FlexTOE: Flexible TCP Offload with Fine-Grained Parallelism

Rajath Shashidhara ¹, Tim Stamler ², Antoine Kaufmann ³, Simon Peter ¹

¹ University of Washington, ² MPI-SWS, ³ The University of Texas at Austin
High CPU Overhead of TCP

- TCP remains the default protocol in the datacenter
- But TCP stacks have high CPU overhead
  - Even with modern optimized stacks (TAS, Snap, …)

**CPU profile of Memcached with 32B requests/responses**

To go further, we need to offload…
Need for **Flexible** TCP Offload

- **Flexibility**: Datacenter networks evolve rapidly
  - Operators need flexibility for agile development
- **Existing TOEs are hardwired**: slow upgrade cycles

**CPU profile of Memcached with Chelsio Terminator TOE**

- **TCP/IP Stack**
- **Other**
- **POSIX sockets**
- **Application**

Only 16%
TCP Offload:
Can we get *flexibility* and *performance*?
FlexTOE: Flexible, High Performance TCP Offload

- Eliminates all host TCP stack overheads
- Supports POSIX-sockets, DCTCP/Timely congestion control
- Fully extensible (software development velocity), with eBPF support
TCP Offload to SmartNICs - Challenges

SmartNICs are flexible but restrictive:

- Eg: Netronome Agilio, Mellanox BlueField, Pensando DSC, Fungible DPU, ...
- Parallel architectures geared towards stateless offloads
- Many wimpy cores with limited memories

TCP connections are processed sequentially:

- Stateful code paths track in-flight segments
- Stringent per-packet time budgets
- Sensitive to reordering

Traditional TCP stacks perform poorly on SmartNICs
**FlexTOE: Flexible, High-Performance TCP Offload with Fine-grained Parallelism**

To provide **high performance** and **flexibility**, FlexTOE leverages:

- **Modularity**: fine-grained modules keep private state and communicate explicitly
- **Fine-grained parallelism**: Modules may be replicated, sharded, execute out-of-order
- **One-shot data-path offload**: Payload is never buffered on the NIC
**FlexTOE Flexibility: XDP**

Supports eXpress Data Path (XDP) modules implemented in eBPF

- Operate on raw packets
- Shared state via BPF maps

Implemented common datacenter features

- Tracing, Statistics & Profiling
- Connection Firewalling
- VLAN encapsulation/decapsulation
- tcpdump

AccelTCP’s [NSDI20] connection splicing in 24 lines of eBPF at NIC line rate!
FlexTOE Offload Architecture

- SmartNIC
  - Data-path
    - Segment Generation & Transmission
    - Loss Detection & Recovery
    - Payload Transfer
    - Application Notification
    - Flow Scheduling

- Host
  - Control-plane
    - Application Interface Mgmt.
    - Connection Control
    - Congestion Policy
  - Application / libTOE
    - POSIX Sockets
**Data-path**: per-packet transport logic for established connections
FlexTOE Offload Architecture

- **Control-plane**: policy, management and infrequent recovery code-paths
libTOE library: provides POSIX sockets to the application with kernel-bypass
Parallelizing the TCP Data-path for Offload
Parallelizing the TCP Data-path for Offload
Parallelizing the TCP Data-path for Offload

Baseline

Pipeline

Replicated Modules

Replicated Pipeline
Parallelizing the TCP Data-path for Offload
Parallelizing the TCP Data-path for Offload
Parallel TCP Processing Example: Transmit (TX)

TX Seg #1: Alloc, Head, Steer

TX Seg #2: Alloc, Head, Steer

enters the pipeline first
Parallel TCP Processing Example: Transmit (TX)

TX Seg #1: Alloc, Head, Steer, Seq, Pos, Payload, TX

Assign sequence number

TX Seg #2: Alloc, Head, Steer, Seq, Pos, Payload, TX

Time
Parallel TCP Processing Example: Transmit (TX)
Parallel TCP Processing Example: Transmit (TX)
TCP requires processing in-order for loss detection but ...

Data-parallel modules have varying processing times and may reorder segments

Parallel TCP Processing Example: Transmit (TX)
Parallel TCP Processing Example: Transmit (TX)

**FlexTOE:**
Assign sequence number on data-path ingress → reorder segments on egress

![Diagram of parallel TCP processing example]
Evaluation
Evaluation Setup

Intel Xeon Gold 6138 CPU, 20 cores @ 2 GHz with 40GB RAM

Compare:

- **FlexTOE** (flexible offload) on Netronome Agilio CX40 SmartNIC @ 40 Gbps
- **Linux** (in-kernel stack): Intel XL710 @ 40 Gbps
- **TAS** (kernel-bypass): Intel XL710 @ 40 Gbps
- **Chelsio TOE** (inflexible offload): Terminator 6 @ 100 Gbps

**Identical application binaries** across all baselines.
Benefits of Offload: Throughput Scalability

Memcached throughput, varying number of server cores

**FlexTOE saves up to 81% CPU cycles** versus Chelsio and **50%** versus TAS

Offloaded CPU cycles may be used for application work
Benefits of Offload: Low Tail-Latency

Memcached latency distribution across different stack combinations

FlexTOE achieves the lowest median and tail latencies

![Graph showing Memcached latency distribution across different stack combinations.]

Offload provides excellent performance isolation
Is Fine-grained Parallelism Necessary?

Exploiting both intra- and inter-connection parallelism is necessary.
Data-path Parallelism: Does it Generalize across Platforms?

Single connection speedup by 4x on Bluefield (and 2.4x on x86)
FlexTOE: High-performance and Flexible TCP Offload

- Eliminates all host TCP stack overheads to save CPU cycles for the application
- Data-path parallelism via fine-grained modules with out-of-order processing
- Easily extensible with full user-space programmability
  - tcpdump with packet filtering
  - VLAN encap/decap
  - Firewall
  - Connection splicing

FlexTOE is open-source: https://github.com/tcp-acceleration-service/FlexTOE