Federated Architectures
Lecture 9

Kernel Interfaces

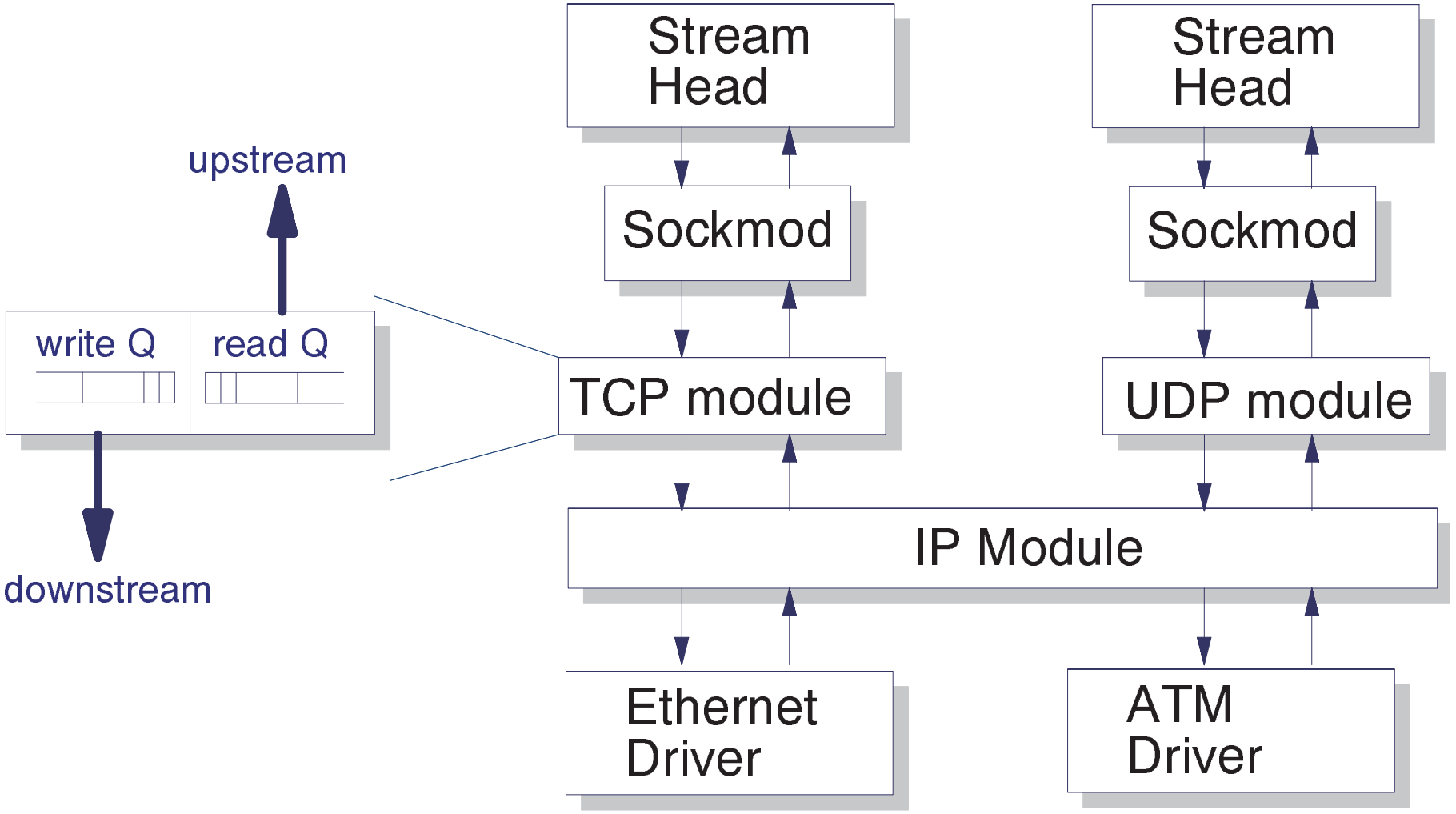
Today

- Kernel Interfaces
  - What are the internal interfaces inside the OS kernel for the filesystem and networking?

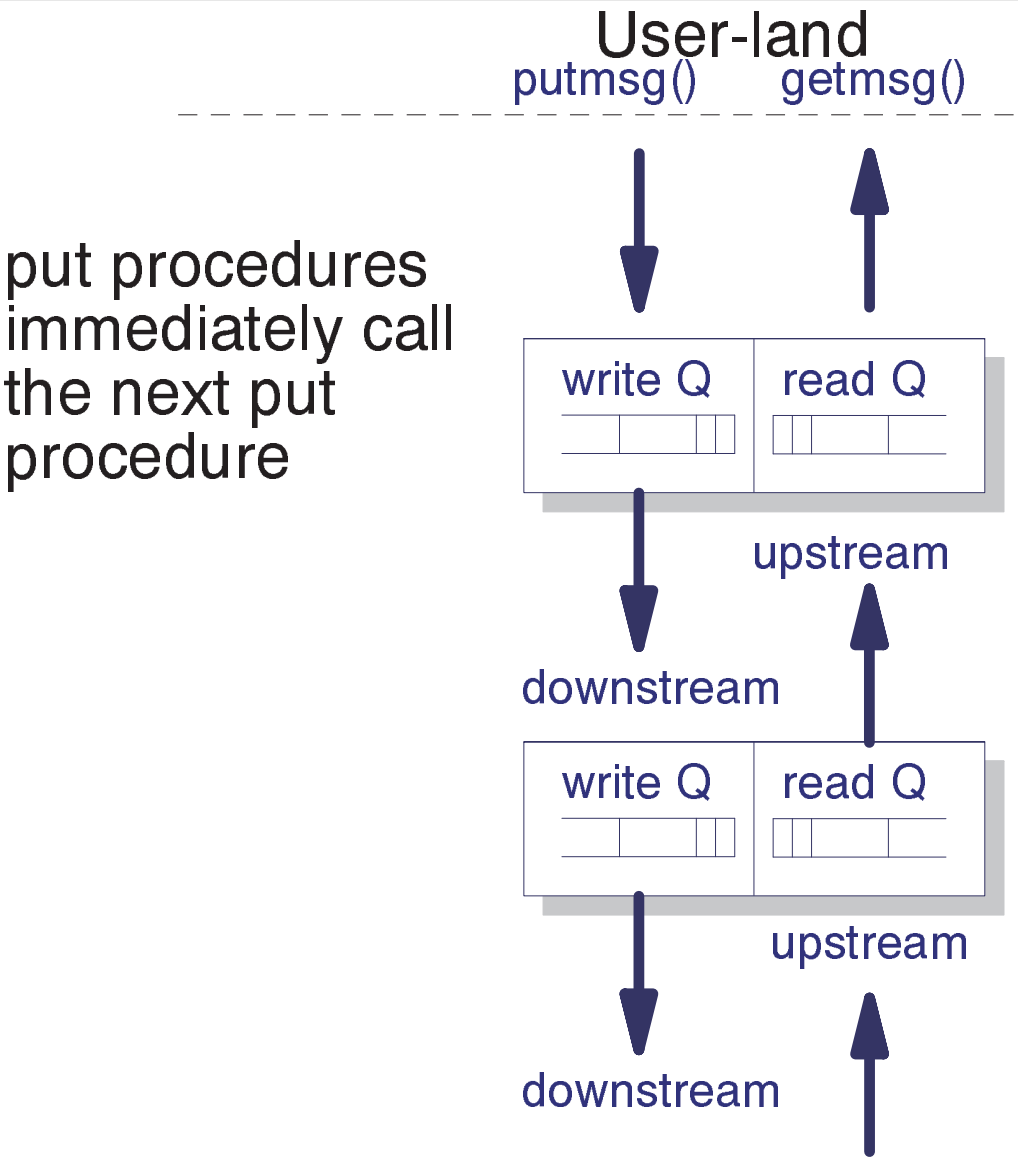
- Review of Streams and *BSD networking
- Demultiplexed architectures
- Review of Vnodes
- Critique of vnodes

SYSV STREAMS framework

- Generic Framework for network protocols



SYSV STREAMS framework



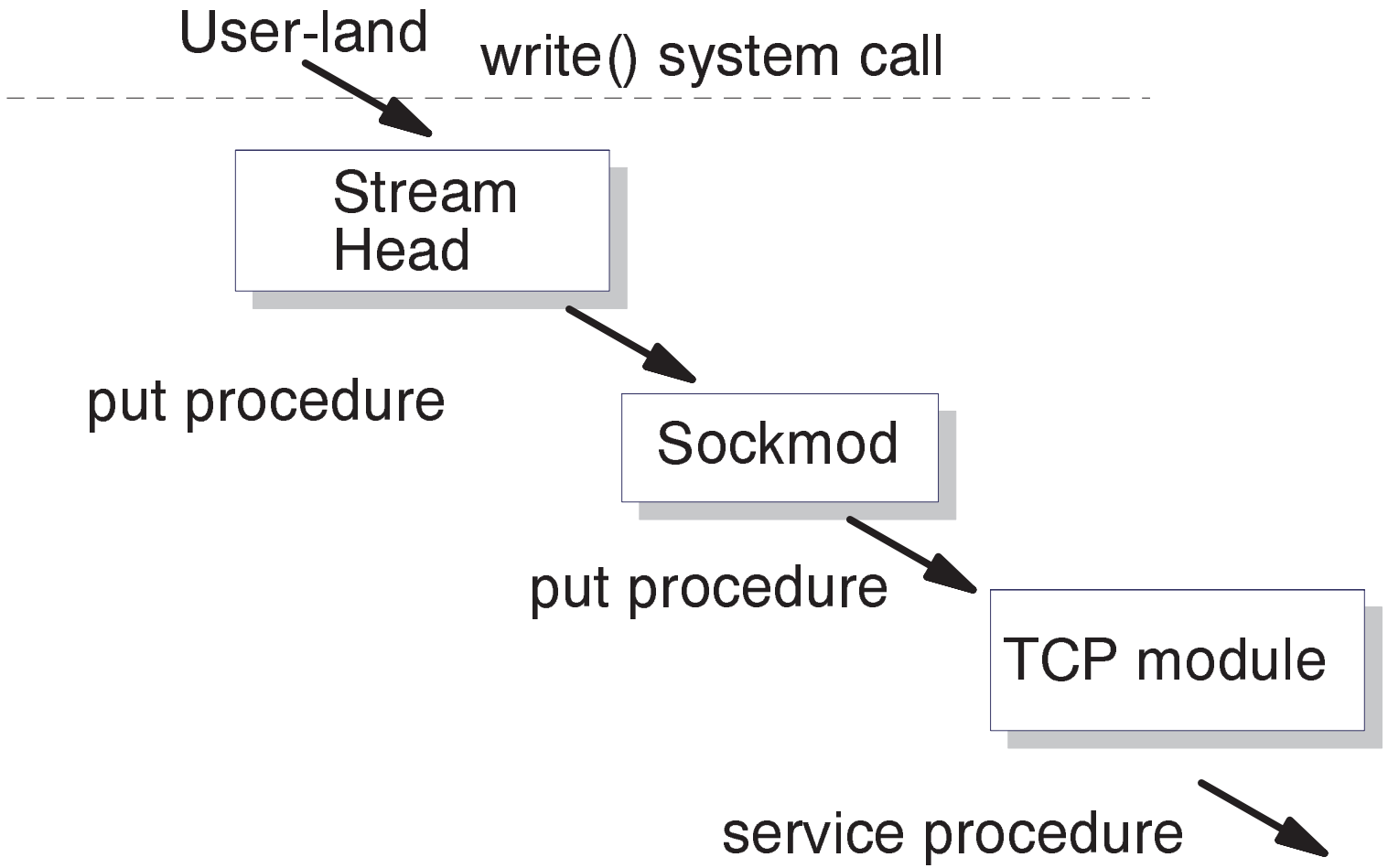
put procedures immediately call the next put procedure

A service procedure can be scheduled by the kernel

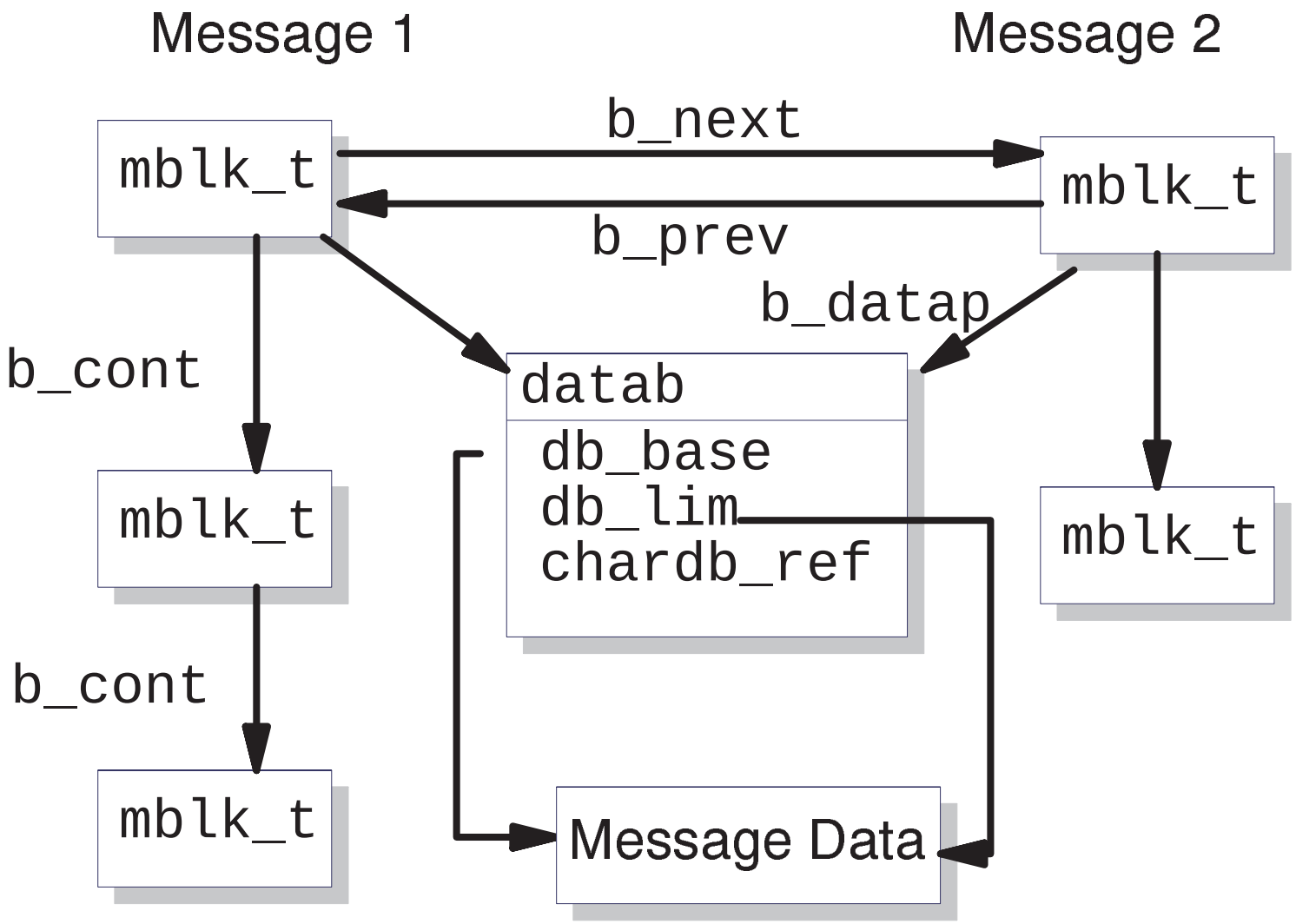
STREAM scheduling

- Modules call put procedure in context of same thread
- Call to service proc. links message onto Q and returns, later scheduler runs service procedure
  - Increases Overhead and Latency
  - Why bother? increases in gap?
- Some nasty race conditions possible
  - e.g. in read service procedure, get interrupt

STREAMS scheduling

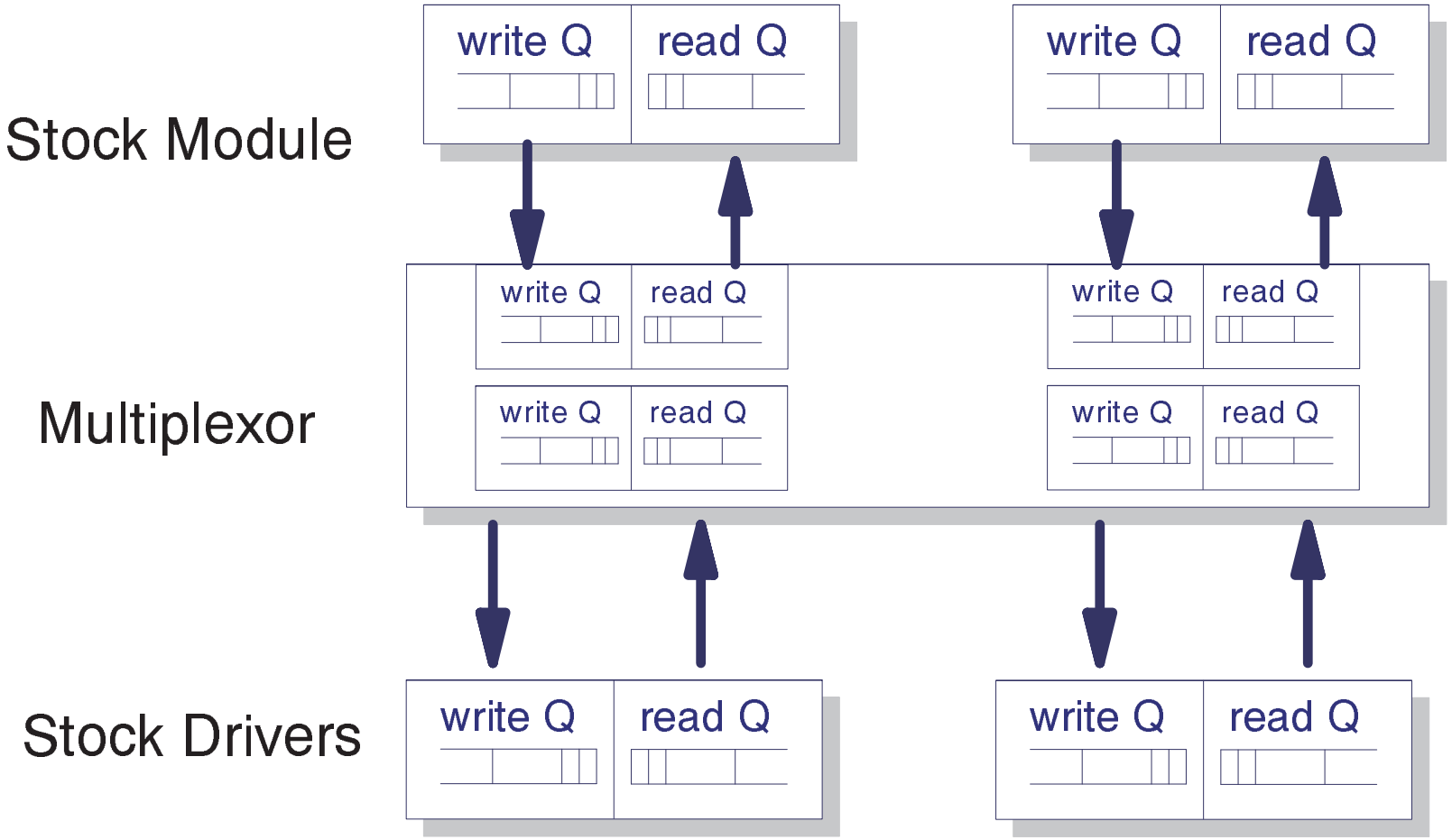


Streams Messages



Streams linkage

What if not a 1-to1 mapping?

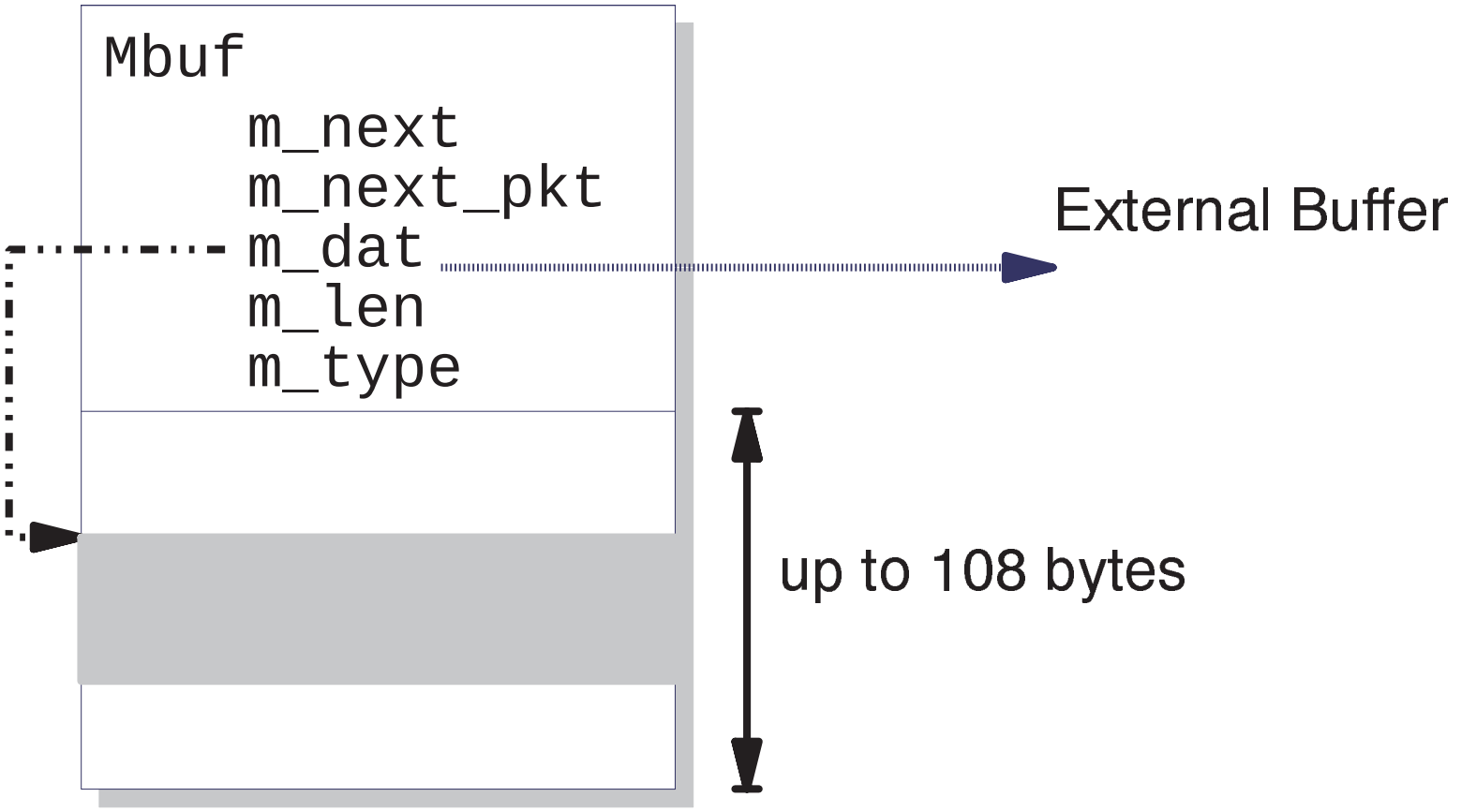


*BSD Framework

	Control Functions	Data Structures
Socket Layer	socket() bind() listen() soisconnected()	filedesc{} file{}[] file{} socket{}
Network Layers	tcp_input() ip_output() arpresolve()	protosw{} tcpcb{} inpcb{}
Device Layer	if_attach() if_conf() ether_output()	ifnet{} ifaddr{} le_softc{}

*BSD Mbufs

Simpler data structure than Streams bufs



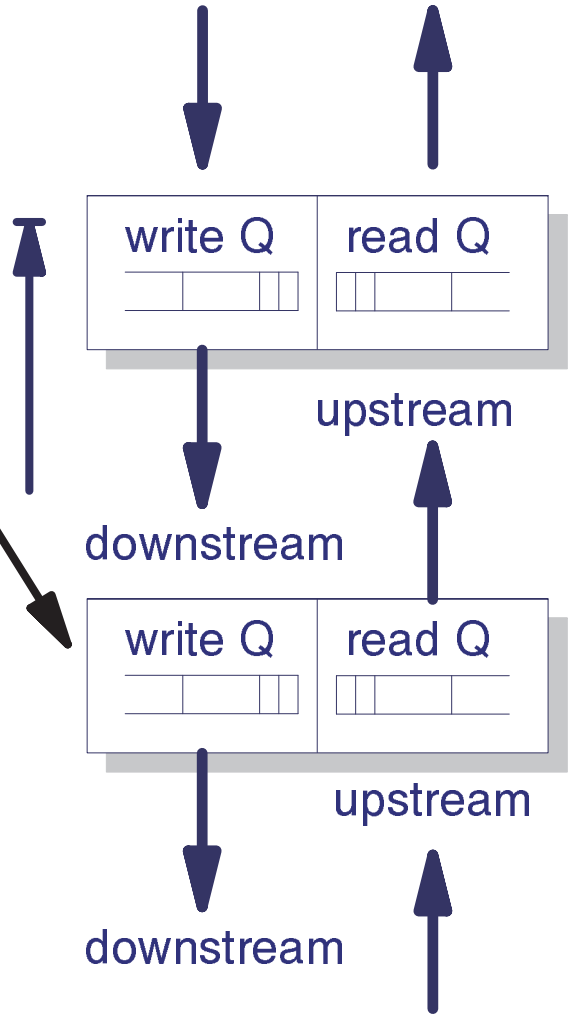
STREAMS issues

- Flow Control
  - How to stop senders when net is overloaded?
- Parallelism
  - How to get parallelism out of the communication system?
- Access method
  - How is data transfered between user/kernel boundary

Streams Flow Control

When a queue fills up here

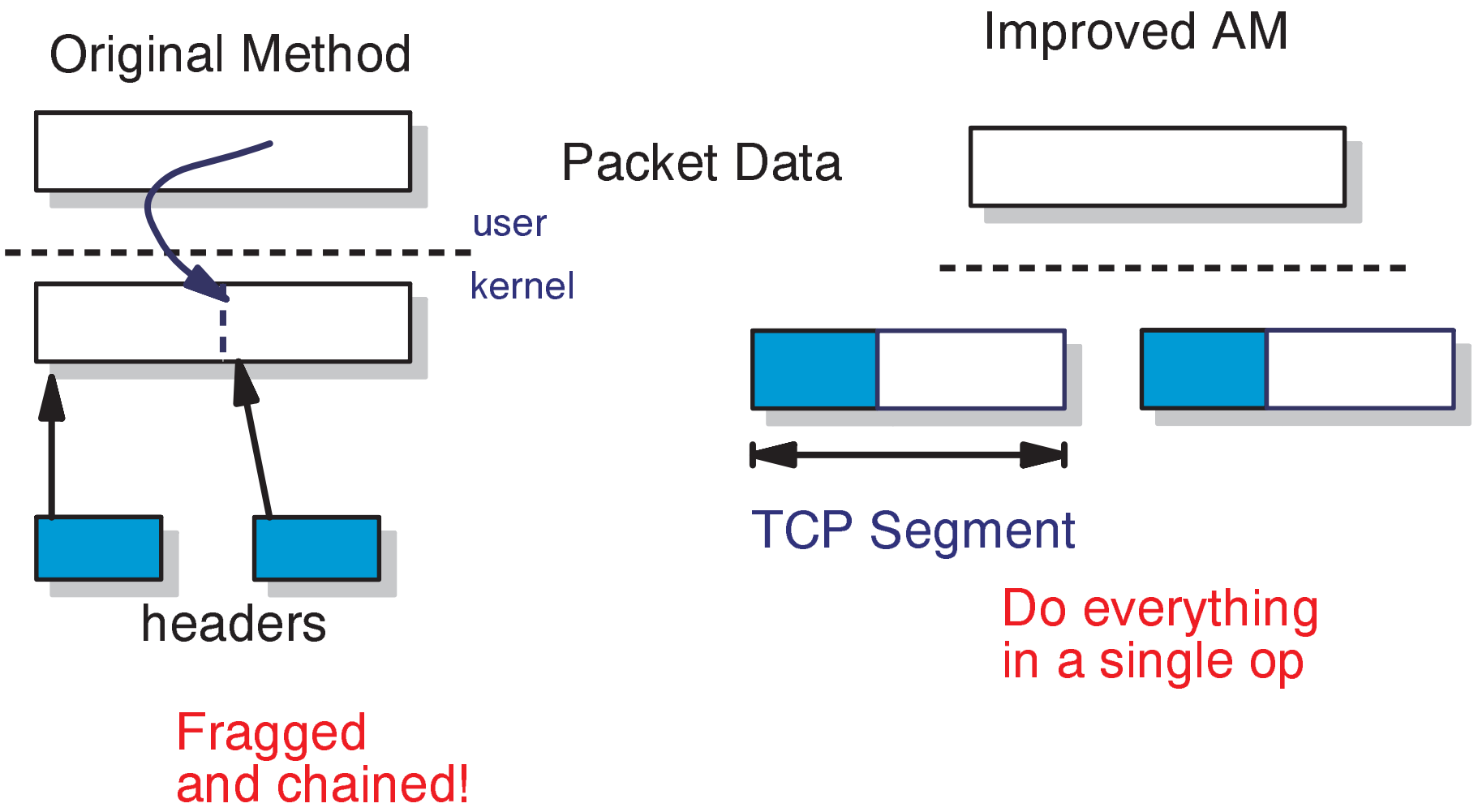
How long does it take the upstream mods to figure out to stop sending data?



Demultiplexed Arch.

- Single anchorage driver at bottom of stack
- Communication Channels anchored to bottom driver
  - CC's are a new abstraction in the system
    - like scout paths
  - Can reason about CC's
  - Early demux in the system prevents overlapping work
  - Try to maintain streams flexibility
  - Stream notion maps to CC cleanly
    - Unlike BSD

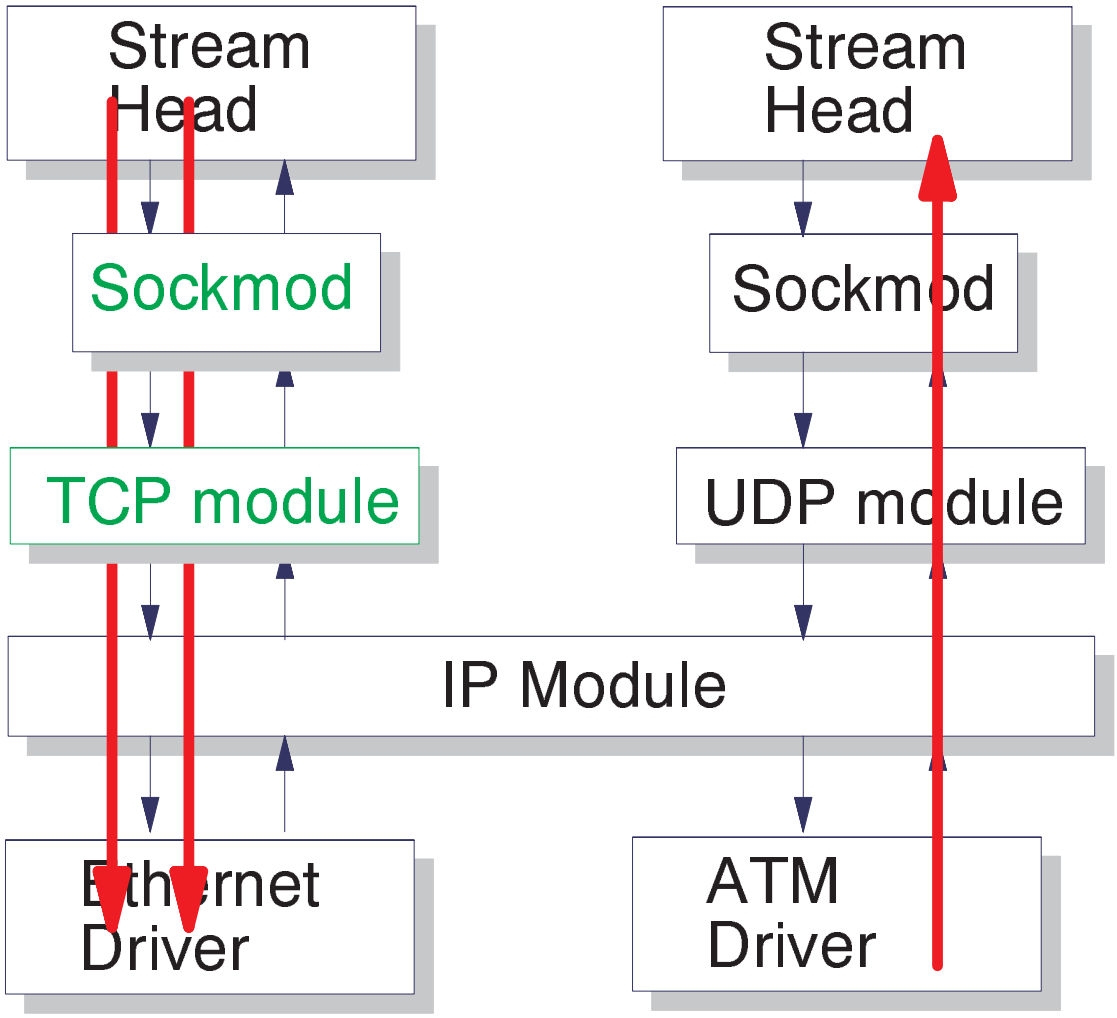
Access Method



Parallelism

Per-context
e..g across put()
= horizontal

Per-module/Q
e..g between put()
= vertical



Performance- Big G

- G 5 Mb/s
  - For both streams and BSD stacks
  - Terrible? (42 Mhz power-1?)

- A Region where BSD is worse than streams
  - Cluster mbuf effect?

Performance- little g

Recall g: messages per second

- 500 Kb/sec with 256 byte packets?
- 512 μsec / packet

Is it g or o we're observing?

- Given that not even close to the wire speed

Notice jump in BSD stack crossing 1KB boundary

- Cluster mbuf effect?

Performance- overhead

Can we also compute o given the RTT?

- 256 byte packet RTT ≅ 1.1 msec
- Recall: RTT = 4o+2L
  - Wire Time: 0 (loopback interface)
  - o ≅ 275 μsec

What could explain the discrepancy?

- Is there a discrepancy?
- What does g mean in the steady-state?
- Who's bandwidth are we counting?

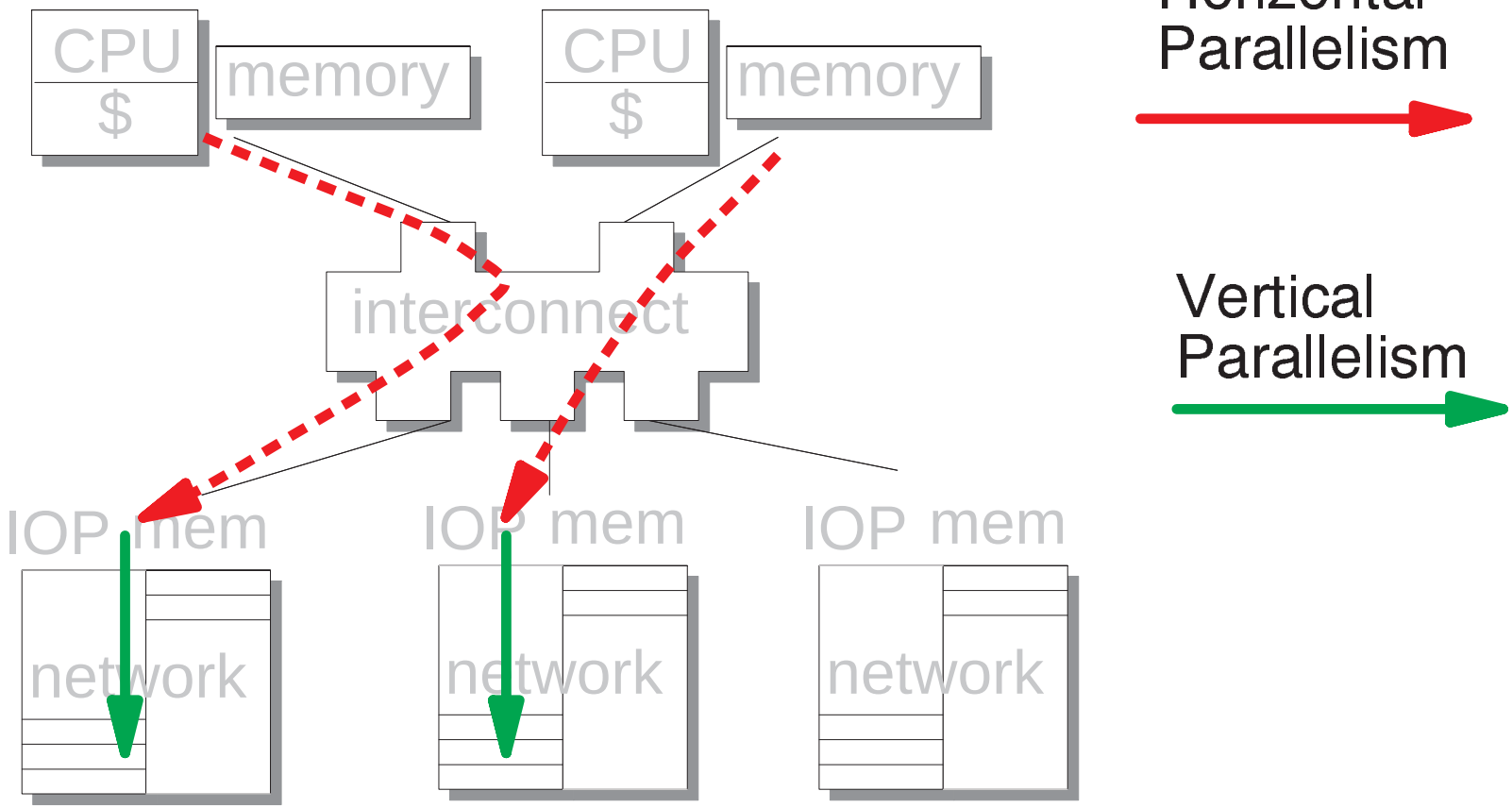
Performance: CPI

Estimate CPI from data in the paper

- Use iron triangle
  - 8517 instructions
  - RTT 1.2 ms; one way ≈ 0.6 μs. Time ≈ 600 μs
  - 42 Mhz CPU = 42 cycles/ 1 μs

$$600\text{us} \times \left(\frac{42\text{cycles}}{1\text{us}} \right) \times \left(\frac{1}{8517\text{instructions}} \right) \approx 2.9 \frac{\text{cycles}}{\text{instruction}}$$

Mapping to Fed. Architecture?



Vnode architecture

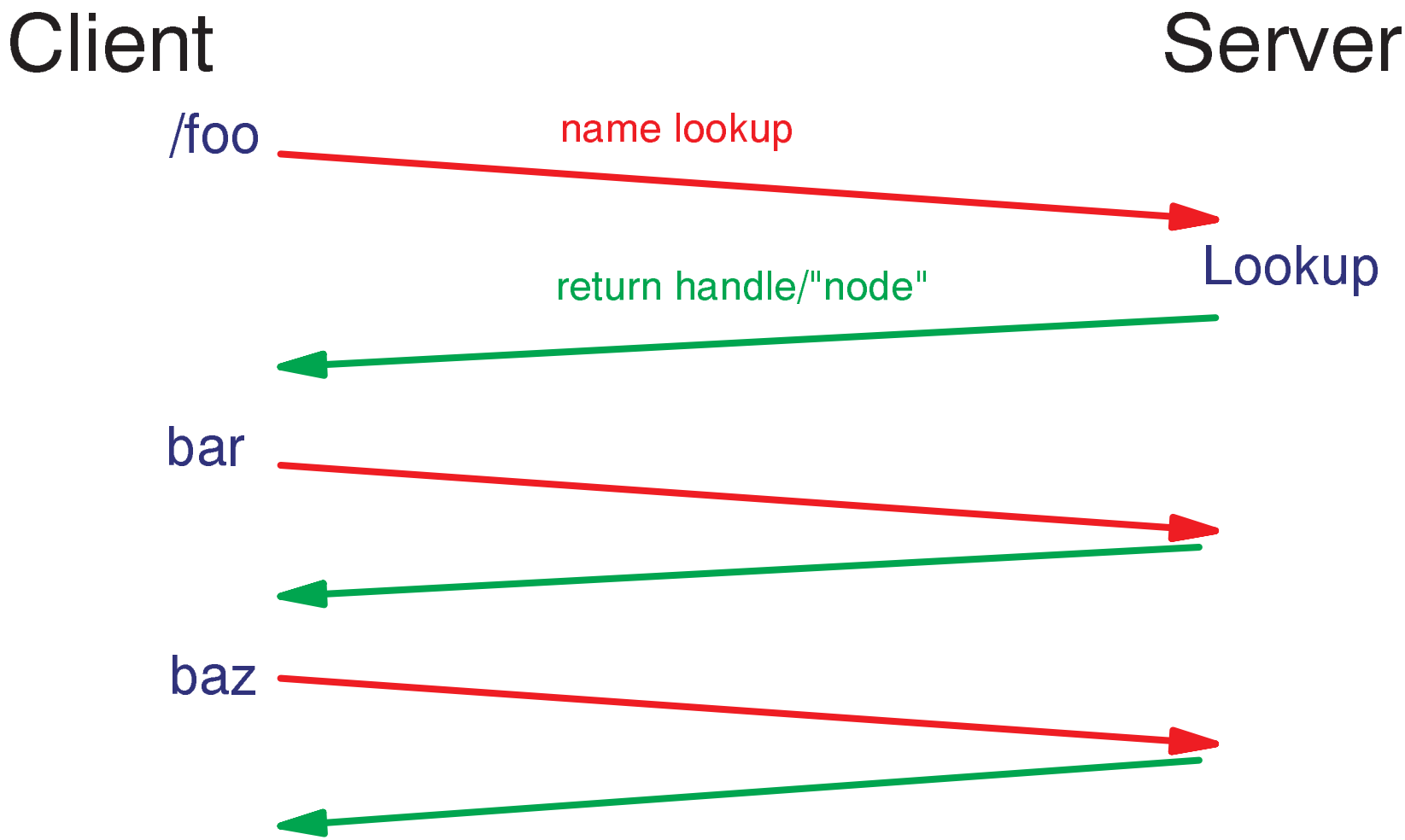
- What if the kernel doesn't have a generic FS structure?
  - How to access NTFS, VFAT, UFS, etc?

Vnode layer is an abstraction of a generic file system object
  - Compare to inodes, which define the object

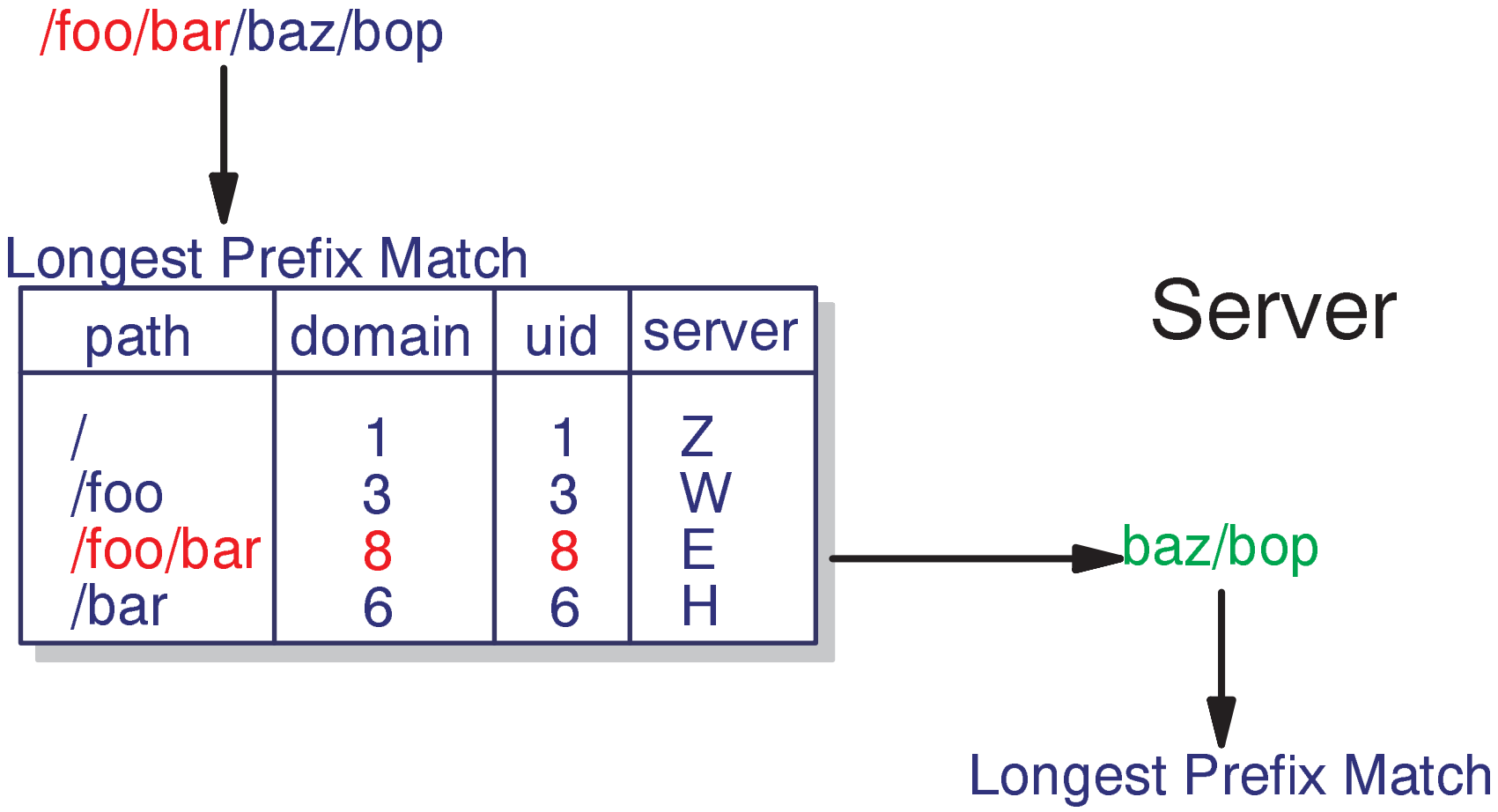
Real file system has direct mapping of vnodes to internal nodes
  - inodes in UFS
  - handles for NFS

Classic FS name mapping

Get a handle to /foo/bar/baz on the right server



Prefix Table Mapping

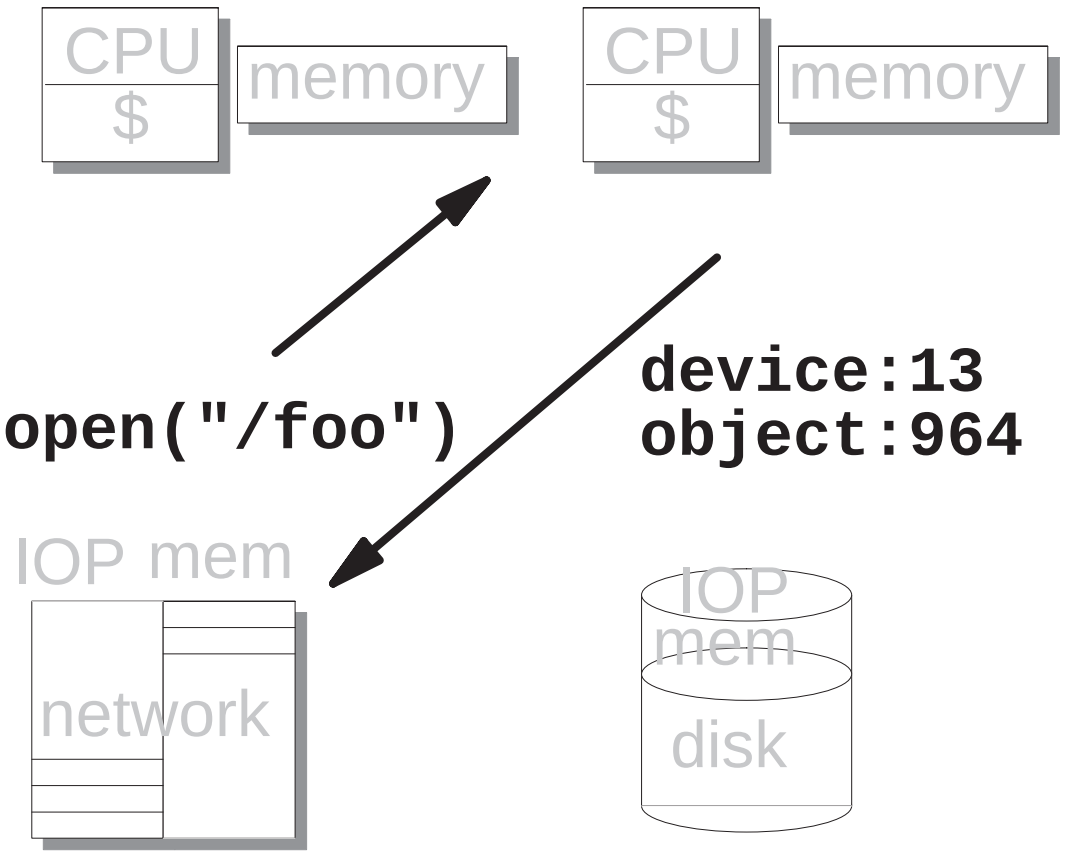


Unified vs Unique Namespace

For what is a namespace valid?

- Per process
  - Pro's: Create abstraction per user!
  - Con's: management? Conventions?
- Per machine
  - Pro's: Backwards compatible
  - Con's:
    - Number of FS's total (1000?)
    - Management?
- Unified global
  - Pro's: easy to manage
  - Cons: device naming and semantics

Federated Arch Nameserver?

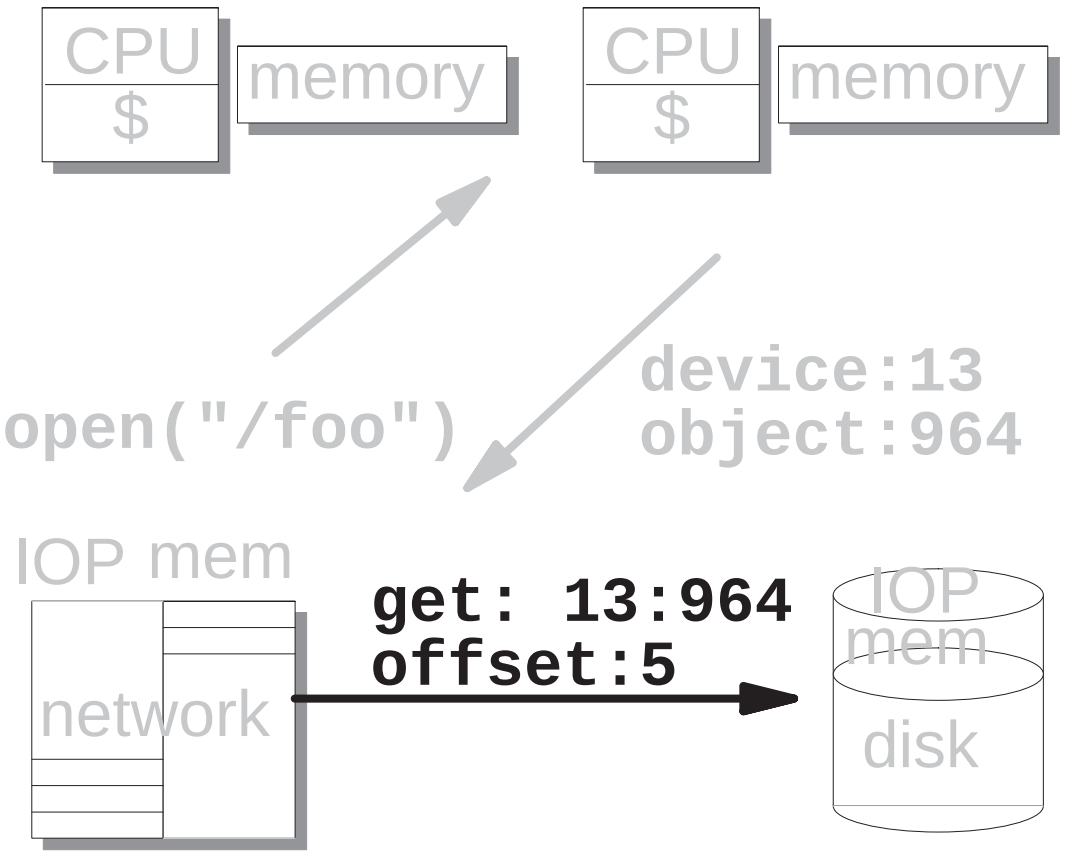


Develop a global namespace for the whole machine

Devices can access any name in the FS

Nameserver maps names to [device:object]

Sprite-Style I/O



Devices only contain abstract objects

Name of object controlled by a higher level

Easy to add semantics (different name in name space)

Stateful vs. Stateless

Why is NFS stateless?

- ND (Network Disk) was Sun's previous method of sharing disks
  - Connection state made recovery a nightmare
  - What if server crashes in middle of an update?
  - Think of surviving an open TCP connection
- Solution: shove all recoverly logic onto client
  - Current logic: Hang process, timeout
  - Similar to Swarm methodology

Crash Recovery

What happens on a client or server crash?

- Per-client list on each server
  - Booting, alive and dead state
  - Version info on each RPC
- Object version information
- Reopen protocol
  - What to do when you get back stale info?
- Idempotent recovery protocol
  - What if crash in the middle of recovery Try again!
- Crash detection
  - boot-stamped RPC
  - pingers

Federated Arch stateful?

What happens if these crash:

Main OS?

- Reset whole system?
- Part of the system?

Network

- Connections?
- Routing tables?
- Interface tables?
- Socket?

Disk

- Blocks?
- Disk cache?
- Objects?
- Extents?
- atomic transactions?

Where to partition?

- μ-kernel is nodeID, messages only
- Use a higher level of namespace?
- What is accessable on the remote devices?
  - Regular files
  - Special files
  - Devices
  - Process/ports only?

Next Week

- Classic paper on why machines fail
- Six-Sigma paper: How HW designers have approached reliability
- Study of actual machine failures