Fast-Path Data Movement for IP Transports

A Tour of the Alternatives

Jeff Chase

Duke University

The Big Questions

- Can/should IP be competitive in “high performance” domains (e.g., “SAN” markets)?
  - iSCSI vs. FibreChannel
  - IP over 10+GE vs. Infiniband
  - IP offers many advantages
- What will it take to get us there?
- What does this have to do with IETF?

Focusing on the Issue

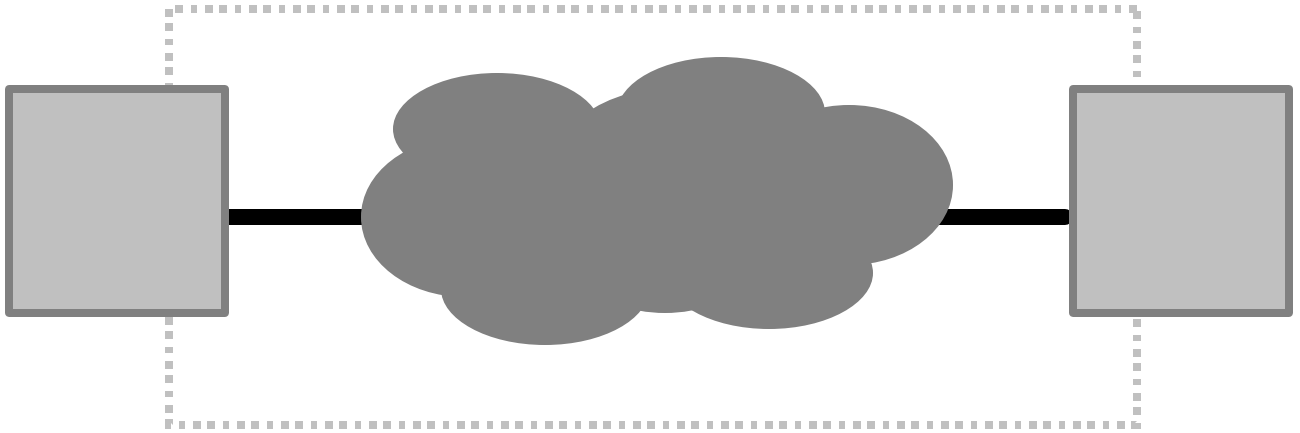
- The key issue IS NOT:
  - *The pipes:* Ethernet has come a long way since 1981.
    - Add another zero every three years?
  - *Transport architecture:* generality of IP is worth the cost.
  - *Protocol overhead*: run better code on a faster CPU.
  - *Interrupts, checksums, etc*: the NIC vendors can innovate here without us.

All of these are part of the bigger picture, but we don't need an IETF working group to "fix" them.

The Copy Problem

- The key issue IS *data movement within the host.*
  - Combined with other overheads, copying sucks up resources needed for application processing.
- The problem won't go away with better technology.
  - Faster CPUs don't help: it's the memory.
- General solutions are elusive…on the receive side.
- The problem exposes basic structural issues:
  - interactions among NIC, OS, APIs, protocols.

What is IETF's Role?



- Can we fix it locally within the end systems?
- To what degree does it involve the protocols?
- Haven't we been doing zero-copy TCP etc. for decades?

“Zero-Copy” Alternatives

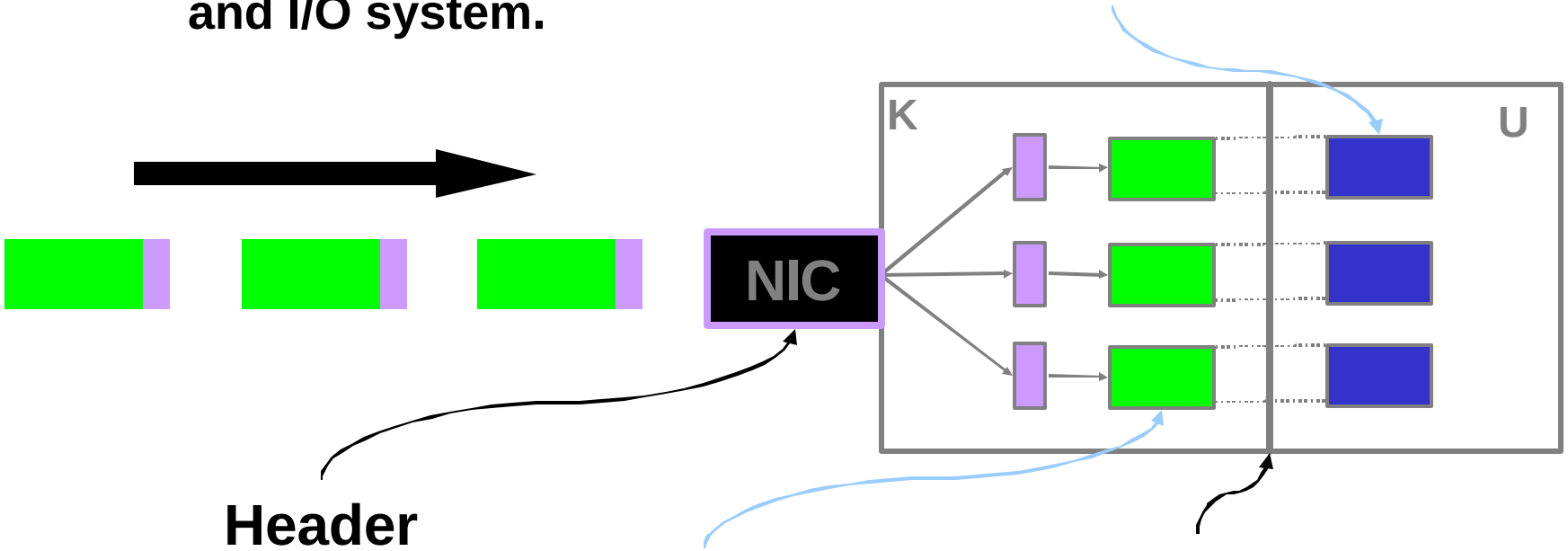
- Option 1: page flipping
  - NIC places payloads in aligned memory; OS uses virtual memory to map it where the app wants it.
- Option 2: scatter/gather API
  - NIC puts the data wherever it want; app accepts the data wherever it lands.
- Option 3: direct data placement
  - NIC puts data where the headers say it should go.

Each solution involves the OS, application, and NIC to some degree.

Page Flipping: the Basics

Goal: deposit payloads in aligned buffer blocks suitable for the OS VM and I/O system.

Receiving app specifies buffers (per RFC 793 copy semantics).



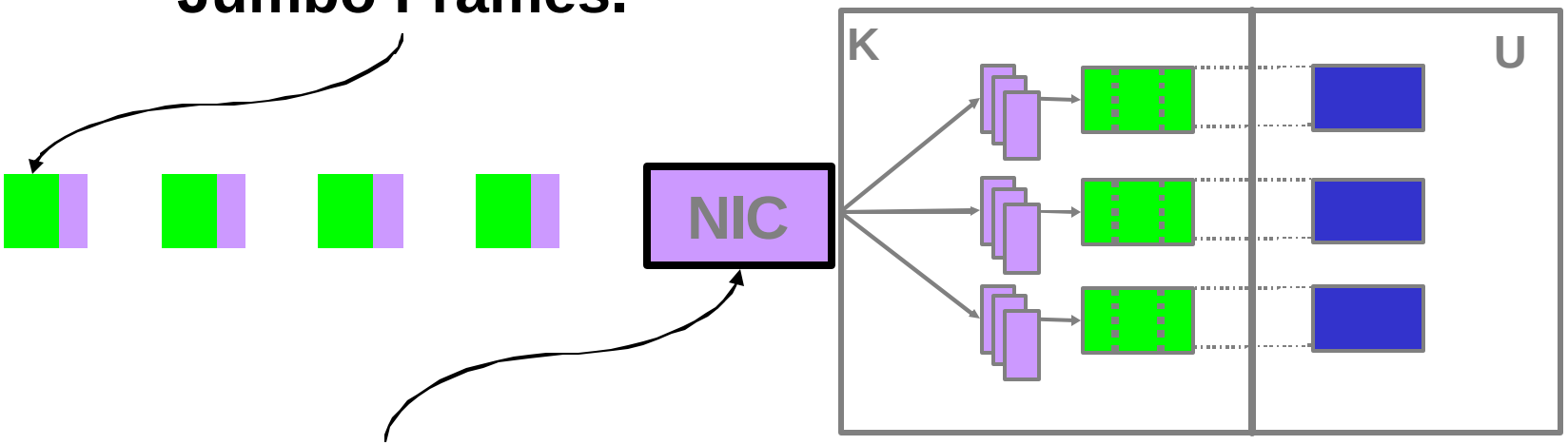
Header splitting

Aligned payload buffers

VM remaps pages at socket layer

Page Flipping with Small MTUs

Give up on
Jumbo Frames.

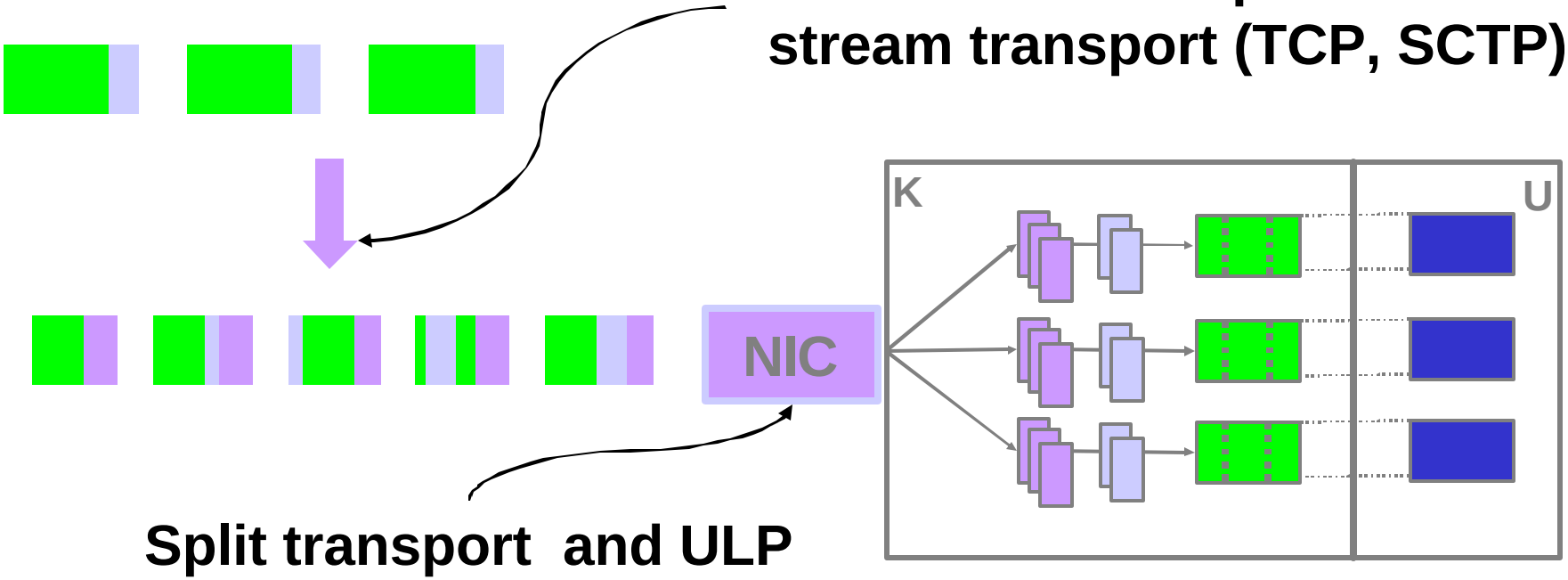


Split transport headers, sequence and coalesce payloads for each connection/stream/flow.

Host

Page Flipping with a ULP

ULP PDUs encapsulated in stream transport (TCP, SCTP)



Split transport and ULP headers, coalesce payloads for each stream (or ULP PDU).

Host

<u>Example</u>: an NFS client reading a file

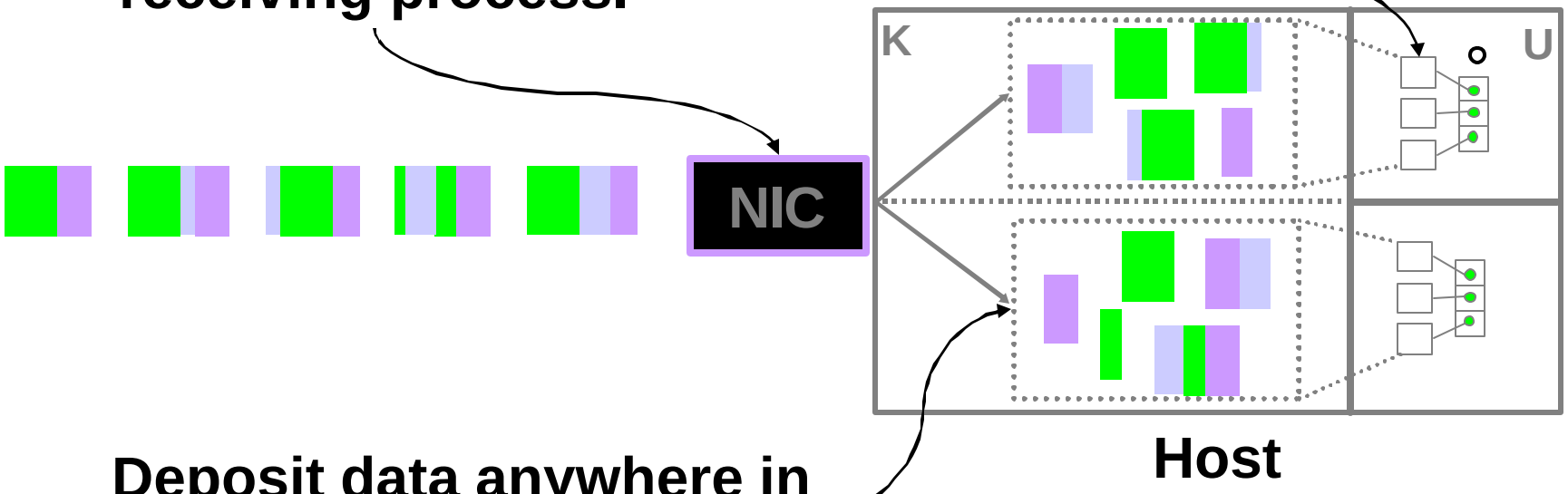
Page Flipping: Pros and Cons

- Pro: sometimes works.
  - Application buffers must match transport alignment.
- NIC must split headers and coalesce payloads to fill aligned buffer pages.
- NIC must recognize and separate ULP headers as well as transport headers.
- Page remap requires TLB shootdown for SMPs.
  - Cost/overhead scales with number of processors.

Option 2: Scatter/Gather

System and apps see data as arbitrary scatter/gather buffer chains (readonly).

NIC demultiplexes packets by ID of receiving process.



Host

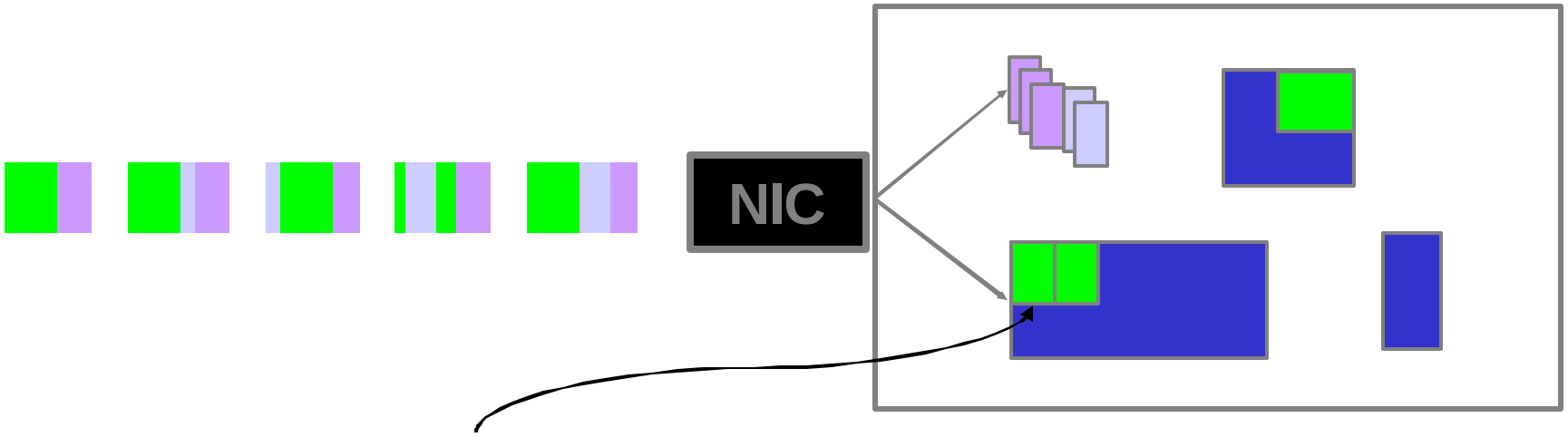
Deposit data anywhere in buffer pool for recipient.

Fbufs and IO-Lite [Rice]

Scatter/Gather: Pros and Cons

- Pro: just might work.
- New APIs
- New applications
- New NICs
- New OS
- May not meet app alignment constraints.

Option 3: Direct Data Placement



NIC “steers” payloads directly to app buffers, as directed by transport and/or ULP headers.

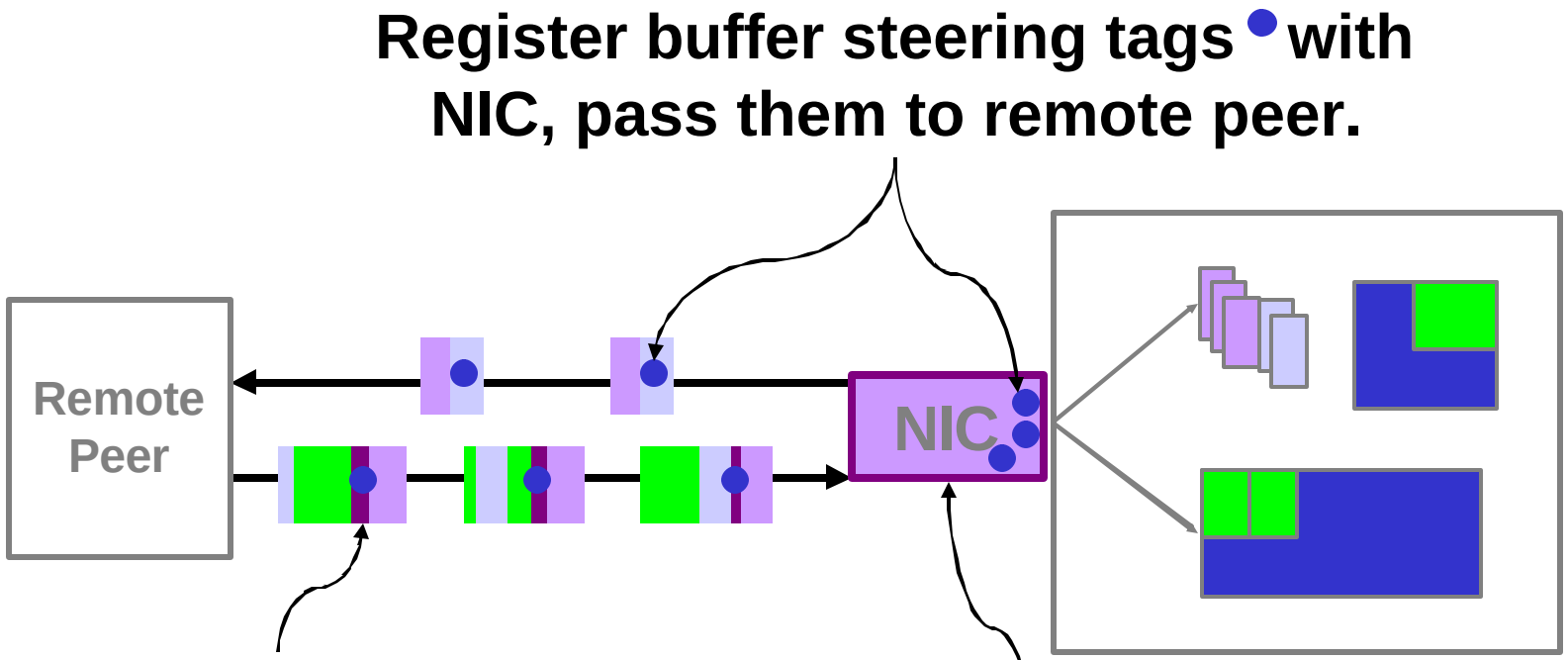
DDP: Pros and Cons

- Effective: deposits payloads directly in designated receive buffers, without copying or flipping.
- General: works independent of MTU, page size, buffer alignment, presence of ULP headers, etc.
- Low-impact: if the NIC is “magic”, DDP is compatible with existing apps, APIs, ULPs, and OS.
- Of course, there are no magic NICs…

DDP: Examples

- TCP Offload Engines (TOE) can steer payloads directly to preposted buffers.
  - Similar to page flipping (“pack” each flow into buffers)
  - Relies on preposting, doesn’t work for ULPs
- ULP-specific NICs (e.g., iSCSI)
  - Proliferation of special-purpose NICs
  - Expensive for future ULPs
- RDMA on non-IP networks
  - VIA, Infiniband, ServerNet, etc.

Remote Direct Memory Access



The Case for RDMA over IP

- RDMA-like functions offer a general solution for fast-path data movement.
  - New transport functions to enhance performance.
- Lots of experience with RDMA on non-IP interconnects.
  - RDMA is an accepted direct-access model.
  - Protocols and APIs already exist to use RDMA.
- Leverages IP infrastructure, generality, cost.
  - IP everywhere
- RDMA NICs are general across a range of apps and ULPs.

The Case Against

- Requires a standard protocol for direct data placement over IP transports.
  - Interoperability
  - Must leverage and conform to existing/future framework for security and management.
- Requires some extension of ULPs or apps to use it.
    - Low to moderate
- “No RDMA protocol exists that can solve X.”

Summary/Conclusion

- Avoiding receiver copies is a complex problem.
  - No easy answers, but partial solutions abound.
- General fast-path data movement can benefit any protocol/application that moves a lot of data.
  - Position IP as competitive for high-performance networking;
  - currently effective only for NIC-based ULPs.
- RDMA standards will promote a market for *interoperable* general-purpose direct data placement NICs.
  - Build on growing market for IP NICs (TOE, iSCSI, VI)
  - Generalize to a wide range of ULPs and future ULPs.

RDMA Issues

- Negotiating RDMA functionality at connection setup
- Buffer registration and protection
  - Memory semantics vs. steering semantics
  - Persistence of buffer tags
  - Access control granularity
- Read vs. write
- Round trip costs and RDMA buffer “window” on LFNs
- Standardize NIC interface to host, or value diversity?
- Interaction of protocols and NIC implementation
  - Efficient implementation is paramount (cost and performance).
  - Requires support for out-of-order data placement?
- Related messaging semantics

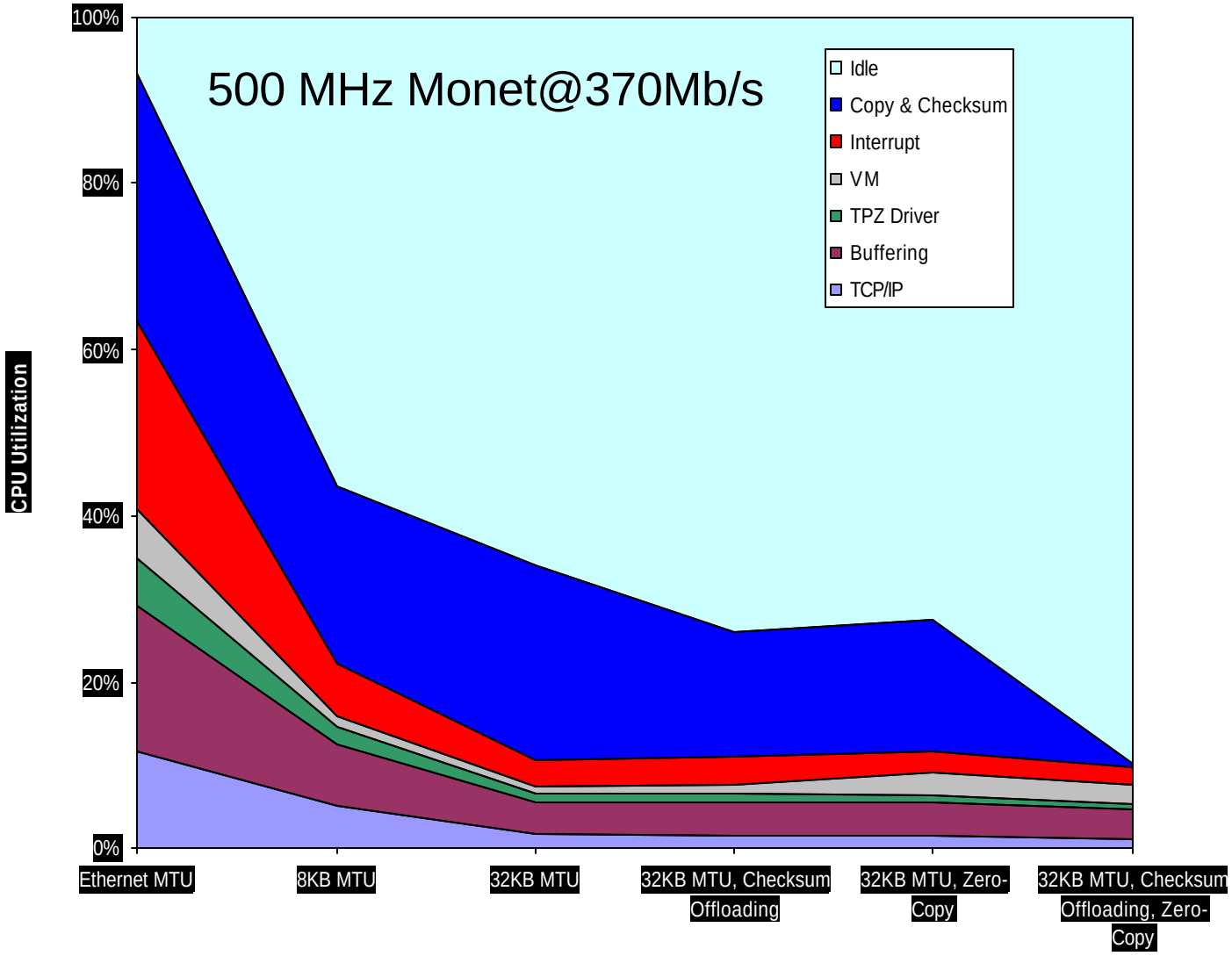
SCTP?

- IP transports should be considered on an equal basis with respect to fast-path data movement.
- Everything I have said applies equally to SCTP.
  - Same problem, same solutions.
- SCTP stream IDs could be used for demux…
  - …just like early demux for TCP connections.
  - Separate the ULP headers on a different stream?
  - Requires close scrutiny + still have to flip or DDP.
- SCTP could be an excellent basis for RDMA.
  - Could enable out-of-order placement given a chunk type extension for RDMA.

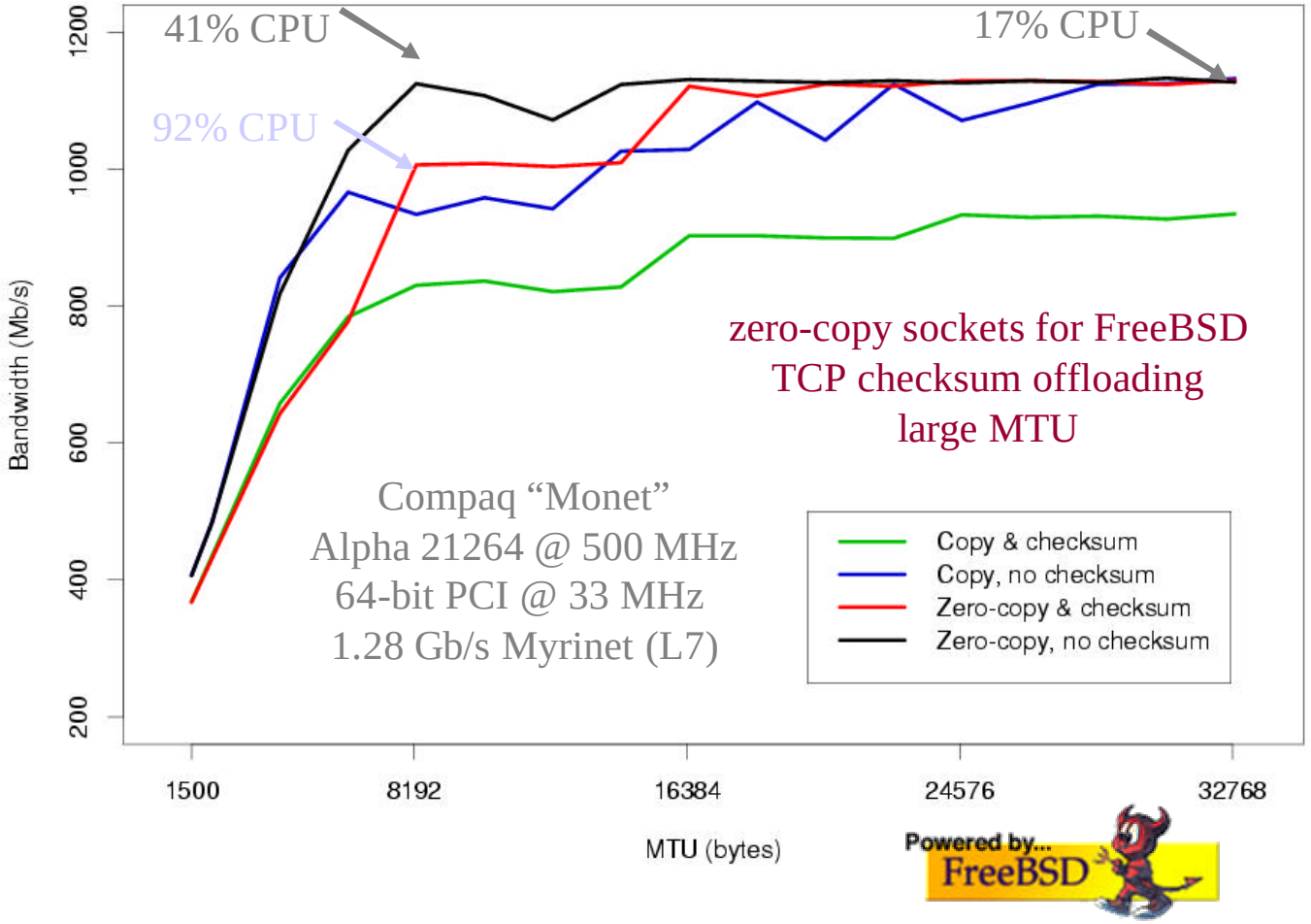
Duke Experiments

- Extend a public-domain Unix kernel with high-speed networking optimizations known in the literature.
  - FreeBSD 4.0...
  - ...extended for zero-copy sockets and checksum offloading.
  - Zero-copy TCP and NFS/UDP
  - Tigon GE and Trapeze/Myrinet
- Use *netperf* to benchmark TCP/IP on different configurations, using *iprobe* to profile host CPU activity.
  - 2 types of PCs and 2 types of Alphas
  - 32- and 64-bit PCI
  - vary MTUs from 1500 bytes up to 32KB
- Experiment with various combined optimizations, and study the numbers.

CPU Overhead for TCP/IP



Trapeze/L7 TCP/IP Results



Netperf/TCP at 2 Gb/s