



**POLITECNICO**  
**MILANO 1863**

SCUOLA DI INGEGNERIA INDUSTRIALE  
E DELL'INFORMAZIONE

**NETWORK COMPUTING**  
**Computer Science and Engineering**  
**Ingegneria Informatica**

# **Final Project 2024-25**

Author:

**Shreesh Kumar Jha**

Student ID:

**11022306**

Advisor:

**Prof. Gianni Antichi & Prof. Sebastiano Milano**

Academic Year:

**2024-25**



# Contents

<b>Contents</b>	<b>i</b>
<b>1 TCP Connection Tracker Analysis and Performance Report</b>	<b>1</b>
1.1 Source Code Analysis . . . . .	1
1.2 Critical Bugs Identified . . . . .	2
1.2.1 Faulty TCP Flag Logic . . . . .	2
1.2.2 No RST/FIN Handling . . . . .	2
1.2.3 No Garbage Collection . . . . .	2
1.2.4 UDP Not Supported . . . . .	2
1.3 Corner Cases . . . . .	2
1.4 Implemented Fixes . . . . .	2
1.4.1 TCP Flag Detection . . . . .	2
1.4.2 RST & FIN Handling . . . . .	3
1.4.3 TTL-Based Garbage Collection . . . . .	3
1.4.4 UDP Flow Tracking . . . . .	4
1.5 Performance Metrics . . . . .	4
1.6 Conclusion . . . . .	4



# 1 — TCP Connection Tracker

## Analysis and Performance

### Report

#### 1.1. Source Code Analysis

In this report, I analysed, tested, and debugged the simplified eBPF/XDP TCP connection tracker provided for this project. Originally a basic prototype that evolved into a high-performance system sustaining 3.57 Gbps TCP throughput, I focused on:

- **TCP State Machine & Handshake Logic:** Identified and corrected cases where SYN/SYN-ACK/ACK transitions were mishandled—preventing half-open or simultaneous-open connections from slipping through.
- **RST/FIN Teardown Handling:** Found and fixed missing or incorrect state updates on connection close/reset, ensuring torn-down connections no longer linger in the tracking map.
- **TTL-Based Garbage Collection:** Verified that stale entries are purged reliably, uncovering edge cases where inactive flows persisted past their timeout and tightening expiry checks accordingly.
- **UDP Flow Tracking:** Reviewed minimal UDP “flow” state logic, addressing scenarios where packet bursts could overflow tracking buckets or bypass timeout rules.

Based on this analysis, I implemented targeted fixes—correcting state transitions, closing teardown gaps, reinforcing TTL expiry, and hardening UDP timeouts—to ensure accurate state maintenance under multi-gigabit loads.

## 1.2. Critical Bugs Identified

### 1.2.1. Faulty TCP Flag Logic

**Issue:** Incorrect Boolean expressions prevented SYN-ACK/ACK detection. **Impact:** Broken three-way handshake.

### 1.2.2. No RST/FIN Handling

**Issue:** No connection cleanup on termination. **Impact:** “Zombie” entries persisted indefinitely.

### 1.2.3. No Garbage Collection

**Issue:** Connections never expired. **Impact:** Map exhaustion under churn.

### 1.2.4. UDP Not Supported

**Issue:** All UDP packets dropped. **Impact:** Single-protocol limitation.

## 1.3. Corner Cases

- **Handshake Progression:** SYN\_SENT → SYN\_RECV → ESTABLISHED transitions failed
- **Connection Persistence:** No cleanup mechanism for completed connections
- **Protocol Limitations:** TCP-only support limiting deployment

## 1.4. Implemented Fixes

### 1.4.1. TCP Flag Detection

```
1 // SYN detection
2 if (pkt.flags == TCPHDR_SYN) { /* Insert SYN_SENT */ }
3
4 // SYN+ACK detection
5 if ((pkt.flags & (TCPHDR_SYN | TCPHDR_ACK)) == (TCPHDR_SYN |
    TCPHDR_ACK)) {
```

```
6      /* Transition SYN_SENT -> SYN_RECV */ }
7
8  // ACK detection
9  if (pkt.flags == TCPHDR_ACK) { /* Transition SYN_RECV ->
    ESTABLISHED */ }
```

Listing 1.1: Corrected flag logic

**Result:** Complete three-way handshake functionality.

### 1.4.2. RST & FIN Handling

```
1  // RST handling
2  if (pkt.flags & TCPHDR_RST) {
3      bpf_map_delete_elem(&connections, &key);
4      goto PASS;
5  }
6
7  // FIN state transitions
8  if (saved_state == ESTABLISHED && (pkt.flags & TCPHDR_FIN)) {
9      v->state = FIN_WAIT_1;
10 }
11 if (saved_state == LAST_ACK && pkt.flags == TCPHDR_ACK) {
12     bpf_map_delete_elem(&connections, &key);
13 }
```

Listing 1.2: Connection cleanup

**Result:** Proper connection termination per RFC 793.

### 1.4.3. TTL-Based Garbage Collection

```
1  // Expire check on lookup
2  if (v && v->ttd < now) {
3      bpf_map_delete_elem(&connections, &key);
4      v = NULL;
5  }
6
7  // Set TTL on creation
8  newEntry.ttd = now + TCP_SYN_SENT_TIMEOUT; // 2 minutes
```

Listing 1.3: Automatic cleanup

**Timeouts:** TCP SYN (2min), ESTABLISHED (5 days), FIN (2min), UDP (5-10min).

#### 1.4.4. UDP Flow Tracking

```

1 // New UDP flow
2 newEntry.state = NEW;
3 newEntry.ttl = now + UDP_FLOW_TIMEOUT;
4
5 // Bidirectional detection
6 if (!same_dir) {
7     v->state = ESTABLISHED; // Promote to bidirectional
8 }

```

**Listing 1.4: UDP support**

**Result:** 817 Mbps UDP throughput at 1 Gbps offered (4% loss).

### 1.5. Performance Metrics

Performance Metric	Original	Optimized	Improvement
TCP Throughput (Gbps)	0.283	3.57	12.6x
Connection Stability	1 sec	10+ sec	10x
Packet Retransmissions	5 per test	0	100% reduction
UDP Throughput (Mbps)	0	817	New capability
Protocol Support	TCP only	TCP + UDP	2x protocols

**Table 1.1: Comprehensive performance comparison between original and optimized implementations**

### 1.6. Conclusion

The analysis identified four critical bugs preventing optimal functionality and successfully implemented comprehensive fixes. Key achievements:

- **Performance:** Achieved 3.57 Gbps TCP throughput (12.6x improvement)
- **Protocol Support:** Added UDP with 817 Mbps capability
- **Resource Management:** TTL-based garbage collection prevents map exhaustion
- **Compliance:** RFC 793-compliant TCP state machine implementation