BOAD: Optimizing Distributed Communication with In-Kernel Broadcast and Aggregation

Jianchang Su[1], Yifan Zhang[1], Linpu Huang[1], Wei Zhang[1]

[1]University of Connecticut

UCONN

Motivation

- Efficient communication is crucial for big data and distributed computing systems
- Broadcasting and aggregation are key communication patterns
- Traditional socket-based methods suffer from significant latency due to user-kernel crossing and network stack processing

Challenges in Distributed Communication

- User-kernel crossings and network stack traversals introduce significant overhead
- Overhead increases with more nodes and larger data sizes
- Scalability and performance bottlenecks in distributed systems

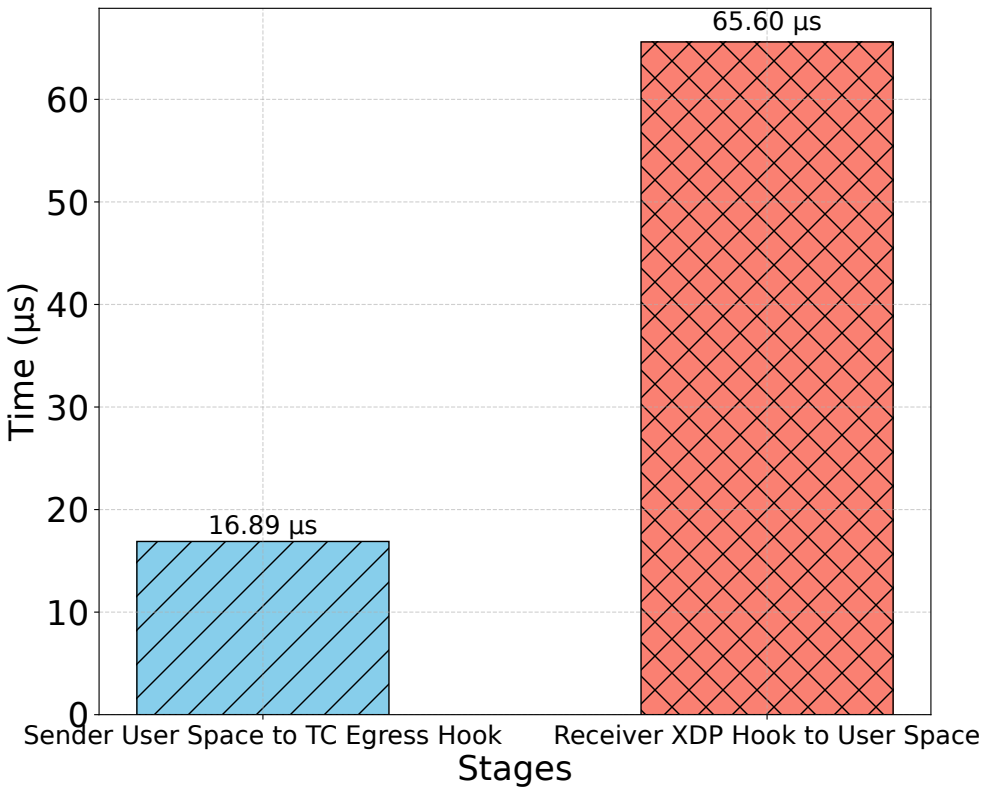


Figure: User-kernel and kernel-user crossing latency

Goal

- Propose BOAD: a system to enhance distributed communication by leveraging eBPF and kernel hooks (XDP and TC)
- Offload broadcasting and aggregation tasks to the kernel space to minimize overhead and reduce latency

Background: eBPF and Kernel Hooks

- eBPF: allows running custom code in the kernel space without modifying the kernel
- XDP: eBPF-based mechanism for high-performance packet processing
- TC: enables traffic control and packet manipulation by attaching eBPF programs

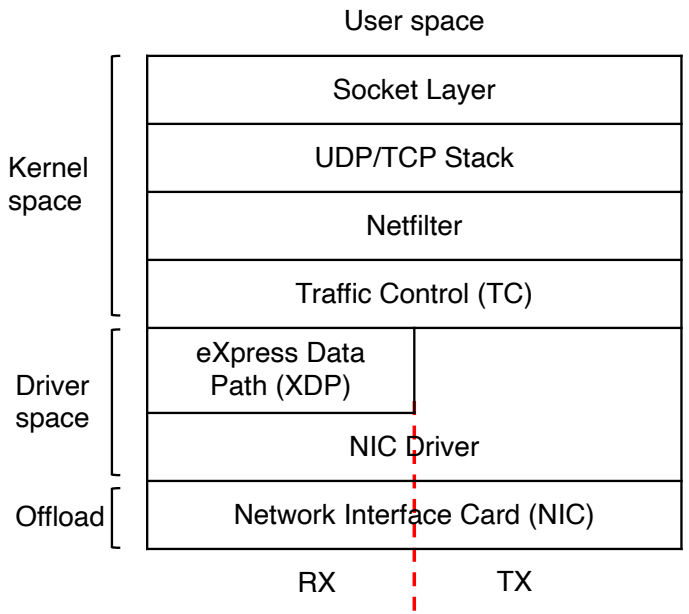


Figure: eBPF, XDP, and TC in Kernel

BOAD Overview

- Runtime library: user-space APIs for kernel interaction
- In-kernel broadcast: eBPF and TC hooks for packet cloning/forwarding
- In-kernel aggregation: eBPF and XDP hooks for data aggregation

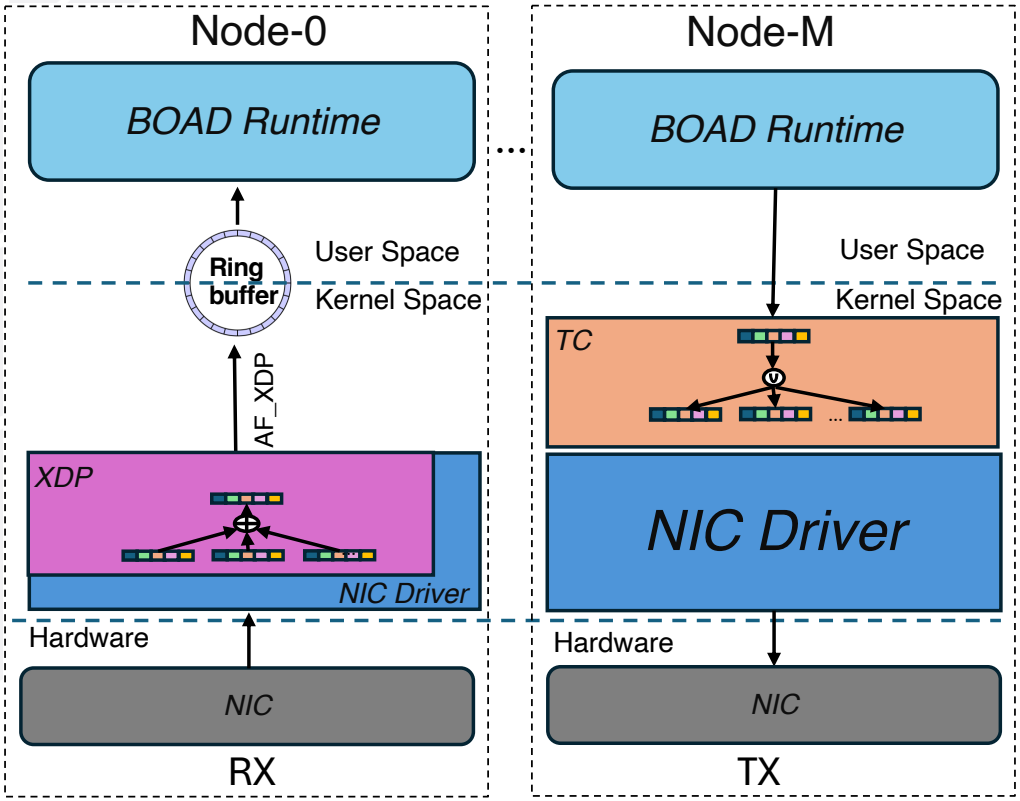


Figure: Architecture of BOAD

Runtime Library and Protocol Design

Runtime Library

- Key functions:
  - `append_machine_config(ip, port)`: Adds machine config to eBPF map
  - `broadcast(data, ip_list)`: Broadcasts data to IP addresses
  - `send_aggregated_data(data)`: Sends data for aggregation
  - `get_aggregated_data()`: Retrieves aggregated data

Broadcast Protocol

- Packet format: `[UDP Header][MAGIC][MASK][SEQ][PAYLOAD]`

Aggregation Protocol

- Packet format: `[UDP Header][MAGIC][KEY][OP][SEQ][PAYLOAD]`

In-kernel Components and Retransmission Mechanism

In-kernel Broadcasting

- Extracts index mask, determines destinations, and forwards packets
- Uses `bpf_clone_redirect()` to clone and forward packets to each destination

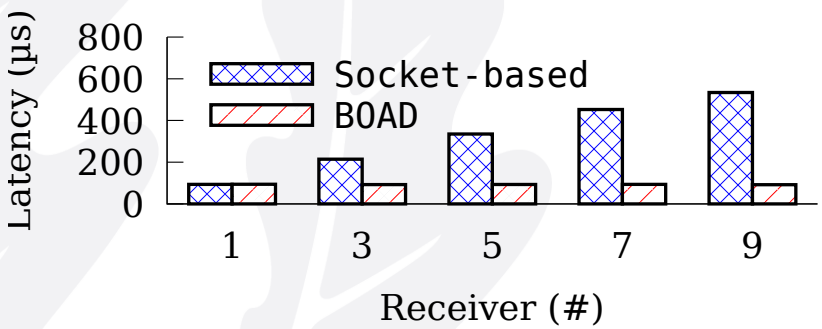
In-kernel Aggregation

- Extracts fields (`KEY`, `OP`, `PAYLOAD`), performs aggregation
- Stores aggregated results in eBPF maps for user-space access

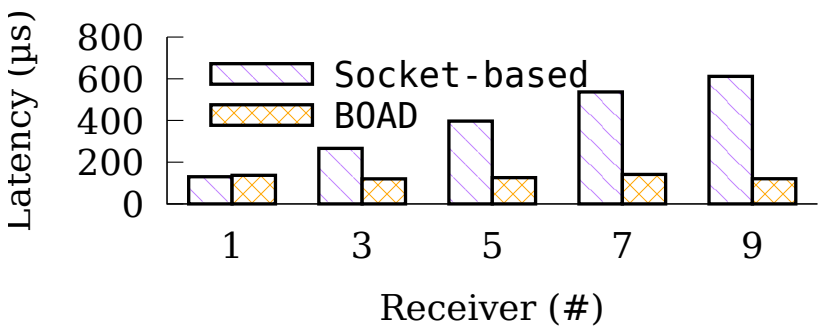
Retransmission Mechanism

- Sender maintains sequence number-packet map in eBPF then receiver detects missing packets and retransmission

Evaluation: Broadcast Latency



(a) Average Broadcast latency



(b) 99th percentile latency

Figure: Comparison of broadcast latencies

- BOAD consistently achieves lower latency compared to the baseline:
  - Up to 82.8% reduction in average latency with 9 receivers
  - Up to 80.3% reduction in 99th percentile latency with 9 receivers

Evaluation: Impact of Packet Size

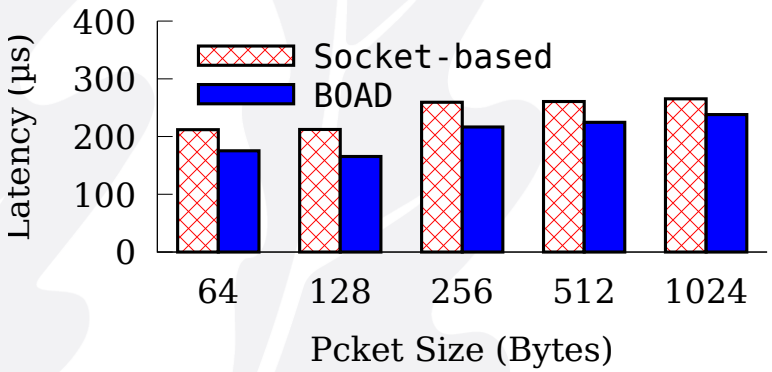
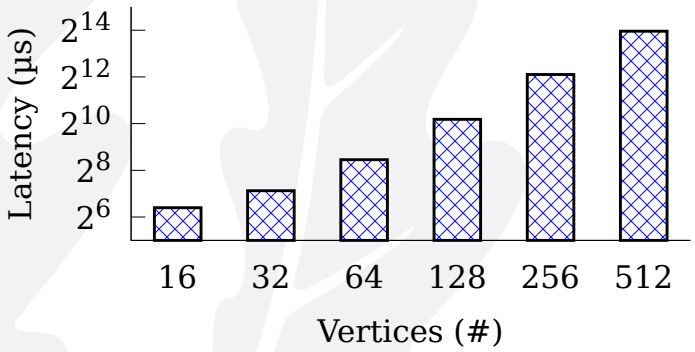


Figure: Broadcast latency with varying packet sizes

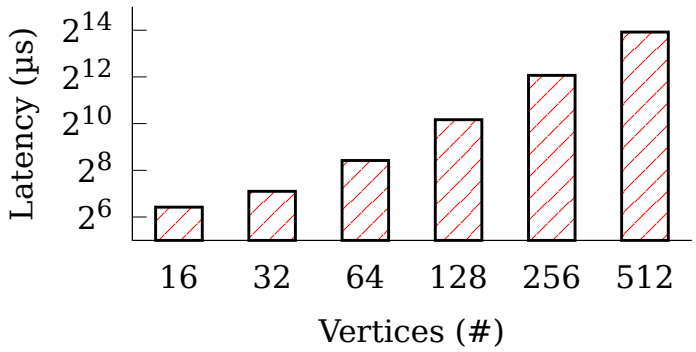
- BOAD maintains significantly lower broadcast latency across all packet sizes
- Broadcast latency remains relatively stable for BOAD

Evaluation: Application-level Performance

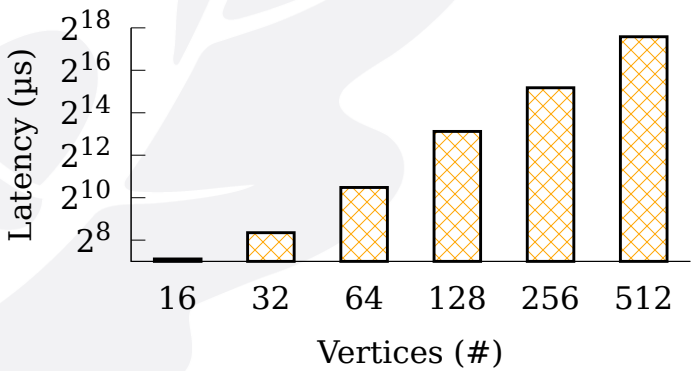
- Integrated BOAD into representative distributed applications: BFS, DFS, PageRank, Fibonacci
- Diverse computational characteristics of these applications



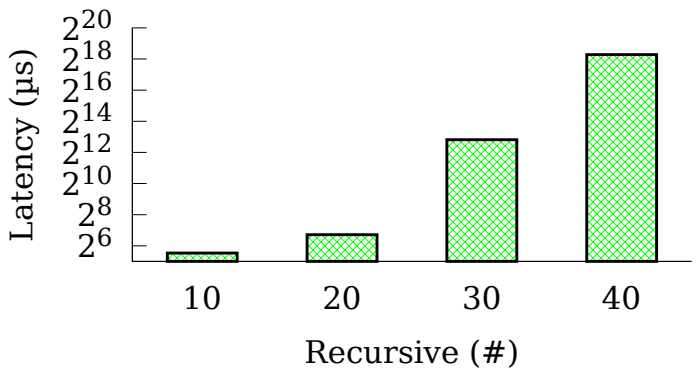
(a) BFS



(b) DFS



(c) PageRank

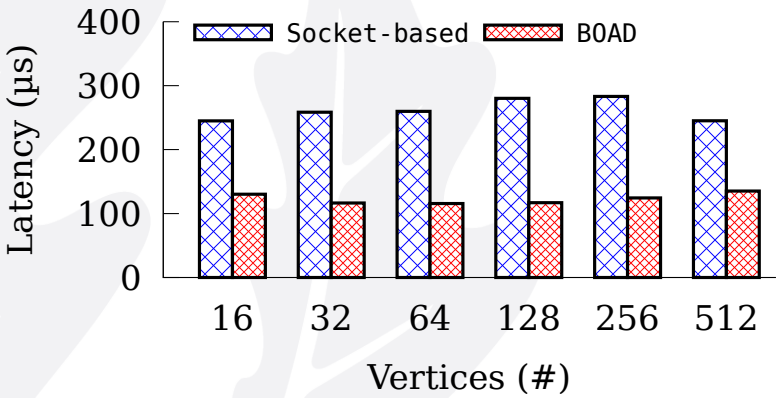


(d) Fibonacci

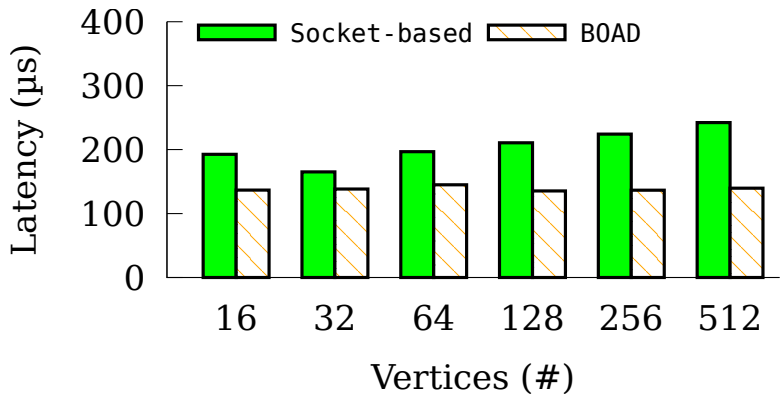
Figure: Computation time for various applications with different input sizes

Evaluation: Broadcast Latency

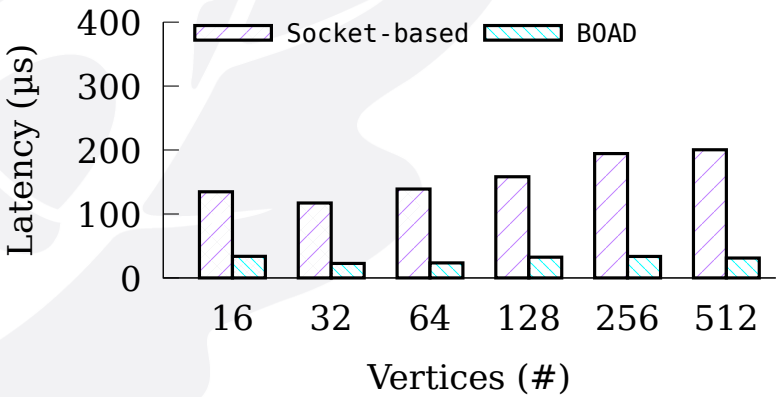
- Up to 84.5% latency reduction for PageRank (512 vertices)
- Major gains when communication overhead dominates



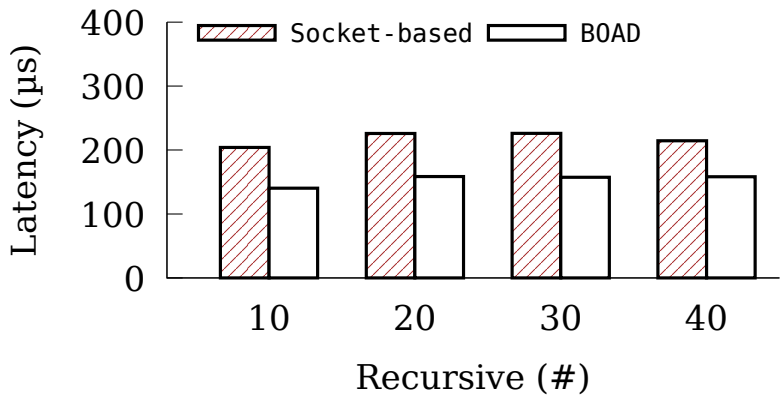
(a) BFS



(b) DFS



(c) PageRank



(d) Fibonacci

Figure: Broadcast latency for various applications with different input sizes

Discussion

- Adapting BOAD to different network architectures and environments
- Enhancing compatibility with different network architectures and providing more flexibility for diverse application requirements

Related Work

- eBPF Applications: Electrode, XAgg, BMC, SPRIGHT, Syrup, XRP
- Distributed in-network computing: ClickINC, Flightplan, ATP
- BOAD focuses on a generic framework for distributing in-network aggregation and broadcast communication without relying on specific network characteristics

Summary

- BOAD leverages eBPF and kernel-level packet processing to optimize broadcasting and aggregation in distributed systems
- Reduces latency and overhead by offloading operations to the kernel and designing efficient protocols
- Future work: adapting to various use cases and integrating with distributed communication libraries

Thank You!