Two trends in high-performance networking
Two trends in high-performance networking

NIC offloads

NIC
- Checksum
- Multiplexing
- Encryption
- Serialization
- Transport
- Segmentation

CPU
Two trends in high-performance networking

NIC
- Checksum
- Multiplexing
- Encryption

CPU

NIC offloads

Offloads operate at higher network layers
Two trends in high-performance networking

**NIC**

- Checksum
- Multiplexing
- Encryption
- Serialization
- Transport
- Segmentation

**CPU**

- DPDK
- netmap
- io_uring
- Arrakis
- mTCP
- TAS
- IX

NIC offloads

*Offloads operate at higher network layers*
Two trends in high-performance networking

NIC offloads

Offloads operate at higher network layers

Efficient network stacks

Often bypass the kernel and rely on batching
Two trends in high-performance networking

NIC offloads

Offloads operate at higher network layers

Efficient network stacks

Often bypass the kernel and rely on batching

Packetized NIC Interface

NIC:
- Checksum
- Multiplexing
- Encryption
- Serialization
- Transport
- Segmentation

CPU:
- DPDK
- netmap
- io_uring
- Arrakis
- mTCP
- TAS
- IX
Mismatch between how NICs are used and the interface that they provide

Fixing this mismatch can significantly improve performance while paving the way for higher-level offloads.
Existing NICs provide a **packetized** interface
Existing NICs provide a packetized interface
Existing NICs provide a packetized interface
Problem #1 Packetized Abstraction

Packetized abstraction is **unsuitable for higher-level offloads**

NIC

RPCs / Application-level messages

Packetized Interface

Host Memory

Packet Buffer

Packet Buffer

Packet Buffer

Application Buffer
Problem #1 Packetized Abstraction

Packetized abstraction is **unsuitable for higher-level offloads**

### NIC

RPCs / Application-level messages

Message

### Host Memory

Packet Buffer

Packet Buffer

Packet Buffer

Application Buffer
Problem #1 Packetized Abstraction

Packetized abstraction is **unsuitable for higher-level offloads**

NIC

RPCs / Application-level messages

Packetized Interface

Host Memory

Message

Packet Buffer

Application Buffer
Problem #1 Packetized Abstraction

Packetized abstraction is unsuitable for higher-level offloads

NIC

RPCs / Application-level messages

Packetized Interface

Host Memory

Packet Buffer

Packet Buffer

Packet Buffer

Message

Application Buffer
Problem #1 Packetized Abstraction

Packetized abstraction is **unsuitable for higher-level offloads**

NIC

Unbounded bytestream (e.g., TCP)

Bytestream

Packetized Interface

Host Memory

Packet Buffer

Packet Buffer

Packet Buffer

Application Buffer
Problem #1 Packetized Abstraction

Packetized abstraction is **unsuitable for higher-level offloads**

NIC

Unbounded bytestream (e.g., TCP)

Host Memory

Bytestream

Packet Buffer

Packetized Interface

Application Buffer
Problem #1 Packetized Abstraction

Packetized abstraction is **unsuitable for higher-level offloads**

NIC

Unbounded bytestream (e.g., TCP)

Host Memory

Packet Buffer

Bytestream

Packetized Interface

Application Buffer
Problem #1 Packetized Abstraction

Packetized abstraction is unsuitable for higher-level offloads

NIC

Unbounded bytestream (e.g., TCP)

Nic...
Problem #1 Packetized Abstraction

Packetized abstraction is **unsuitable for higher-level offloads**

NIC

Unbounded bytestream (e.g., TCP)

---

Host Memory

Packet Buffer

Packet Buffer

Packet Buffer

Bytestream

Application Buffer
Problem #2 Poor Cache Interaction

Poor cache interaction due to chaotic memory access

NIC

Incoming packets

P4  P3  P2  P1

Packetized Interface

Packet Buffer

Host Memory

Packet Buffer

Packet Buffer

Packet Buffer
Problem #2 Poor Cache Interaction

Poor cache interaction due to chaotic memory access

NIC

Incoming packets

Packetized Interface

Host Memory

P1 Packet Buffer

P2 Packet Buffer

P3 Packet Buffer

P4 Packet Buffer

Problem #2 Poor Cache Interaction

Poor cache interaction due to chaotic memory access

NIC

Incoming packets

Packetized Interface

Host Memory

P1 Packet Buffer

P2 Packet Buffer

P3 Packet Buffer

P4 Packet Buffer

Chaotic Memory Access
Problem #2 Poor Cache Interaction

Poor cache interaction due to chaotic memory access

DPDK echo with E810 NIC

Miss ratio (%)

L1d | L2
---|---
0   | 60

55% Miss Ratio for the L2 Cache

Host Memory

Chaotic Memory Access
Problem #3: Metadata Overhead

Overhead (PCIe bandwidth and CPU cycles) due to per-packet metadata
Problem #3 Metadata Overhead

Overhead (PCIe bandwidth and CPU cycles) due to per-packet metadata
Problem #3 Metadata Overhead

Overhead (PCIe bandwidth and CPU cycles) due to per-packet metadata
Problem #3 Metadata Overhead

Overhead (PCIe bandwidth and CPU cycles) due to per-packet metadata
Problem #3 Metadata Overhead

Overhead (PCIe bandwidth and CPU cycles) due to *per-packet metadata*

**Diagram:**
- NIC
  - Incoming packets
    - P4
    - P3
    - P2
- Host Memory
  - Packet Buffer
  - Packet Buffer
  - Packet Buffer
  - Packet Buffer
  - Descriptor Ring Buffer

*Direction arrows:*
- Descriptor Read
- Packet Write
Problem #3 Metadata Overhead

Overhead (PCIe bandwidth and CPU cycles) due to per-packet metadata
Problem #3 Metadata Overhead

Overhead (PCIe bandwidth and CPU cycles) due to per-packet metadata

DPDK echo with E810 NIC

Up to 39% of PCIe bandwidth consumed with metadata
Problem #3 Metadata Overhead

Overhead (PCIe bandwidth and CPU cycles) due to per-packet metadata

DPDK echo with E810 NIC

Up to 39% of PCIe bandwidth consumed with metadata

Similar process to transmit packets
Mismatch between how NICs are used and their interface

#1 Packetized Abstraction

#2 Poor Cache Interaction

#3 Metadata Overhead
Ensō

New interface for NIC-Application Communication
Ensō

New interface for NIC-Application Communication

Key Idea: Streaming abstraction
Ensō

New interface for NIC-Application Communication

**Key Idea:** Streaming abstraction
What is a Streaming Abstraction?
What is a Streaming Abstraction?

Provide the illusion of an unbounded buffer
What is a Streaming Abstraction?

Provide the illusion of an *unbounded* buffer

Packetized Abstraction
What is a Streaming Abstraction?

Provide the illusion of an **unbounded** buffer

Packetized Abstraction

Streaming Abstraction

Unbounded Buffer
What is a Streaming Abstraction?

Provide the illusion of an unbounded buffer
What is a Streaming Abstraction?

Provide the illusion of an unbounded buffer
What is a Streaming Abstraction?

Provide the illusion of an unbounded buffer

Packetized Abstraction

Streaming Abstraction
Flexibility of a Streaming Abstraction

Example 1: NIC with no offloads

NIC → Ensō Pipe → Application
Flexibility of a Streaming Abstraction

**Example 1:** NIC with **no offloads**

- **NIC**
  - Raw Packets
  - P3 P2 P1

- **Application**
  - Ensō Pipe
Flexibility of a Streaming Abstraction

Example 2: NIC that is aware of application-level messages

Diagram:
- NIC
- Application Messages
- Ensō Pipe
- Application
Flexibility of a Streaming Abstraction

**Example 3:** NIC that implements a transport protocol

NIC | Ensō Pipe | Application
Flexibility of a Streaming Abstraction

Example 3: NIC that implements a transport protocol
How to implement a streaming abstraction?
1. How to implement a streaming abstraction?

2. How can a streaming abstraction improve performance?
① How to implement a streaming abstraction?
① How to implement a streaming abstraction?

Provide the illusion of an unbounded buffer
How to implement a streaming abstraction?

Provide the illusion of an unbounded buffer

Each pipe consists of a single contiguous buffer
How to implement a streaming abstraction?

Provide the illusion of an unbounded buffer

Each pipe consists of a single contiguous buffer

We treat this buffer as a ring buffer for data
How to implement a streaming abstraction?

Provide the illusion of an unbounded buffer

Each pipe consists of a single contiguous buffer
We treat this buffer as a ring buffer for data
How to implement a streaming abstraction?

Provide the illusion of an unbounded buffer

Each pipe consists of a single contiguous buffer

We treat this buffer as a ring buffer for data
② How can a streaming abstraction improve performance?
② How can a streaming abstraction improve performance?

Packetized Interface

Poor Cache Interaction

Metadata Overhead
② How can a streaming abstraction improve performance?

- Packetized Interface
- Poor Cache Interaction
- Metadata Overhead

Ensō
② How can a streaming abstraction improve performance?

Packetized Interface

Poor Cache Interaction

Metadata Overhead

Sequential Memory Access
How can a streaming abstraction improve performance?

Packetized Interface

Poor Cache Interaction

Metadata Overhead

Ensō

Sequential Memory Access

Reduces L1 misses by 95.9% and L2 misses by 99.5%
How can a streaming abstraction improve performance?

- **Packetized Interface**
- **Poor Cache Interaction**
- **Metadata Overhead**
- **Sequential Memory Access**
- **Notifying Batches**

Ensō reduces L1 misses by 95.9% and L2 misses by 99.5%.
How can a streaming abstraction improve performance?

- **Packetized Interface**
  - Reduces L1 misses by 95.9% and L2 misses by 99.5%

- **Poor Cache Interaction**

- **Metadata Overhead**

- **Enso**
  - Sequential Memory Access
  - Notifying Batches
  - Reduces PCIe metadata traffic by 96.9%
How often should the NIC notify a batch?
How often should the NIC notify a batch?

**Naïve strategy:** send an update for every piece of data
How often should the NIC notify a batch?

**Naïve strategy:** send an update for every piece of data
How often should the NIC notify a batch?

**Naïve strategy:** send an update for every piece of data

**Problem:** Per-packet overhead
Notification Pacing in Ensō

Ensō combines two techniques

① Reactive Notifications
② Notification Prefetching
Reactive Notifications

The NIC updates its pointer in *reaction* to CPU pointer updates
Reactive Notifications

The NIC updates its pointer in *reaction* to CPU pointer updates.
① Reactive Notifications

The NIC updates its pointer in *reaction* to CPU pointer updates.
① Reactive Notifications

The NIC updates its pointer in *reaction* to CPU pointer updates.
Reactive Notifications

The NIC updates its pointer in *reaction* to CPU pointer updates.
① Reactive Notifications

The NIC updates its pointer in *reaction* to CPU pointer updates.
Reactive Notifications

The NIC updates its pointer in *reaction* to CPU pointer updates.
Reactive Notifications

The NIC updates its pointer in *reaction* to CPU pointer updates.
① Reactive Notifications

The NIC updates its pointer in *reaction* to CPU pointer updates.

Only sends notifications that are strictly necessary.
Problem: PCIe Latency

Software may need to wait up to 1 PCIe RTT for a notification
Problem: PCIe Latency

Software may need to wait up to 1 PCIe RTT for a notification
Problem: PCIe Latency

Software may need to wait up to 1 PCIe RTT for a notification.
Problem: PCIe Latency

Software may need to wait up to 1 PCIe RTT for a notification
Problem: PCIe Latency

Software may need to wait up to 1 PCIe RTT for a notification
Problem: PCIe Latency

Software may need to wait up to 1 PCIe RTT for a notification.
Notification Prefetching

Software can *explicitly* request pointer updates from the NIC.
Software can *explicitly* request pointer updates from the NIC.
Software can *explicitly* request pointer updates from the NIC.
Software can *explicitly* request pointer updates from the NIC.
③ Notification Prefetching

Software can *explicitly* request pointer updates from the NIC.
Many other design challenges...

How to notify pointer updates efficiently?

How to deal with data that wrap around?

How to design a scalable hardware?

How to avoid copies in applications that send data back (e.g., Network Functions)?
Many other design challenges...

- How to notify pointer updates efficiently?

- How to deal with data that wrap around?

- How to design a scalable hardware?

- How to avoid copies in applications that send data back (e.g., Network Functions)?

Refer to the paper for details
Ensō Implementation

Hardware

FPGA

Software

CPU
Ensō Implementation

Hardware

FPGA

Ensō NIC (SystemVerilog)

Software

CPU
Ensō Implementation

**Hardware**
- FPGA
- Ensō NIC (SystemVerilog)

**Software**
- CPU
- Kernel Module (C)
- Ensō Library (C++17)
Evaluation

Machine 1
(Packet Generator)

- CPU
- EnsōGen
  Packet Generator
- NIC
  Ensō NIC

Machine 2
(Design Under Test)

- NICs
  - Ensō NIC
  - E810 NIC
- CPU
  - Application with Ensō Lib
  - Application with DPDK
Ensō achieves 100 Gbps line rate (148.8 Mpps) using a single core.
Ensō achieves 100Gbps line rate (148.8 Mpps) using a single core

“Impressive results. Soundly destroys DPDK for many of the types of microbenchmark applications that are popular in the academic literature [...]” — Reviewer D
Ensō improves application throughput by **up to 6x**

<table>
<thead>
<tr>
<th>Application</th>
<th>Throughput Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maglev Load Balancer [NSDI ’16]</td>
<td>Up to 6x</td>
</tr>
<tr>
<td>Network Telemetry with NitroSketch [SIGCOMM ’19]</td>
<td>Up to 3.5x</td>
</tr>
<tr>
<td>MICA Key-Value Store [NSDI ’14]</td>
<td>Up to 47%</td>
</tr>
<tr>
<td>Log Monitor</td>
<td>Up to 95%</td>
</tr>
</tbody>
</table>
Reactive Notifications + Notification Prefetching improve throughput without impairing latency

![Graph showing latency vs offered load for Ensō with and without reactive notifications]
Ensō achieves similar latency to the E810 NIC with DPDK, while sustaining a much greater load.
Ensō outperforms the packetized interface even when copying data.
Conclusion

Ensō is a **streaming interface** for NIC-Application communication
Conclusion

Ensō is a **streaming interface** for NIC-Application communication

Improves application throughput by **up to 6x** even with no offloads
Conclusion

Ensō is a **streaming interface** for NIC-Application communication

Improves application throughput by **up to 6x** even with no offloads

Allows easier and more efficient **high-level offload** implementations
Conclusion

Ensō is a streaming interface for NIC-Application communication

Improves application throughput by up to 6x even with no offloads

Allows easier and more efficient high-level offload implementations

Ensō is open source: enso.cs.cmu.edu

Contact: sadok@cmu.edu