State of the union in TCP land (2026)

Eric Dumazet @ Google

Netdev 0x1a, July 2026

Agenda

01 Overall health status & patch stats

02 AccECN (RFC 9741): Handshake, Options & Byte Counters

03 PSP (Proposal for Security Protocol) Datapath

04 TCP DDP (Direct Data Placement): Hardware-Direct DMA

05 TCP-AO & TCP-MD5 Cryptographic overhaul

06 Out-of-window packet rejection & RFC 7323 window retraction

07 Receive buffer autotuning, OOO queue & RTT adjustments

08 Fast Open, close/RST cleanup & orphan count deprecation

09 BPF TCP iterator snapshotting & tcp_diag modularization

10 Concurrency, KCSAN annotations & Lockless TCP_MAXSEG

11 SKB Drop reasons, Devmem TCP & MPTCP Interop

Linux TCP & MPTCP Activity (2025-2026)

Core TCP Activity

~270 Patches

net/ipv4/tcp*.c, include/net/tcp.h, net/ipv6/tcp_ipv6.c, net/psp

MPTCP Activity

~200 Patches

net/mptcp/

Special Recognition

Huge thanks to top authors, reviewers, and maintainers:

Eric Dumazet, Kuniyuki Iwashima, Matthieu Baerts, Paolo Abeni, Chia-Yu Chang, Ilpo Järvinen, Eric Biggers, Jakub Kicinski, Jiayuan Chen, Jordan Rife, Dmitry Safonov, Geliang Tang, Simon Baatz, Neal Cardwell, Daniel Zahka, Jason Xing, Kees Cook, and others!

AccECN Core Negotiation & SYN/ACK Retransmit

Core Integration

RFC 9741 (Accurate ECN) core integration merged in 6.18 (Ilpo Järvinen, Chia-Yu Chang)

3-Way Handshake Negotiation

- Negotiates **AccECN capability** during SYN/SYN-ACK exchange NEEDS_ACCECN, ECT_1_NEGOTIATION
- Automatically **unsets ECT** if receiving or sending ACE=0 during negotiation

SYN/ACK Retransmit & Downgrade

- Introduced new TCP_SYNACK_RETRANS dynamic synack_type
- **Gracefully retransmits** downgraded SYN or SYN/ACK without AccECN option if peer lacks RFC 9741 support

AccECN Options, Byte Counters & SACK Fitting

RX Byte Counters

- Added `delivered` and `delivered_ce` tracking for precise congestion estimation

Multi-Wrap Heuristics

- Implemented heuristics for `ceb` (Congestion Experienced Bytes) and `cep` (Congestion Experienced Packets)

SACK Option Fitting

- Fits **AccECN option** alongside SACK blocks within the strict **40-byte TCP header limit**

Persistence & Loss Detection

- `TCP_ACCECN_OPTION_PERSIST` handles lost ACKs carrying AccECN options
- Reorganized ECN code into `include/net/tcp_ecn.h` and `tcp_info`

PSP (Proposal for Security Protocol) Datapath

Protocol Integration

- Lightweight host-to-host & container encryption protocol integrated into TCP TX/RX path

TX skb->decrypted Bit Logic

- Marks skb with `skb->decrypted` when PSP key is active at enqueue time
- Drivers only encrypt packets with this bit set, preventing retransmission mismatches

Socket Safety

- Uses `sk->sk_validate_xmit_skb` to ensure PSP skbs cannot escape without validation

Dynamic MSS Adjustment

- Automatically updates TCP MSS to account for PSP header/trailer overhead, avoiding IP fragmentation

TCP DDP: Direct Data Placement Motivation

Memory Bandwidth Bottleneck

- **Payload Copying:** Copying TCP payload (DMA-Write CPU-Read CPU-Write) saturates host memory bandwidth at 100G+ line rates
- **GCU Overhead:** Consumes **>1.2M GCU cycles** in copy functions on host networking stacks

Software RX Zerocopy Overhead

- **Strict Constraints:** Requires 4KB MTUs and strictly page-aligned payloads
- **Page-Table Remapping:** mmap() page-table remapping `(~112K GCUs)` and page dropping `(~429K GCUs)` due to TLB shootdowns and lock contention
- **Small RPCs:** Disabled for small RPCs **(<56KB)**

TCP DDP: Hardware DMA Architecture & API

Hardware-Direct DMA

- NIC DMA-writes TCP payload directly into pre-registered user/GPU virtual memory
- Zero page-table modifications and zero TLB shootdowns during active RX

Separation of Concerns

- **Core TCP State:** PAWS, RTT, ACK generation, and congestion control remains in the kernel

Socket API & Data Path

- **Pre-registration:** `TCP_DDP_ADD_BUFFER` and `REUSE_BUFFER`
- **Data RX:** `epoll POLLIN` to `recvmsg(MSG_TRUNC)`
- **SKB structure:** Page-less DDP fragments (`skb->ddp=1`), `truesize` shrunk to header

TCP DDP: Performance & Security Pairing

CPU & Memory Efficiency

- **~50–66% CPU Reduction:** Compared to software zerocopy (~80% vs non-zerocopy)
- **Flexible MTUs:** Supports arbitrary MTUs (1500, 4KB, 8KB) and 2MB HugePages / THP

Scalability

- **NIC State Tracking:** Per-connection state tracked on NIC (O(1000) flows)
- **Elephant Flows:** Ideal for high-throughput storage and ML data ingestion

Security Pairing

- **Mitigating Injection:** Blind injection attacks (overwriting DDP buffers before host stack header validation) are prevented
- **Cryptographic Pairing:** Achieved by pairing DDP with PSP or TCP-AO authentication and encryption

Cache Locality & Socket Data Structure Shrinking

Cacheline Grouping

- **struct tcp_sock:** Optimized grouping of fields
- **write_tx:** `max_packets_out`, `cwnd_usage_seq`, `rate_delivered`, `rate_interval_us`, `bytes_acked`, `delivered`
- **write_txrx:** `segs_in`, `segs_out`
- **read_tx:** `tcp_clean_acked`, `rcv_tstamp`

Footprint Shrinking

- **64-Byte Savings:** Socket footprint reduced by a full cache line per socket
- **The SLUB Constraint:** `SLAB_TYPESAFE_BY_RCU` under `SLAB_HWCACHE_ALIGN` forced an extra 64B for the freelist pointer
- **Smart Aliasing:** Placed freelist pointer in `sk_freeptr` (aliased with `sk_rcu`)

Queue Optimizations

- **struct request_sock_queue:** Reclaimed 8 bytes in connection queue structure
- **Field Reordering:** Savings achieved strictly by reordering structure fields to close alignment holes

TCP-AO & TCP-MD5 Cryptographic Overhaul

Removal of tcp_sigpool

By Eric Biggers

- **Eliminated Workarounds:** Eliminated per-CPU preallocated tcp_sigpool workaround required by legacy crypto_ahash
- **Direct Library Calls:** Switched to direct lib/crypto/ library calls with stack-allocated MAC and key buffers
- **Zero Allocations:** Zero dynamic allocations in TCP-AO / TCP-MD5 sign and verify paths

Security & Lifecycle

Enhancements & Fixes

- **Side-Channel Prevention:** Strict constant-time MAC comparison (crypto_memneq) preventing side-channel attacks
- **Key Destruction:** Key destruction in .sk_destruct() without RCU grace period delays
- **Critical Fixes:** Fixed UAF in del_async path and NULL pointer dereference with TCP_REPAIR

Out-of-Window Packet Rejection & RFC 7323

Out-of-Window Rejection

Oh well...

- **Sequence Validation:** Validates packet end-sequence in `tcp_sequence()` in both fast and slow paths
- **Flood Prevention:** Prevents buffer bloat and out-of-order queue flooding from malformed/out-of-window packets
- **Drop Reasons:** New MIB & Drop Reason: `LINUX_MIB_BEYOND_WINDOW` and `SKB_DROP_REASON_TCP_OVERWINDOW`
- **Relaxed Constraint:** Constraint relaxed when receive queue is empty to avoid locking up buggy peers

RFC 7323 Window Retraction

By Simon Baatz

- **Receiver Requirements:** Implemented receiver-side requirements when sender retracts advertised window
- **FIN Packet Acceptance:** Re-enabled FIN packet acceptance when `RWIN == 0`, avoiding teardown deadlocks

TCP Receive Buffer, OOO Queue & RTT Adjustments

OOO Receive Buffer Growth

Memory & Estimation Optimizations

- **Refined Growth:** Refined sk_rcvbuf growth for OOO packets to prevent high-loss/reordering flows from over-consuming memory
- **MSS Estimation:** Call `tcp_measure_rcv_mss()` for OOO packets so receiver MSS estimation stays accurate
- **Critical Fix:** Fixed UAF in `tcp_prune_ofo_queue()`

Buffer Bounds & TSO Deferral

Autotuning & Flow Control

- **Lower Bounds:** Enforced lower bounds to prevent setting zero-size receive buffer in autotuning
- **TSO Deferral:** Fixed `tcp_tso_should_defer()` calculation for large RTT connections (>100ms), preventing flow stalls

MPTCP Interoperability & Integration

MPTCP Subsystem Updates

~190 patches in net/mptcp/

- **Buffer Growth:** Made `tcp_rcvbuf_grow()` accessible to MPTCP subflow management
- **Timestamps:** Allowed MPTCP to selectively drop TCP Timestamps for specific subflows
- **Option Size:** Updated `mptcp_established_options()` to return option size (`opt_size`)

Collaboration

Joint Development & Alignment

- **Core Integration:** Strong, ongoing collaboration between core TCP and MPTCP maintainers to ensure seamless upstream integration and robust protocol interoperability.

Socket Lifecycle, Fast Open & RST Refactoring

__tcp_close() & RST

RST Generation Optimization

- **Fast Free:** Switched to `tcp_eat_recv_skb()` in `__tcp_close()` to rapidly free unread socket buffers under attack
- **RFC Compliance:** Refactored close logic to send RST only when strictly required by RFC

Fast Open Cleanup

TCP Disconnect & Removal

- **State Reset:** Clear `tcp_sk(sk)->fastopen_rsk` on `tcp_disconnect()`
- **Code Consolidation:** Moved `reqsk_fastopen_remove` to `tcp_fastopen.c` and eliminated redundant calls

Legacy Hook Removal

Orphan & TW Destructors

- **Cleanups:** Removed global `orphan_count` and `twsk_destructor()`
- **Leak Fix:** Fixed per-CPU `tcp_tw_isn` leak, enabling secure ISN prediction

BPF TCP Iteration Overhaul & tcp_diag

BPF TCP Iterator

Jordan Rife

- **Problem:** Offset-based hash table iteration skipped or repeated sockets during mutation
- **Solution:** Replaced offset tracking with cookie-based bucket snapshotting `bpf_tcp_iter_batch_item`
- **Guarantees:** All existing sockets are visited EXACTLY ONCE; removed `st_bucket_done`

BPF Sockmap & Diagnostics

Performance & Modularization

- **Sockmap Fix:** Fixed `copied_seq` calculation and `FIONREAD` for BPF sockmap
- **Modularization:** Moved TCP diag functions from core inet to `net/ipv4/tcp_diag.c`
- **ACK Timer:** Added delayed ACK timer information to `inet_diag`

Concurrency, KCSAN Annotations & Lockless TCP_MAXSEG

Lockless TCP_MAXSEG

setsockopt(TCP_MAXSEG) Optimization

- **Lockless Path:** Made TCP_MAXSEG lockless via `tcp_sock_set_maxseg()`
- **No Lock Overhead:** Removes unnecessary socket lock requirements to improve concurrency.

KCSAN Annotations

Data-Race Annotations Matrix

- `tp->rx_opt.user_mss`: Lockless setsockopt read
- `icsk_probes_out & retransmits`: Lockless inet_diag reads
- `plb_rehash & timeout_rehash`: Concurrent rehash check
- `tp->srtt_us & write_seq`: Stats / unsent calculation
- `sk_data_ready & write_space`: Callback pointer read

SKB Drop Reasons & Devmem TCP

SKB Drop Reasons & Counters

Drop Tracking & Standardization

- **Memory Pressure:** Added `SKB_DROP_REASON_PFMEMALLOC` for memory pressure drops
- **IPv6 Routing:** Added `SKB_DROP_REASON_IP_OUTNOROUTES` in IPv6 response generation

Devmem TCP Zero-Copy

Debugging & Troubleshooting

- **Error Codes:** Exposed `tcp_recvmsg_locked()` error codes to userland for improved zero-copy devmem troubleshooting

Micro-Optimizations & Micro-Inlining

Memory & Execution

Performance & Size Optimizations

- **Memset Reduction:** Reduced per-packet memset size in `__tcp_transmit_skb()`
- **Micro-Inlining:** Inlined `tcp_chrono_start()`, `tcp_filter()`, and `__tcp_v4_send_check()`
- **Struct Cleanups:** Removed unused `hash_size` field from `struct tcp_out_options`

Alignment

Memory Layout

- **False Sharing:** Moved `tcp_memory_allocated` into `net_aligned_data` to prevent cache-line thrashing