High-Speed I/O: The OS As A Signalling Mechanism

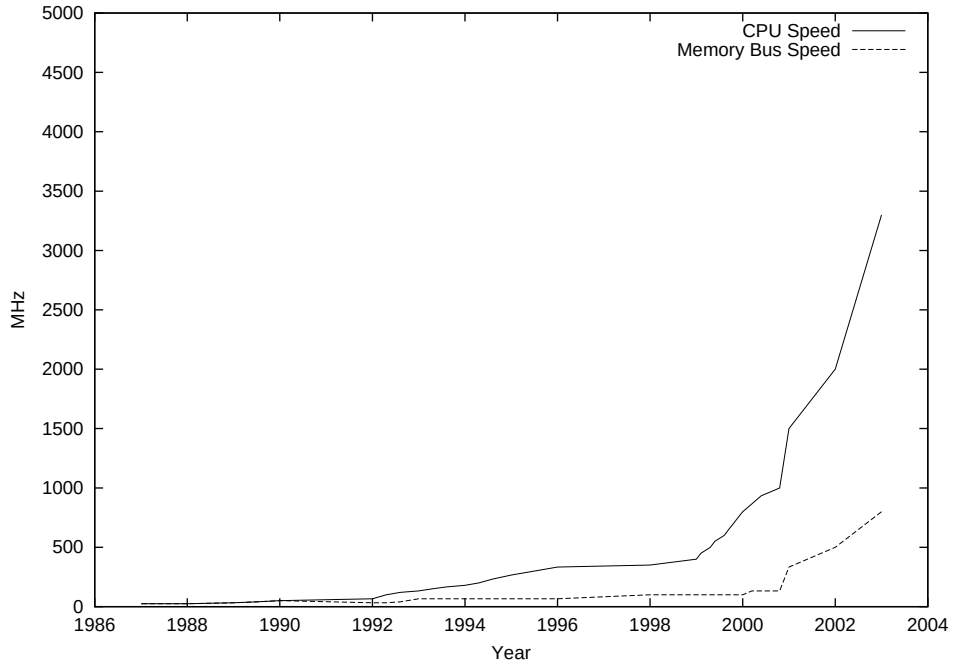
Angelos Keromytis
Columbia University
angelos@cs.columbia.edu

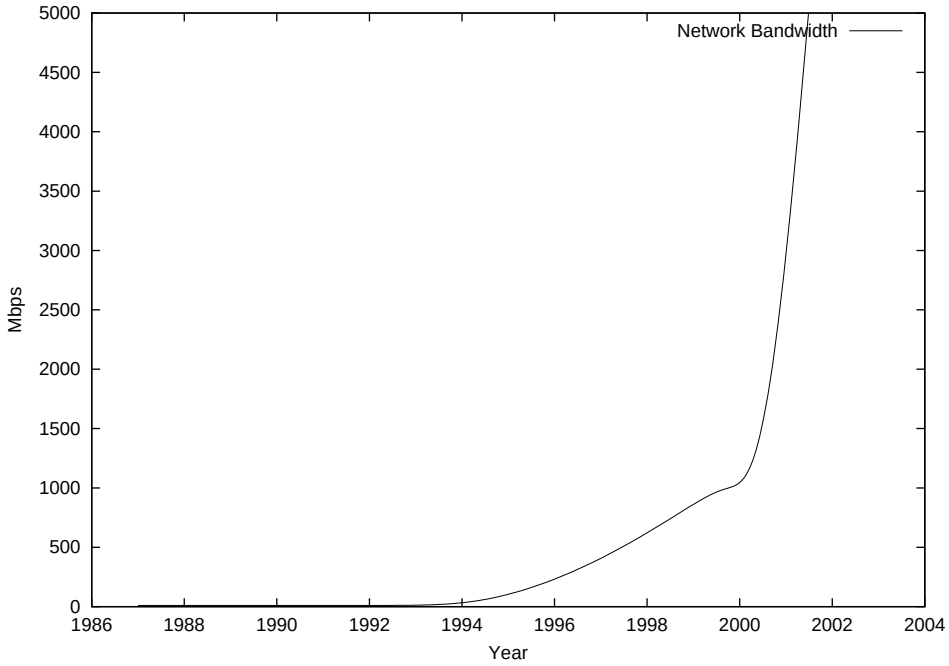
Summary

- New I/O technologies designed to cope with higher network speeds
  - Interconnects, 10GigE, etc.
  - Daisy-chain busses (FireWire)
- Operating system design has not evolved to accomodate this
  - Remains in the critical (data) path
- Proposal: structure a system as a data switch
  - Application in the policy plane
  - OS in the control plane
  - Direct data flows across peripherals

Problem

- As usual: the memory
  - Memory bandwidth vs. I/O devices (HDD, NIC, ...)



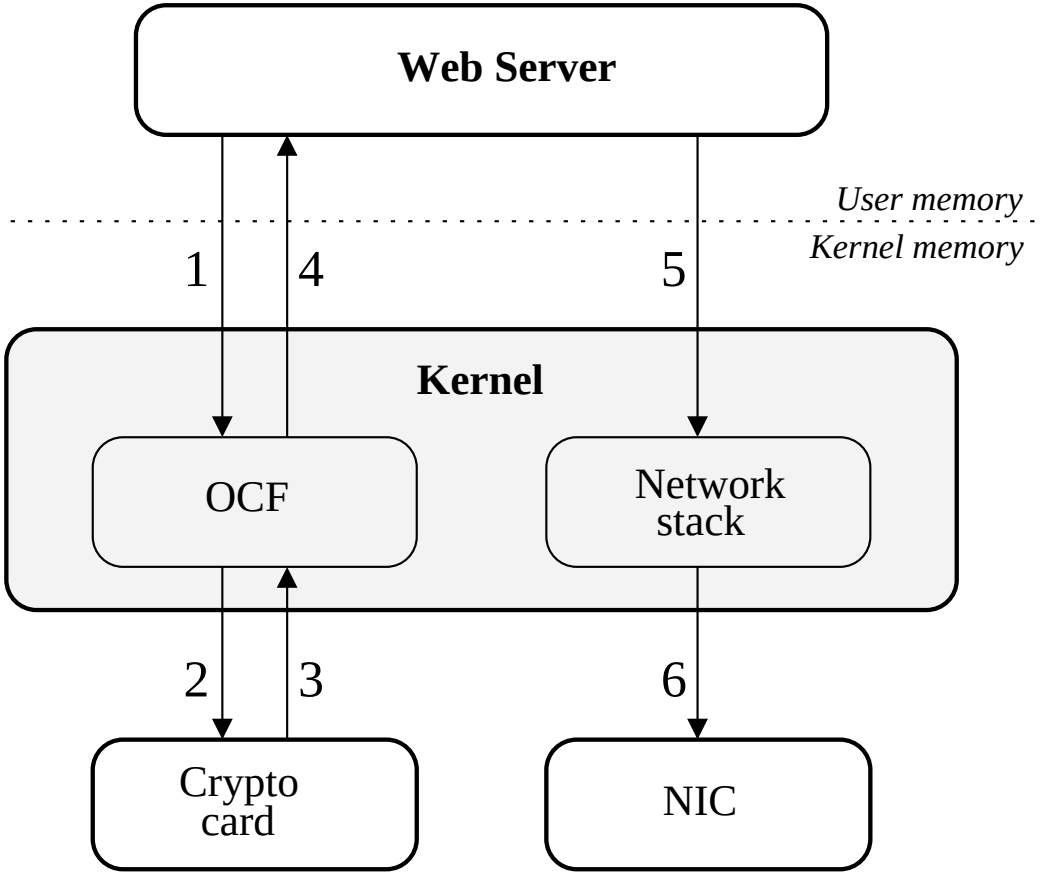


Other Trends

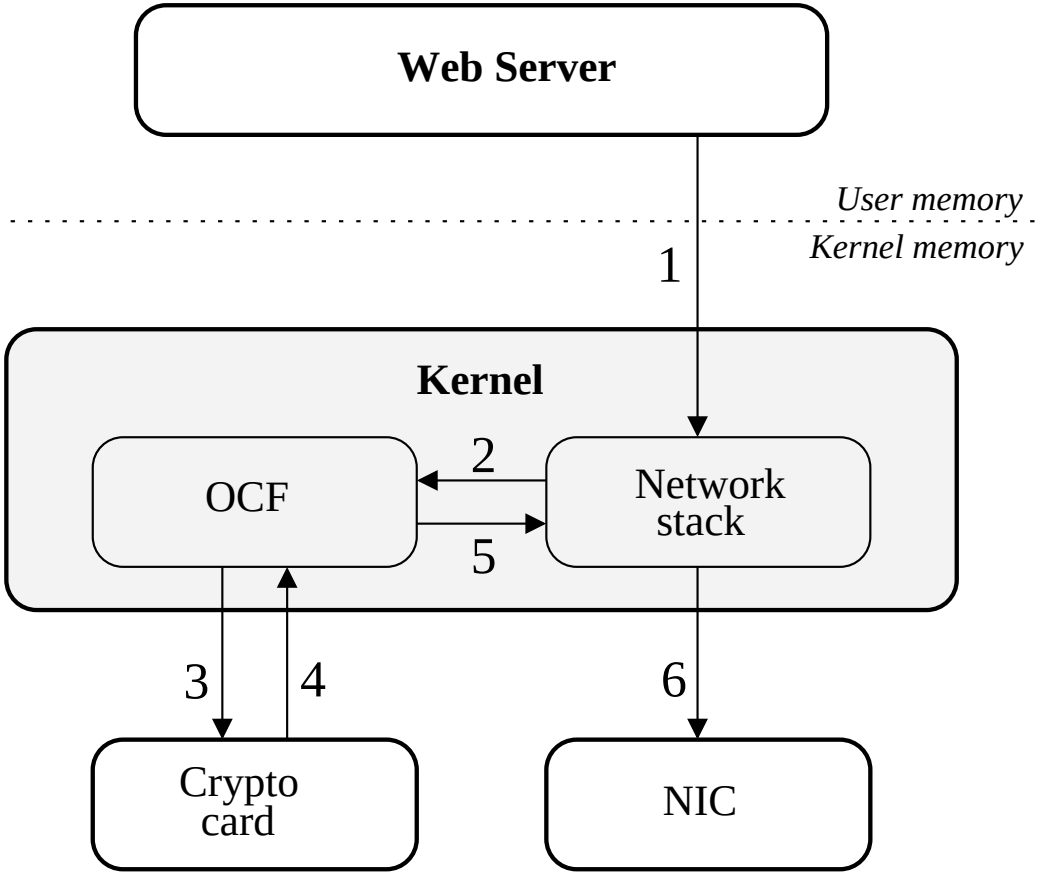
- Multiple transactions per "request"
  - MPEG encoding, DSP processing, crypto, ...
- Specialized hardware
  - e.g., protocol offloading
  - Integrated into devices
- Programmable hardware
  - Stand-alone (FPGAs)
  - NICs, HDDs (?)
- Daisy-chain buses

- Observation: operating systems don’t take advantage of these

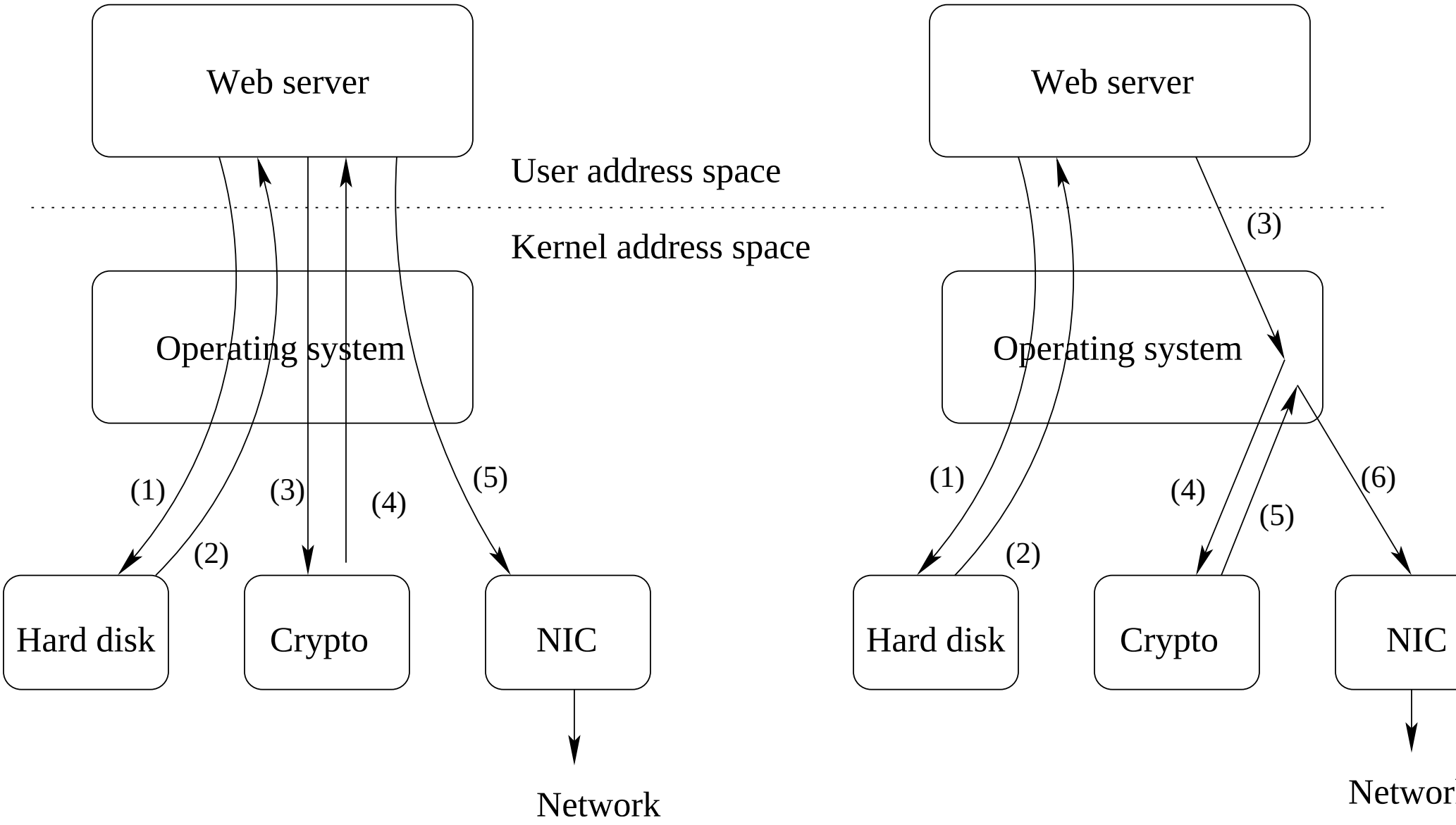
Background: Fast Web Crypto



Approach: Cut Down On Copies

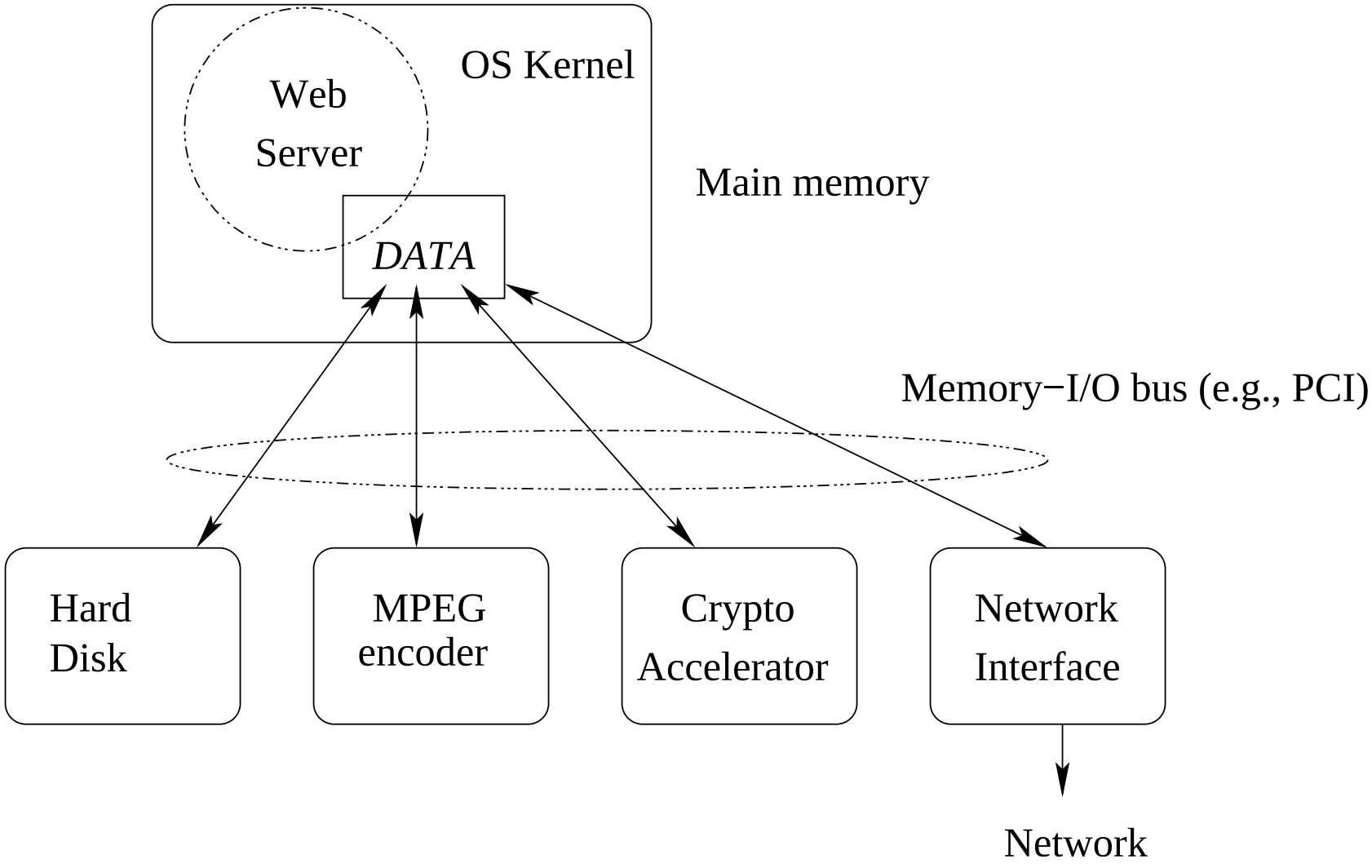


Current Operating Systems



System As Network

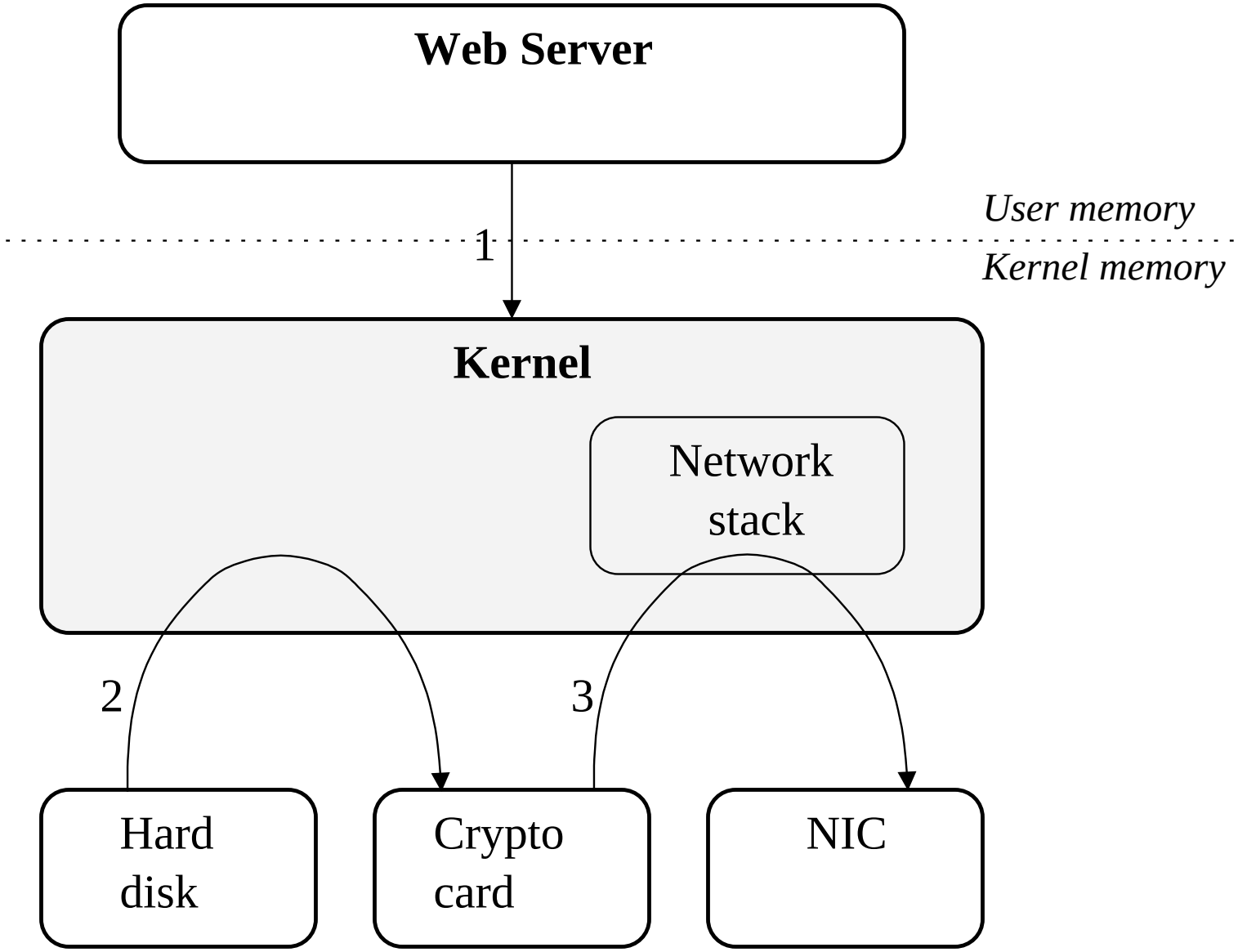
- Memory in the center of a "star" topology



Solution

- Treat system as a network switch
  - Data flows between peripherals
  - Sources, destinations, and data-transform devices
- New role for applications
  - Admission control, data transfer policy
- New role for OS
  - Control plane, exception handling, HW composability
  - Dynamically program hardware (FPGAs)
    - Use current stack (DCG) ?

Data Flows


Web Server
User memory
1
Kernel memory
Kernel
Network stack
2
3
Hard disk
Crypto card
NIC

Advantages

- Removes memory from critical path
  - Make the interconnect the limitation again
    - We know how to make that faster!
- Naturally takes advantage of special-purpose hardware
  - "Create" special-purpose hardware (using FPGAs)
- Direct device transfers on daisy-chained buses
  - Need for FPGA USB/FireWire "blobs" ?

Challenges

- Current generation of devices makes card-to-card transfers tricky
  - Possible with FPGAs, increases utilization of interconnect
- Other limitations in PC architecture (IRQs)
- Composition
- Data flow scheduling
- Exception handling

- Half-baked idea: simple language for managing flows
  - Specified by the application
  - Executed/interpreted by the kernel

Conclusions

- New architecture for fast server I/O
  - Eliminate memory as the "whiteboard"
  - Direct data transfers between devices
  - Leverage programmability to eliminate OS dependency
    - Natural programmability
    - Introduced via FPGAs
  - Take advantage of daisy-chain or proprietary buses
- Future directions: prototype, evaluation