TCP Maintenance and Minor Extensions (tcpm)
Internet-Draft
Intended status: Experimental
Expires: 19 November 2026

L. Guo
F. Xue
J. Y. B. Lee
CUHK

18 May 2026

Stateful-TCP: Bypassing TCP Slow-Start Using Cached Per-Destination Path Bandwidth
draft-lee-tcpm-stateful-tcp-00

Abstract

This document specifies Stateful-TCP, an experimental sender-side mechanism that accelerates the startup of TCP connections by reusing path-bandwidth information estimated from earlier connections to the same destination. When a usable estimate is available, the sender bypasses Slow-Start and instead enters a paced startup phase whose initial congestion window and pacing rate are derived from the cached estimate. When no usable estimate is available, the sender falls back to standard Slow-Start.

Stateful-TCP is sender-only and does not require any change to the TCP receiver or to the wire format. It is orthogonal to the congestion control algorithm in use after the first round-trip time and may therefore be combined with existing TCP variants such as CUBIC. This document also specifies a Gap-Compensated Bandwidth Estimation (GCBE) procedure used to produce the cached per-destination estimate.

Stateful-TCP extends the framework of TCP Control Block Interdependence (RFC 9040) by sharing additional per-destination state across connections. The mechanism is published as Experimental to enable independent implementation, evaluation, and review.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction

TCP [RFC9293] traditionally begins each new connection with a Slow-Start phase [RFC5681], during which the congestion window (cwnd) starts from a small initial value and grows exponentially as acknowledgments are received. Slow-Start is a conservative bandwidth-probing procedure designed to avoid overwhelming network paths whose capacity is unknown to the sender.

On modern high bandwidth-delay-product (BDP) paths, however, the time spent in Slow-Start can dominate the completion time of short and medium-sized flows. Existing mitigations include increasing the initial window [RFC6928], refining the Slow-Start exit condition (Hystart++ [RFC9406]), and Limited Slow-Start [RFC3742]. These approaches improve but do not eliminate the underlying problem: a sender that has no information about the path is forced to ramp its sending rate up gradually, regardless of how much bandwidth is actually available.

TCP Control Block (TCB) Interdependence [RFC9040] (which obsoletes [RFC2140]) already permits a sender to share a small set of TCB variables, including smoothed RTT and ssthresh, between connections to the same host. This document specifies an experimental extension to that framework. Specifically, an implementation of Stateful-TCP caches an estimated path bandwidth and a minimum RTT per destination and uses them, when available, to bypass Slow-Start on subsequent connections to that destination.

The mechanism has three properties that distinguish it from prior work:

* The startup phase uses both an enlarged initial cwnd _and_ sender pacing. Pacing the first round of transmissions at the previously estimated bottleneck rate prevents the line-rate burst that would otherwise occur and that is the principal cause of buffer overflow when an enlarged initial cwnd is used in isolation.

* The receiver advertised window (rwnd) is suppressed during the startup and estimation phases (except when rwnd is zero, which retains its semantics from [RFC9293]) so that small rwnd at the start of a connection does not cap the sending rate.

* The bandwidth estimation procedure (GCBE, Section 6) excludes inter-ACK gaps that arise from cwnd-limited transmission to avoid potential underestimation in classical bandwidth estimators during Slow-Start.

The design and an evaluation of Stateful-TCP applied to CUBIC [RFC9438] -- referred to in the cited paper as S-Cubic -- are described in [STATEFUL-TCP]. That paper is the normative source of motivation, design rationale, and experimental results; this document defines the on-the-wire-equivalent specification suitable for independent implementation.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terms and symbols are used throughout this document. Units are given in parentheses where applicable.

MSS: TCP Maximum Segment Size, in bytes.

cwnd: Sender congestion window, in bytes (or, equivalently, in units of MSS).

IW: The default initial value of cwnd in the absence of any cached state, as defined by [RFC5681] and updated by [RFC6928].

rwnd: Receiver advertised window, in bytes, as carried in the TCP Window field [RFC9293].

SRTT: Smoothed round-trip time, computed as defined in [RFC6298], in seconds.

RTTmin: Minimum observed RTT for the connection, in seconds.

BDP: Bandwidth-Delay Product, expressed in bytes, equal to the product of an estimated bandwidth and an estimated RTT.

bw_est: The path bandwidth estimate produced by GCBE (Section 6), in bytes per second.

peer_IP: The IP address of the remote endpoint of a TCP connection, as observed by the sender.

State Cache: The sender-local data structure described in Section 4 that maps peer_IP to a tuple {bw_est, RTTmin}.

Startup phase: The phase of a connection running Stateful-TCP that extends from the completion of the three-way handshake until the first ACK that advances the cumulative acknowledgment number is received. See Section 5.

Estimation phase: The phase that follows the Startup phase and lasts until connection termination, during which GCBE updates the bandwidth estimate. See Section 5.

Termination phase: The processing performed when the connection is closed (whether gracefully via FIN or abnormally via RST), during which the cache is updated. See Section 5.

3. Overview

3.1. Design Goals

Stateful-TCP is designed to satisfy the following goals.

* Avoid the bandwidth underutilization caused by Slow-Start on paths whose capacity is already known to the sender from earlier connections.

* Avoid the line-rate startup burst that occurs when an enlarged initial cwnd is used without pacing.

* Require no change to the TCP receiver, no new TCP option, and no change to the wire format.

* Coexist with any existing TCP congestion control algorithm; the mechanism takes effect only during the Startup phase, after which the connection behaves exactly as defined by its congestion control algorithm.

* Degrade gracefully to standard Slow-Start when no usable cached state is available or when a cache lookup collides.

3.2. Architecture

A Stateful-TCP implementation consists of two components that reside in the TCP sender:

* A _State Cache_, which stores {bw_est, RTTmin} per peer_IP across connections and is described in Section 4.

* A _per-connection state machine_, comprising the Startup, Estimation, and Termination phases described in Section 5, that consumes and updates the cache and that drives initial cwnd, pacing, and rwnd suppression.

Bandwidth estimation during the Estimation phase is performed by GCBE, specified in Section 6.

The mechanism extends the framework of [RFC9040]: bw_est and RTTmin are additional, per-destination cached items, used in addition to (not in place of) the items already shared under that framework.

4. The State Cache

4.1. Entry Format

A State Cache entry MUST contain at least the following three fields:

peer_IP: The IP address of the peer for which the entry was recorded. Implementations MUST store the full IP address (including address family) so that hash collisions can be detected as described in Section 4.3.

bw_est: The most recent bandwidth estimate produced by GCBE for the connection that closed and updated this entry.

RTTmin: The minimum RTT observed during that connection.

Implementations MAY store additional fields, such as the last update time (used for expiry) or fields inherited from the TCB Interdependence framework [RFC9040]. Such fields are out of scope of this specification but MUST NOT alter the semantics of the three required fields.

4.2. Indexing

Implementations MAY implement the State Cache as a hash table indexed by an H-bit hash of peer_IP, where H is implementation-defined. Implementations SHOULD choose H, the table size, and the hash function so that the expected hash-collision rate is low for the workload anticipated.

The choice of hash function is local to the sender and is not visible on the wire.

4.3. Lookup Outcomes

A cache lookup performed at the start of the Startup phase produces exactly one of the following three outcomes:

Miss: The hashed bucket is empty. The connection MUST proceed using standard Slow-Start [RFC5681].

Hit: The hashed bucket is non-empty and the stored peer_IP equals the peer_IP of the new connection. The connection MUST apply the Hit branch of the Startup phase as specified in Section 5.1.

Collision: The hashed bucket is non-empty but the stored peer_IP differs from the peer_IP of the new connection. The connection MUST proceed using standard Slow-Start.

In the Miss and Collision cases, the cache entry SHALL be updated by the new connection upon its termination, subject to the conditions in Section 4.4.

4.4. Write Conditions

When a connection terminates (see Section 5.3), an implementation SHALL update the cache entry indexed by the peer_IP of that connection with the {bw_est, RTTmin} pair observed for the connection, except that the cache MUST NOT be updated when:

* bw_est multiplied by RTTmin, expressed in MSS-sized segments, is less than or equal to IW; or

* bw_est is not available (for example, the Estimation phase did not complete a single GCBE measurement cycle); or

* RTTmin is not available (for example, no valid RTT sample was obtained).

In the first case, retaining the entry is unnecessary because Stateful-TCP would not produce any benefit by setting the initial CWnd to below the default IW. In the latter two cases, the values would be invalid for subsequent connections.

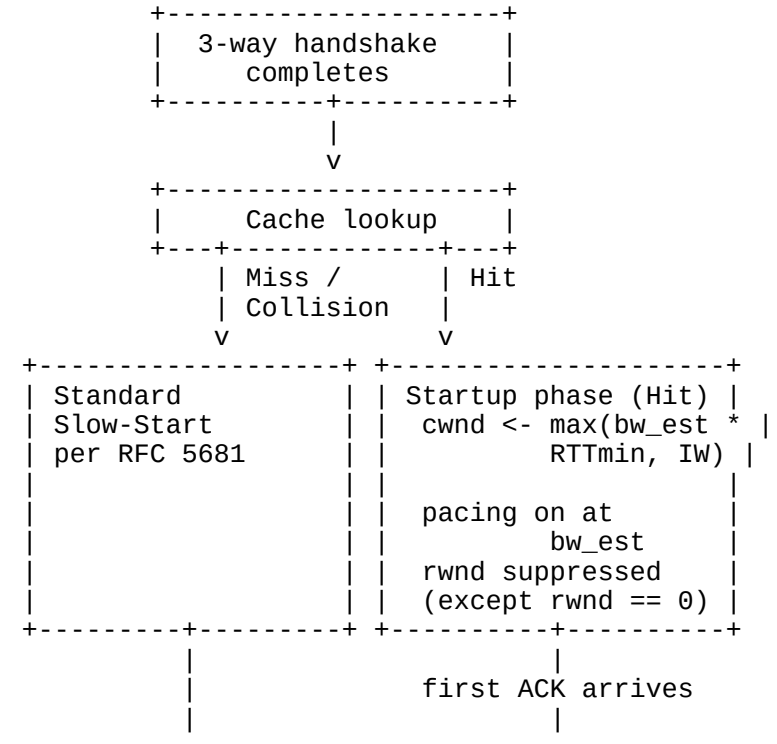
4.5. Expiry and Eviction

Implementations SHOULD associate each cache entry with a maximum age and MUST NOT use an entry whose age exceeds the configured maximum. The maximum age is implementation-defined and SHOULD be configurable.

Implementations MUST bound the memory used by the cache. When the bound is reached, an implementation SHOULD evict entries using a least-recently-used (LRU) policy or an equivalent policy that preserves the entries most likely to be useful. This requirement is also a defence against the cache-exhaustion denial-of-service vector discussed in Section 8.

5. Per-Connection State Machine

Each connection that uses Stateful-TCP transitions through three phases as illustrated in Figure 1.



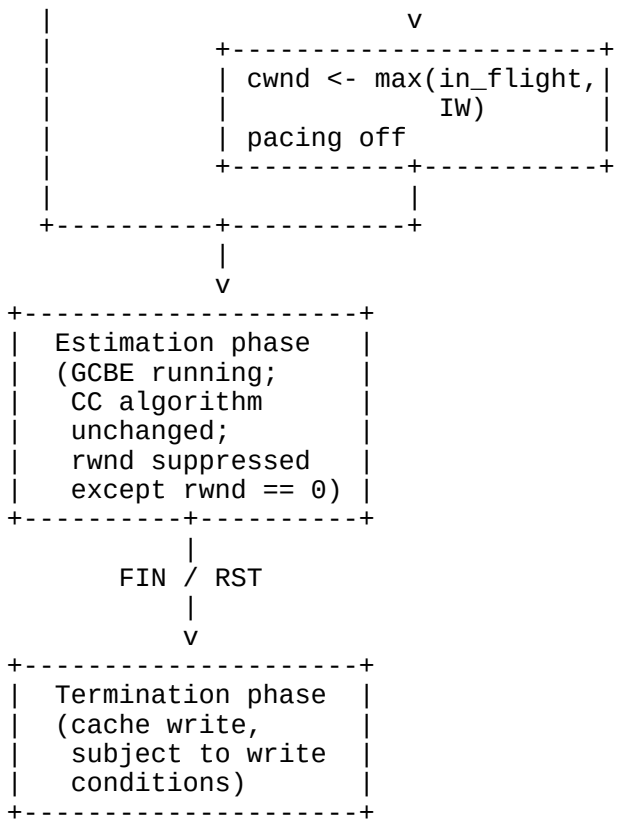


Figure 1: Stateful-TCP per-connection state machine.

5.1. Startup Phase

The Startup phase begins immediately after the three-way handshake completes. The sender MUST perform a State Cache lookup as specified in Section 4.3.

On a _Miss_ or _Collision_ outcome, the sender MUST proceed using standard Slow-Start [RFC5681] and the Startup phase ends.

On a _Hit_ outcome, the sender MUST apply all of the following before transmitting any data segment:

1. Set cwnd to BDP, where BDP is computed as the cached bw_est multiplied by the cached RTTmin and expressed in bytes. The result MUST be at least IW.

2. Suppress the receiver advertised window, as defined in Section 5.4.

3. Enable sender pacing for outgoing data segments at a rate equal to the cached bw_est.

The Startup phase SHALL end when the first ACK that advances the cumulative acknowledgment number is received. At that point the sender MUST:

1. Disable pacing.

2. Set cwnd to the maximum of (a) the number of bytes currently in flight and (b) IW.

The rationale for the cwnd update is that the bytes in flight at this instant approximate the path's BDP.

Hystart-style premature exits from Slow-Start, including Hystart++ [RFC9406], are not applicable during the Startup phase of a Hit-branch connection because the connection is not in Slow-Start. An implementation MUST NOT apply Hystart-style exit triggers to the Hit-branch Startup phase.

5.2. Estimation Phase

The Estimation phase begins immediately after the Startup phase ends and continues until connection termination.

During the Estimation phase the connection's congestion control algorithm operates without modification. The sender MUST continue to suppress rwnd as defined in Section 5.4. The sender MUST run GCBE (Section 6) to maintain bw_est and MUST maintain RTTmin as the minimum of all valid RTT samples observed for the connection.

5.3. Termination Phase

When the connection terminates -- whether gracefully via FIN exchange or abnormally via RST or local abort -- the sender SHALL attempt a cache update as specified in Section 4.4. The update MUST be applied to the cache bucket corresponding to the connection's peer_IP, regardless of whether the connection used the Hit, Miss, or Collision branch in its Startup phase, except that the update MUST NOT be performed if any of the conditions enumerated in Section 4.4 hold.

5.4. Receiver Window Suppression

During the Startup and Estimation phases, the sender MUST compute its effective send window as equal to cwnd, ignoring rwnd, except in the case described below.

When rwnd equals zero, the sender MUST honour the zero window: no new data may be transmitted until the receiver advertises a non-zero rwnd, exactly as required by [RFC9293]. Stateful-TCP MUST NOT override the zero-window semantics.

The justification for suppressing rwnd outside the zero-window case is that on contemporary end systems the receiver is, in practice, almost always able to consume incoming data faster than the network delivers it, so flow-control back-pressure is rarely required; allowing a small initial rwnd to cap the sending rate would limit the performance gains of Stateful-TCP.

6. Gap-Compensated Bandwidth Estimation (GCBE)

6.1. Rationale

A bandwidth estimator that simply divides the bytes acknowledged in a measurement window by the duration of that window may underestimate the path bandwidth if the sender’s transmission has been paused due to cwnd exhaustion. The pause shows up as a long inter-ACK gap, which inflates the denominator of the estimate but not the numerator. This effect is most pronounced when cwnd is small, e.g., during Slow-Start.

GCBE compensates by removing, from each measurement cycle, the single ACK whose inter-arrival time is the longest; both its acknowledged bytes and the gap it spans are excluded from the estimate. Note that GCBE does not replace the congestion control algorithm’s bandwidth estimator. It operates in parallel and the estimated bandwidth is only used for cache update as described in Section 5.3.

6.2. Variables

i: Index of an estimation cycle; i = 0, 1, 2, ...

t_i: Wall-clock time at which cycle i begins.

d: Current SRTT value, in seconds.

n_i: Number of ACKs received during cycle i.

h_{i,j}: Wall-clock arrival time of ACK j in cycle i, for j = 0, 1,

..., n_i - 1.

a_{i,j}: Number of bytes newly acknowledged by ACK j in cycle i.

z_i: Index, within cycle i, of the ACK whose inter-arrival time from its predecessor is the largest.

c_i: Bandwidth estimate produced by cycle i, in bytes per second.

L: Implementation-defined number of recent estimates retained.

m: Total number of completed measurement cycles for the connection.

bw_est: The bandwidth value that will be written to the cache on connection termination.

6.3. Cycle Boundary

Cycle i begins at time t_i. When an ACK is received at time t such that t - t_i is greater than or equal to d (the current SRTT), the cycle SHALL end with that ACK. The next cycle SHALL begin immediately, with t_{i+1} set to t.

6.4. Per-Cycle Estimate

Within cycle i, let

$$z_i = \underset{j \in [1,\, n_i - 1]}{\operatorname{argmax}} \left(h_{i,j} - h_{i,j-1} \right)$$

The ACK at index 0 of the cycle is excluded from the estimate because the time at which the data it acknowledges was placed on the wire is not known to the sender. The ACK at index z_i is excluded because its long inter-arrival gap is taken to be a transmission-suspension gap.

$$c_i = \frac{\sum_{j=1}^{n_i - 1} a_{i,j} - a_{i,z_i}}{\left(h_{i,n_i - 1} - h_{i,0} \right) - \left(h_{i,z_i} - h_{i,z_i - 1} \right)}$$

If n_i is less than or equal to an implementation-defined threshold, then the cycle does not contain enough samples to apply the formula above; the implementation MUST in that case skip the cycle (produce no estimate) and continue with the next cycle.

6.5. Stored Value

The implementation MUST retain the L most recent values of c_i. When the cache is updated at connection termination, the value bw_est SHALL be set to the maximum of the retained values:

```
bw_est = max  c_i
         i in last L cycles of the connection
```

Taking the maximum, rather than the most recent value or a smoothed value, prevents an isolated congestion or loss event near the end of the connection from skewing the recorded estimate downward.

If fewer than L cycles have completed by the time the connection terminates, the maximum is taken over the completed cycles only. If no cycle has completed, bw_est is unavailable and the cache MUST NOT be updated (see Section 4.4).

6.6. Pseudocode

```
# Inputs:
#   - on_ack(ack):
#       called for every ACK that advances the cumulative
#       acknowledgment number, with:
#         ack.t        = arrival time of the ACK
#         ack.bytes    = bytes newly acknowledged by the ACK
#         srtt()       = current SRTT in seconds
#
# Configurations:
#   minACK           # minimum number of ACKs required
#   IW               # initial congestion window (segments)
#   MSS              # maximum segment size (bytes)
# State (per connection):
#   t_i              # start time of current cycle
#   acks[]           # tuples (t, bytes) for the current cycle
#   ring             # ring buffer of size L holding completed c_i
#   RTTmin           # minimum RTT observed (UNAVAILABLE if none)

initialize_gcbe():
    t_i      = now()
    acks     = []
    ring     = empty ring buffer of capacity L

on_ack(ack):
    acks.append( (ack.t, ack.bytes) )

    if (ack.t - t_i) >= srtt():
        # close the current cycle
```

```
            if length(acks) > minACK:
                # find z_i: index of the largest inter-arrival gap
                z = 1
                max_gap = acks[1].t - acks[0].t
                for j in 2 .. length(acks) - 1:
                    gap = acks[j].t - acks[j-1].t
                    if gap > max_gap:
                        max_gap = gap
                        z = j

                sum_bytes = 0
                for j in 1 .. length(acks) - 1:
                    sum_bytes = sum_bytes + acks[j].bytes
                sum_bytes = sum_bytes - acks[z].bytes

                duration  = acks[ length(acks) - 1 ].t - acks[0].t
                duration  = duration - max_gap

                if duration > 0 and sum_bytes > 0:
                    c_i = sum_bytes / duration
                    ring.push(c_i)

            # start a new cycle
            t_i    = ack.t
            acks   = []
            acks.append( (ack.t, ack.bytes) )

    on_connection_close():
        if ring is not empty:
            bw_est = max(ring)
        else:
            bw_est = UNAVAILABLE
        # Section 4.4 write conditions; cache MUST NOT be updated when:
        if bw_est == UNAVAILABLE:
            return NO_CACHE_UPDATE
        if RTTmin == UNAVAILABLE:
            return NO_CACHE_UPDATE
        if (bw_est * RTTmin) / MSS <= IW:
            return NO_CACHE_UPDATE
        return { bw_est, RTTmin }
```

6.7. Edge Cases

* If SRTT changes during a cycle, the change applies starting from the next cycle.

* If the path is non-bottlenecked during a cycle (cwnd never limits transmission), the longest inter-arrival gap identified by the formula is simply one of the regular inter-ACK gaps, and excluding it amounts to dropping a single sample.

* If retransmissions occur within a cycle, the bytes newly acknowledged by an ACK (a_{i,j}) include only the bytes that the cumulative acknowledgment number advances past for the first time; bytes covered solely by SACK ranges [RFC2018] are not counted in a_{i,j}. Implementations that integrate GCBE with modern loss-detection mechanisms such as RACK-TLP [RFC8985] need to apply the same rule.

7. Operational Considerations

7.1. Pacing

A Stateful-TCP sender MUST support sender pacing of outgoing data segments during the Hit branch of the Startup phase, at a rate equal to bw_est. In its absence, transmitting an enlarged initial cwnd at line rate is equivalent to the unpaced initial-cwnd schemes that motivated this work and would be expected to cause buffer overflow at the path bottleneck.

7.2. Relation to RFC 9040 (TCB Interdependence)

[RFC9040] specifies a framework for sharing a small set of TCB variables, including SRTT and ssthresh, between connections to the same host. Stateful-TCP extends this framework with two additional shared items (bw_est and RTTmin) that are used specifically to drive the Startup phase of new connections.

An implementation that already implements [RFC9040] MAY store the additional Stateful-TCP fields in the same per-destination state structure used for RFC 9040 sharing, provided that the conditions in Section 4.4 are observed for the additional fields.

7.3. Relation to TCP Fast Open

TCP Fast Open (TFO, [RFC7413]) reduces startup latency by carrying data in the SYN. TFO and Stateful-TCP are orthogonal: TFO accelerates the first round-trip of a connection, while Stateful-TCP accelerates the bandwidth ramp-up that follows. Implementations MAY enable both simultaneously.

7.4. Fairness

A Hit-branch Stateful-TCP connection ramps to its target rate in one RTT instead of over many RTTs. When such a connection shares a bottleneck with one or more standard (Slow-Start) connections, the standard connections will experience the Stateful-TCP connection as a long-lived flow that is already at steady state, rather than as a newcomer that is still ramping up. This shifts the short-term throughput share towards the Stateful-TCP connection.

The cited evaluation ([STATEFUL-TCP]) quantifies this effect for S-Cubic and reports that, at the link utilizations measured, both fairness among Stateful-TCP connections of the same kind and friendliness to standard connections remain within ranges considered acceptable for an experimental TCP variant. Implementers and operators SHOULD evaluate fairness in their own deployment environments before enabling Stateful-TCP at scale.

7.5. Aged Estimates

The accuracy of a cached estimate degrades as time passes since the estimate was recorded, because path conditions may change. Implementations SHOULD bound the maximum age of an entry as required in Section 4.5. An overestimate larger than the true path bandwidth at the time a Hit-branch connection starts will be corrected by the connection’s ordinary congestion control response after the first RTT, although it may cause additional queueing or retransmissions during the first RTT. An underestimate will reduce the bandwidth utilization of the Hit-branch connection but will not cause additional congestion.

8. Security Considerations

Stateful-TCP introduces sender-local state that is updated and consumed without any additional information on the wire. It does not change the TCP authentication or integrity properties of the connection itself. Nevertheless, the introduction of cached, peer-keyed state raises several considerations.

8.1. Cache Poisoning

The cache is keyed on the peer_IP that the sender observes for the connection that wrote the entry. An off-path attacker cannot directly write entries because doing so would require completing a TCP handshake with the sender while spoofing the victim peer’s IP address, which is not possible without on-path or address-spoofing capability.

An on-path attacker that can complete a TCP handshake from a spoofed peer_IP can induce the sender to record an arbitrary bw_est for that peer. The consequence on a subsequent connection is bounded: an inflated bw_est causes at most one round-trip of paced over-transmission to that peer before ordinary congestion control reasserts itself; a deflated bw_est causes at most one round-trip of under-utilization. Implementations SHOULD consider these bounds when choosing cache expiry policies in environments where on-path attackers are part of the threat model.

8.2. Cache Exhaustion

A peer that initiates connections from many distinct source addresses can cause an unbounded cache to grow without bound. As required in Section 4.5, implementations MUST bound cache memory and SHOULD use an LRU or equivalent eviction policy.

9. IANA Considerations

This document has no IANA actions.

10. Acknowledgments

The mechanism specified here was originally proposed and evaluated in [STATEFUL-TCP].

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <https://www.rfc-editor.org/info/rfc5681>.

[RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <https://www.rfc-editor.org/info/rfc6298>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC9040] Touch, J., Welzl, M., and S. Islam, "TCP Control Block Interdependence", RFC 9040, DOI 10.17487/RFC9040, July 2021, <https://www.rfc-editor.org/info/rfc9040>.

[RFC9293] Eddy, W., Ed., "Transmission Control Protocol (TCP)", STD 7, RFC 9293, DOI 10.17487/RFC9293, August 2022, <https://www.rfc-editor.org/info/rfc9293>.

[RFC9438] Xu, L., Ha, S., Rhee, I., Goel, V., and L. Eggert, Ed., "CUBIC for Fast and Long-Distance Networks", RFC 9438, DOI 10.17487/RFC9438, August 2024, <https://www.rfc-editor.org/info/rfc9438>.

11.2. Informative References

[BBR] Cardwell, N., Cheng, Y., Gunn, C. S., Yeganeh, S. H., and V. Jacobson, "BBR: Congestion-Based Congestion Control", ACM Queue, vol. 14, no. 5, September 2016.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, DOI 10.17487/RFC2018, October 1996, <https://www.rfc-editor.org/info/rfc2018>.

[RFC2140] Touch, J., "TCP Control Block Interdependence", RFC 2140, DOI 10.17487/RFC2140, April 1997, <https://www.rfc-editor.org/info/rfc2140>.

[RFC3742] Floyd, S., "Limited Slow-Start for TCP with Large Congestion Windows", RFC 3742, DOI 10.17487/RFC3742, March 2004, <https://www.rfc-editor.org/info/rfc3742>.

[RFC6928] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP’s Initial Window", RFC 6928, DOI 10.17487/RFC6928, April 2013, <https://www.rfc-editor.org/info/rfc6928>.

[RFC7413] Cheng, Y., Chu, J., Radhakrishnan, S., and A. Jain, "TCP Fast Open", RFC 7413, DOI 10.17487/RFC7413, December 2014, <https://www.rfc-editor.org/info/rfc7413>.

[RFC8985] Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "The RACK-TLP Loss Detection Algorithm for TCP", RFC 8985, DOI 10.17487/RFC8985, February 2021, <https://www.rfc-editor.org/info/rfc8985>.

[RFC9406] Balasubramanian, P., Huang, Y., and M. Olson, "HyStart++: Modified Slow Start for TCP", RFC 9406, DOI 10.17487/RFC9406, May 2023, <https://www.rfc-editor.org/info/rfc9406>.

[STATEFUL-TCP] Guo, L. and J. Y. B. Lee, "Stateful-TCP - A New Approach to Accelerate TCP Slow-Start", IEEE Access, vol. 8, pp. 195955-195970, DOI 10.1109/ACCESS.2020.3032208, October 2020, <https://doi.org/10.1109/ACCESS.2020.3032208>.

[WESTWOOD] Casetti, C., Gerla, M., Mascolo, S., Sanadidi, M. Y., and R. Wang, "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", Proc. ACM MobiCom, pp. 287-297, July 2001.

Authors' Addresses

Lingfeng Guo
The Chinese University of Hong Kong
Department of Information Engineering
Shatin, N.T.
Hong Kong
Email: gl016@ie.cuhk.edu.hk

Feiyu Xue
The Chinese University of Hong Kong
Department of Information Engineering
Shatin, N.T.
Hong Kong
Email: xf024@ie.cuhk.edu.hk

Jack Y. B. Lee
The Chinese University of Hong Kong
Department of Information Engineering
Shatin, N.T.
Hong Kong
Email: jacklee@computer.org