slow start by adding an explicit exit condition: it monitors the delivery rate and identifies when the increase slows, indicating that the bottleneck has been reached. To do this, SEARCH defines a delivery-measurement window equal to 3.5 × the initial RTT, divided into 10 equal-duration bins. For each ACK, the delivered bytes are accumulated in the current bin until the bin completes. Once 10 bins of data are available, SEARCH computes the delivered-bytes total for the current window and the window one RTT earlier. When the normalized difference between these two windows exceeds a threshold (0.35), this indicates that the delivered throughput is no longer increasing. SS+ then terminates its probing phase and reports the delivered rate. The SEARCH algorithm is open source and has been implemented in Linux, QUIC, and FreeBSD.[1]

Using TCP slow start with this SEARCH algorithm refinement (SS+) allows bandwidth estimates to be provided near the point where capacity is reached, reducing the chance of underestimation from an early exit or wasted data from overshooting into congestion.

B. Metrics and Measurement Approach

To evaluate the performance of SS+ compared to other bandwidth measurement tools, we recorded the following three metrics during each test:

Link bandwidth (B): Estimated download rate at the point when the bandwidth measurement completes. For SS+, this is defined as the delivery rate at the moment slow start exits, calculated as the number of bytes acknowledged as delivered over the most recent RTT:

$$B = \frac{D - D'}{\mathrm{R}}$$

where D is the cumulative delivered bytes when slow start exits, D′ is the cumulative delivered bytes one RTT prior, and

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

R is the RTT. For the other tools, we use the download speed they report at the end of their testing periods.

Bytes sent: Total data transferred during the download test, measured at the client. For SS+, this includes the data sent until the end of slow start; for other tools, this includes the full data volume transferred until the reported link bandwidth is computed.

Download duration: Time between the start of the download and the bandwidth estimate is finalized. For SS+, the finalized time is when slow start exits; for Speedtest, NDT7, and LibreSpeed, this is when the link bandwidth is reported.

These metrics allow us to assess the bandwidth estimates as well as the measurement overhead in terms of test duration and data transmitted.

C. Testbed and Tool Comparison

To ensure a fair and controlled comparison where possible, we deployed a private server for both M-Lab NDT7 and LibreSpeed on a dedicated desktop Linux machine. The server is equipped with an Intel i5-8500 CPU @ 3 GHz and 8 GB of RAM running Linux Mint 20.3 with kernel version 6.13.10. The server connects to our university LAN via gigabit Ethernet, with the campus network uplinked to the Internet through multiple 10 Gb/s connections, each throttled to 1 Gb/s. This server also hosted our custom iperf3 download-based implementation of SS+. Hosting all tools on the same machine eliminated variability due to server location, routing paths, and backend load, ensuring consistent conditions across tests.

Unfortunately, Ookla Speedtest does not support private server deployment without participating in their commercial hosting program. Instead, Speedtest dynamically selects the "closest" public server based on measured latency at runtime. As a result, Speedtest measurements may be subject to external variability across runs, including geographic routing shifts, different hosting providers, or varying server load. While we mitigated these concerns by conducting all measurements back-to-back, some server-induced variability remains inherent to Ookla's architecture. Despite these limitations, we include Ookla Speedtest in our evaluation because it is one of the most widely used bandwidth measurement platforms, with results familiar to both end-users and network operators. Also, Ookla's Speedtest inclusion in our experiments is a feature – it allows us to compare measurements obtained over the same controlled path (by SS+, NDT7, and LibreSpeed) with those obtained over potentially different paths selected dynamically (by Ookla), providing insight into how server-based deployment choices affect reported performance for the client link. These choices – private versus public infrastructures – mirror realistic conditions for widely deployed measurement tools.

While Speedtest, LibreSpeed, and NDT7 all compute and report upload and download bandwidth, as well as "ping" tests that measure network latency, only the download (server to the client) results are used and reported in this paper.

We conducted bandwidth measurements over three types of access networks to capture a range of representative last-mile environments. These included a 4G LTE "hotspot" using an iPhone 15 Pro on the Verizon network, a residential broadband connection (Cable) via Spectrum cable, and a GEO satellite link via Viasat. All tests were performed from the same client device to ensure consistency across runs. The client device is a mobile laptop equipped with an Intel Core Ultra 7 155U CPU and 15 GB of RAM running Ubuntu 24.04.2 LTS with Linux kernel version 6.11.0.

Each tool was executed 50 times per access network within a one-hour window. To minimize temporal bias, tests were interleaved across tools and conducted between 11:00 a.m. and 7:00 p.m. to represent typical "busy" hours. We restricted all measurements to download-only tests, omitting any upload component provided by the tools. For each run, we recorded both server logs and packet capture (pcap) files at the client and server to enable detailed analysis of traffic behavior and delivery rate dynamics.

To generate download traffic for SS+ measurements, we used iperf3 to perform controlled downloads from the private server. The duration of each iperf3 download was set based on network conditions: 10 seconds for the cable and 4G LTE networks, and 35 seconds for the GEO satellite network. These durations ensured sufficient traffic to observe slow start behavior and gather accurate throughput traces for baseline comparison. Note that TCP segmentation offload (TSO) and generic receive offload (GRO) were enabled by default during all tests.

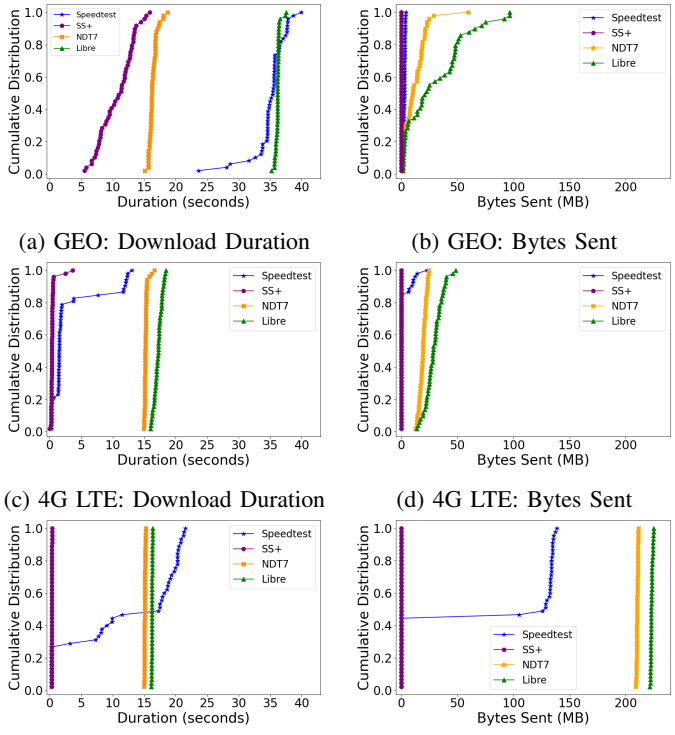
IV. Results

This section analyzes the results from the experiments comparing SS+ to Speedtest, NDT7, and LibreSpeed in terms of overhead (Section IV-A), and bandwidth estimation (Section IV-B), with a summary provided of the tradeoffs (Section IV-C).

A. Download Duration and Bytes Sent

Figure 1 presents the cumulative distributions of download duration (left) and bytes sent (right) for the three networks tested – GEO satellite (Viasat), 4G LTE (Verizon), and residential cable broadband (Spectrum) – for all four bandwidth estimation tools: SS+, Speedtest, NDT7, and LibreSpeed. These plots illustrate the overhead of each tool in terms of both time and data transmitted required to complete a test. For each graph, the x-axis is the metric (seconds for duration and MBytes for bytes sent), and the y-axis is the cumulative distribution. The distributions are grouped by each tool type, taken from the values recorded across all runs for that network type. Better values (i.e., lower overhead) are on the left side of the graph, and worse values (i.e., higher overhead) are to the right.

Figures 1a and 1b focus on the GEO satellite network. Figure 1a shows that SS+ completes in 5-16 seconds, significantly faster than Speedtest and LibreSpeed, which often exceed 35



(a) GEO: Download Duration (b) GEO: Bytes Sent

(c) 4G LTE: Download Duration (d) 4G LTE: Bytes Sent

(e) Cable: Download Duration (f) Cable: Bytes Sent

Fig. 1: Distributions of download duration and bytes sent.

seconds. NDT7's durations fall between 15 and 18 seconds, shorter than Speedtest and LibreSpeed but still longer than SS+. Figure 1b shows that SS+ consistently transmits less than 11 MB of data. Speedtest also has relatively low transmission overhead, with most runs sending under 12 MB. In contrast, NDT7 sends between 10-50 MB, while LibreSpeed incurs the highest overhead, frequently exceeding 50 MB and sometimes approaching 100 MB.

Figures 1c and 1d present results for the 4G LTE network. Figure 1c shows SS+ consistently completes in under 4 seconds, with most runs between 1-2 seconds, making it the fastest tool by a wide margin. Speedtest shows a broader range, with over 90% of its tests finishing within 10 seconds, but some extending to 12-14 seconds. Both NDT7 and LibreSpeed have mostly fixed durations, clustering tightly around 15–17 seconds. Figure 1d shows that SS+ transfers the least data, with all runs under 10 MB. Speedtest usually transfers slightly more data than SS+, and about 15% transfer significantly more, up to about 20 MB. Again, NDT7 and LibreSpeed show the highest overhead, typically sending 12–25 MB for NDT7 and 13-48 MB for LibreSpeed.

Figures 1e and 1f show results for the cable network. Figure 1e shows SS+ completing all runs in under 1 second, with most finishing in about 0.5 seconds. Speedtest, in contrast, exhibits significant variability, with about 25% of tests completing in about a second, but 70% of the tests taking from 7 to 22 seconds. NDT7 and LibreSpeed again cluster around a relatively fixed duration of 16 seconds. Figure 1f shows that SS+ consistently sends less than 5 MB of data. Speedtest shows a wide spread, with about 45% of the tests sending about 5 MB of data, but about half sending about 135 MB of data. Both NDT7 and LibreSpeed send slightly over 200 MB per test.

Table I summarizes the results across all networks for both bytes sent and test duration. The numerical results align with the trends illustrated in the per-network figures. SS+ generally completes with the lowest data overhead, with mean usage in the range of 1-10 MB, compared to tens or hundreds of MB for the other tools. The table also shows that SS+ tends to finish more quickly, requiring under a second on cable and LTE, and around 10 seconds on GEO satellite links.

In summary, SS+ generally sends less data and arrives at a bandwidth estimate faster than Speedtest, NDT7, and LibreSpeed. Speedtest shows the most variability in overhead, but some of that variability is likely due to choosing different servers across the set of runs. SS+, NDT7, and LibreSpeed are generally consistent in overhead for cable and 4G LTE networks, but vary more for the high-bandwidth, high-delay GEO network.

B. Bandwidth Estimation Accuracy

To assess estimation accuracy, we compute the normalized absolute difference between each tool's reported bandwidth and the median throughput recorded by the iperf3 download. We calculate the normalized absolute difference (N) as:

$$\mathrm{N} = \left| \frac{T - B}{T} \right|$$

where T is the median throughput observed during the iperf3 download test (computed from server-side packet captures), and B is the estimated bandwidth from the tool.

To compute the reference throughput, we first derive throughput over time using a sliding time window, and then take the median throughput across the entire session. We tested several different window sizes across all three access networks and observed that any window size within a magnitude of the base RTT resulted in a similar median being computed. So, for each access network, we compute the average RTT observed during the slow start phase over all 50 test cases and use that as the window size for all runs for that network. The resulting average RTTs – and thus the window sizes used – were 1.46 s for GEO (Viasat), 0.08 s for 4G LTE (Verizon), and 0.05 s for cable (Spectrum).

Figure 2 shows the cumulative distribution of normalized absolute differences for all tools across the three networks. Again, better values (i.e., lower error) are on the left side of the graph, and worse values (i.e., higher error) are to the right. Note, the x-axis is clipped at 2 to focus on the primary data range, although NDT7 and LibreSpeed on GEO have data beyond this limit, up to as high as 7.

Figure 2a shows the results for the GEO link. SS+ and LibreSpeed achieve the best overall accuracy, with about 90% of estimates below a normalized 1.0, and most between 0.2

TABLE I: Summary of bytes sent and test duration.

		Bytes Sent (MB)					Duration (s)				
Network	**Tool**	Mean	Std	Min	Max	Median	Mean	Std	Min	Max	Median
GEO	SS+	9.32	1.73	0.56	10.82	9.26	10.69	2.66	5.48	15.88	11.33
	Speedtest	9.83	0.75	1.07	11.08	9.75	34.99	2.64	23.62	39.90	35.41
	NDT7	12.03	10.27	1.18	59.60	10.24	16.37	0.67	15.08	18.73	16.20
	Libre	30.20	26.98	1.44	96.60	22.31	36.17	0.36	35.21	37.56	36.18
4G LTE	SS+	1.11	1.44	0.003	9.60	0.92	0.44	0.55	0.0001	3.63	0.34
	Speedtest	1.77	4.58	0.002	22.82	0.004	3.12	4.01	0.27	13.01	1.48
	NDT7	19.01	3.03	12.50	24.45	19.12	15.22	0.28	14.90	16.62	15.16
	Libre	28.93	7.48	13.84	48.27	28.35	17.19	0.63	15.97	18.46	17.24
Cable	SS+	2.86	0.21	2.39	3.54	2.81	0.33	0.02	0.29	0.38	0.32
	Speedtest	73.35	65.76	0.02	138.49	128.47	12.12	8.45	0.21	21.54	17.46
	NDT7	210.21	0.58	208.93	211.29	210.10	15.07	0.07	14.98	15.25	15.06
	Libre	223.05	0.86	221.52	224.91	222.82	16.23	0.05	16.08	16.34	16.22

and 0.8. Speedtest has about 60% of its estimates at about 1.0, with about 40% of the values uniformly distributed between 0 and 1.0. NDT7 shows performance similar to Speedtest for about 80% of the distribution, but has 20% of its estimates that are well over 1.0.

Figure 2b presents results for the 4G LTE network. LibreSpeed performs best, with over 90% of its estimates below 0.5 and nearly all below 1.0. SS+ and NDT7 perform comparably, with most estimates between 0.1 and 0.8, although the NDT7 distribution tail is higher, with about 5% at or above 1.5. Speedtest performs the worst, with 80% of its estimates between 0.8 and 1.0.

Figure 2c presents the results for the residential cable network. All the approaches demonstrate exceptional accuracy, well below 1.0, with only Speedtest having a notably higher distribution, particularly in the tail, with 10% of values between 0.15 and 0.5.

Table II provides a statistical summary of normalized absolute difference values across tools and networks. On the GEO satellite link, SS+ has the lowest mean error (0.54), followed by Libre (0.73) and Speedtest (0.79), while NDT7 shows the highest mean error (1.20). The NDT7 and Libre show wide variability based on the std and min/max values. For 4G LTE, Libre achieves the best accuracy with a mean error of 0.26, while SS+ (0.38) and NDT7 (0.42) are close behind; Speedtest again performs worst with a mean error of 0.87 despite relatively low variability. On the cable network, all tools perform well, with mean errors below 0.1 for SS+ (0.04), NDT7 (0.03), and Libre (0.02). Speedtest is somewhat higher (0.10) but still low compared to mobile and satellite networks.

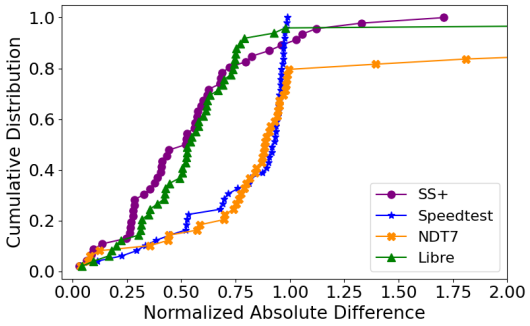
In summary, SS+, Speedtest, NDT7, and LibreSpeed generally produce similar – and low – error distributions for cable networks, whereas only SS+, NDT7, and LibreSpeed are accurate for 4G LTE networks, and only SS+ and LibreSpeed are accurate for GEO networks. All tools are considerably more accurate for the cable networks than either the GEO or 4G LTE networks.

TABLE II: Summary of normalized absolute difference between median throughput and estimated bandwidth.

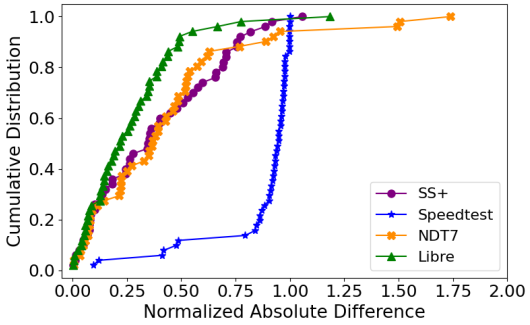
Networks	Tools	Mean	Std	Min	Max	Median
GEO	SS+	0.54	0.34	0.03	1.71	0.52
	Speedtest	0.79	0.26	0.03	0.99	0.93
	NDT7	1.20	1.04	0.04	5.36	0.88
	Libre	0.73	1.06	0.04	7.02	0.54
4G LTE	SS+	0.38	0.29	0.004	1.06	0.35
	Speedtest	0.87	0.21	0.09	1.00	0.95
	NDT7	0.42	0.38	0.006	1.74	0.37
	Libre	0.26	0.22	0.002	1.19	0.22
Cable	SS+	0.04	0.03	0.0004	0.11	0.03
	Speedtest	0.10	0.08	0.03	0.40	0.08
	NDT7	0.03	0.004	0.02	0.04	0.03
	Libre	0.02	0.004	0.006	0.03	0.02

C. *Trade-Off Between Accuracy and Overhead*

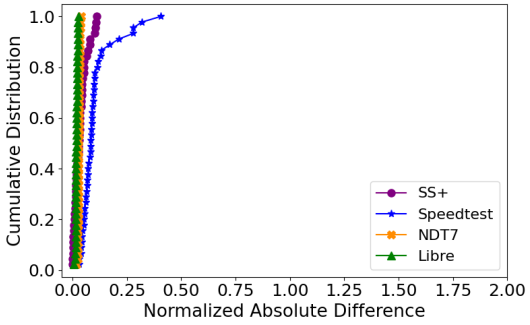
Figure 3 presents the trade-off between estimation accuracy (computed as error, the normalized difference) and the overhead (test duration and data sent) for all four tools and the three access networks. Each point represents the average error (i.e., the average normalized absolute difference) for a specific network-tool pair, plotted against either the average test duration (Figure 3a) or the average bytes sent (Figure 3b). The bottom left corner is better – i.e., low error, low overhead – and the top right corner is worse – i.e., high error, high overhead. Colored ellipses visually group the data by tool, with the Speedtest ellipses omitted for readability (i.e,. the Speedtest ellipses would be super broad, covering much of the graph). SS+ (purple) consistently appears near the bottom left corner in both graphs, confirming that it achieves high estimation accuracy with low duration and data sent. By contrast, NDT7 (orange) and LibreSpeed (green) appear to the right of the origin, reflecting their generally higher test durations and



(a) GEO



(b) 4G LTE



(c) Cable

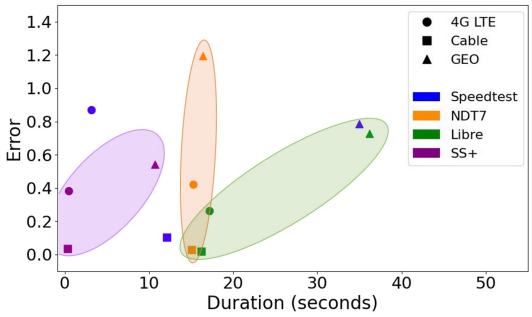
Fig. 2: Distributions of normalized absolute error. Lower values indicate more accurate bandwidth estimates.

data sent, with higher error for individual network types. For example, NDT7 on the GEO link shows both high error and long durations, whereas LibreSpeed has more accurate results on 4G LTE and Cable, but still takes significantly more time than SS+.

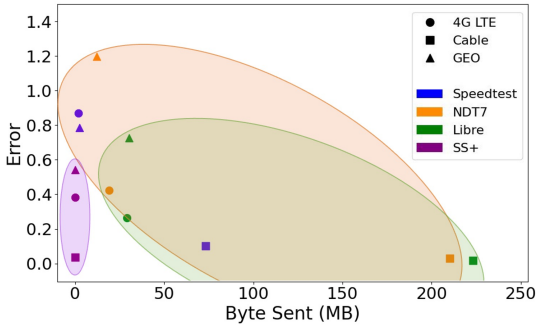
These graphs reinforce that SS+ generally offers a good trade-off in accuracy and overhead, with efficient and relatively accurate bandwidth estimation.

V. Discussion

Our evaluation shows that TCP slow start enhanced so as to exit upon reaching capacity can serve as an efficient bandwidth probing mechanism, providing a fast, low-overhead, and accurate alternative to conventional bandwidth estimation



(a) Error versus Test Duration



(b) Error versus Bytes Sent

Fig. 3: Trade-off between estimation error and overhead. Each colored point represents a tool-network pair. The ellipses were manually added to cluster tests to aid visual comparison.

tools. Compared to long-duration multi-flow measurement tools, SS+ requires only a single short probing phase to determine capacity on the path bottleneck.

A key strength of our study is the controlled server environment shared by SS+, NDT7, and LibreSpeed which eliminates variability caused by disparate backends and isolates the algorithmic behavior of each tool. In contrast, Ookla's Speedtest client dynamically selects a public server for each test based on latency, introducing variability in routing, congestion, and server-side hardware. This illustrates challenges with using endpoints other than the one of interest to determine capacity – in other words, the choice of endpoint matters for accurate bandwidth estimation.

For some applications, SS+ can be used for passive measurement of bandwidth estimation, reporting network performance as part of the application's normal behavior. For example, a streaming media application using TCP to transmit the first chunk of video could obtain a bandwidth estimate when slow start (using SEARCH) determines that network capacity had been reached.

Another extension to SS+ could be to repeat bandwidth estimates periodically over a longer interval – similar to how some traditional tools sustain flows and compute throughput based on data over a longer time period. Rather than relying on a single estimate during slow start, SS+ could be configured to re-enter slow start periodically and collect multiple lightweight

probes over a fixed time window. Aggregating these measurements (e.g., via averaging or smoothing) would yield more robust estimates, particularly in environments where network conditions fluctuate, such as mobile or Wi-Fi networks. This modification would allow SS+ to emulate the temporal coverage of long-duration tests without incurring their full overhead, making it well-suited for real-time applications like adaptive video streaming or network-aware task scheduling.

Another application for this work could be for QUIC. Since QUIC encrypts transport headers, passive bandwidth estimation techniques that rely on visible TCP sequence numbers are less effective. However, QUIC also employs the slow start phase for congestion control, which makes it possible to apply SS+ directly. In this context, slow start provides the probing mechanism, while SS+ offers a lightweight way to refine the exit condition and report bandwidth more accurately without exposing internal protocol details.

Another consideration is that SEARCH is not the only refinement that can be combined with slow start. Alternative mechanisms such as HyStart and HyStart++ [14] also enhance the slow start exit condition, and could be considered for SS+. Future work could extend the evaluation to include these variants, comparing how different exit strategies affect the balance between efficiency and accuracy in bandwidth estimation.

VI. Conclusion

This paper presents SS+, a lightweight TCP-based bandwidth estimation approach that leverages the built-in probing behavior of TCP slow start, coupled with a slow start enhancement to exit before packet loss. In contrast to traditional tools that rely on flooding-based or fixed-duration transfers, SS+ exits slow start at the chokepoint and reports the delivery rate, enabling rapid and efficient bandwidth estimation. Our implementation uses the previously developed SEARCH algorithm [7] as the '+' to refine the exit condition and improve accuracy.

Results measuring performance via controlled experiments across three access networks – GEO satellite, 4G LTE, and cable broadband – show SS+ achieves accuracy comparable to widely used tools Ookla's Speedtest, M-Lab's NDT7, and LibreSpeed's speed test, while taking significantly less time and sending less data. SS+ completes tests up to 6× faster and with 10-20× lower data usage, making it particularly well-suited for resource-constrained, mobile, or high-latency environments, while generally having error rates (inaccuracy in bandwidth estimates) within 10% of the other tools.

Future work includes testing across more access network types (e.g., WiFi, 5G), as well as a multi-phase bandwidth estimation tool that uses SS+ for repeated (configurable) probes over dynamically-determined time windows, reinforced by more extensive statistical analysis. Beyond TCP, SS+ can also apply to QUIC, which also employs a slow start phase but encrypts transport headers, making external measurement more challenging. Future work may also incorporate user-centric metrics, such as startup delay or adaptive bitrate behavior, to better reflect application-level performance. Additional visualizations, such as standard-deviation distributions for duration and overhead, could further characterize cross-run variability. Finally, while our study focused on SEARCH as a refinement, alternative exit strategies such as HyStart and HyStart++ could also be evaluated to compare trade-offs in accuracy and efficiency.

References

[1] C. Timmerer, H. Amirpour, F. Tashtarian, S. Afzal, A. Rizk, M. Zink, and H. Hellwagner, "HTTP Adaptive Streaming: A Review on Current Advances and Future Challenges," *ACM Transactions on Multimedia Computing, Communications and Applications*, 2025.

[2] S. Khairy, G. Mittag, V. Gopal, F. Y. Yan, Z. Niu, E. Ameri, S. Inglis, M. Golestaneh, and R. Cutler, "Bandwidth Estimation in Real Time Communications Challenge," in *Proceedings of the 15th ACM Multimedia Systems Conference*, Bari, Italy, 2024, pp. 339–345.

[3] Ookla, "Speedtest by Ookla," https://www.speedtest.net/, 2025, accessed July 2025.

[4] Measurement Lab, "ndt7 Protocol," https://www.measurementlab.net/tests/ndt/, 2025, accessed July 2025.

[5] LibreSpeed, "LibreSpeed Speedtest," https://github.com/librespeed/speedtest, 2025, accessed July 2025.

[6] X. Yang, X. Wang, Z. Li, Y. Liu, F. Qian, L. Gong, R. Miao, and T. Xu, "Fast and Light Bandwidth Testing for Internet Users," in *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2021, pp. 1011–1026.

[7] 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.

[8] 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, Jan. 2017.

[9] Z. Li, X. Li, X. Yang, X. Wang, F. Qian, and Y. Liu, "Fast Uplink Bandwidth Testing for Internet Users," *IEEE/ACM Transactions on Networking*, vol. 31, no. 4, pp. 1886–1901, 2023.

[10] L. Song, E. Halepovic, A. Mohammed, and A. Striegel, "CUP: Cellular Ultra-light Probe-based Available Bandwidth Estimation," in *IEEE/ACM 29th International Symposium on Quality of Service (IWQOS)*. Tokyo, Japan: IEEE, 2021, pp. 1–11.

[11] F. Ciaccia, I. Romero, O. Arcas-Abella, D. Montero, R. Serral-Gracià, and M. Nemirovsky, "SABES: Statistical Available Bandwidth Estimation from Passive TCP Measurements," in *IFIP Networking Conference (Networking)*. Paris, France: IEEE, 2020, pp. 743–748.

[12] K. MacMillan, T. Mangla, J. Saxon, N. P. Marwell, and N. Feamster, "A Comparative Analysis of Ookla Speedtest and Measurement Labs Network Diagnostic Test (ndt7)," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 7, no. 1, pp. 1–26, 2023.

[13] R. Sharma, T. Mangla, J. Saxon, M. Richardson, N. Feamster, and N. P. Marwell, "Benchmarks or Equity? A New Approach to Measuring Internet Performance," *A New Approach to Measuring Internet Performance*, August 3, 2022.

[14] Y. Balasubramanian, Y. Huang, and M. Olson, "HyStart++: Modified slow start for TCP," RFC 9406, May 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9406