Intro: What is a System?

COS 316: Principles of Computer System Design

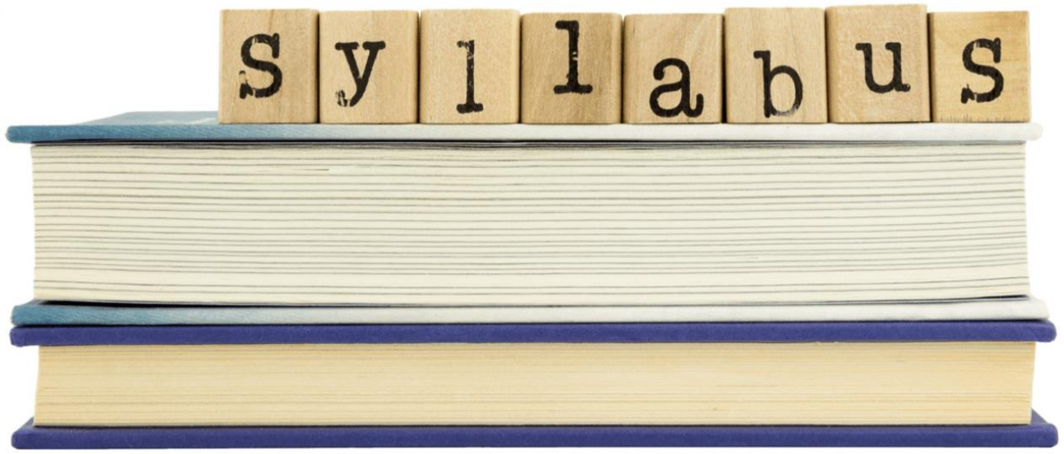
https://cos316.princeton.edu/

Lecture 1

Amit Levy & Ravi Netravali

- Today: Systems!

- Next time:



Example Systems

- Operating system (OS) kernel
- The Internet
- Database
- Distributed file system
- Web framework
- Game engine

You make it fun,
we make it run!

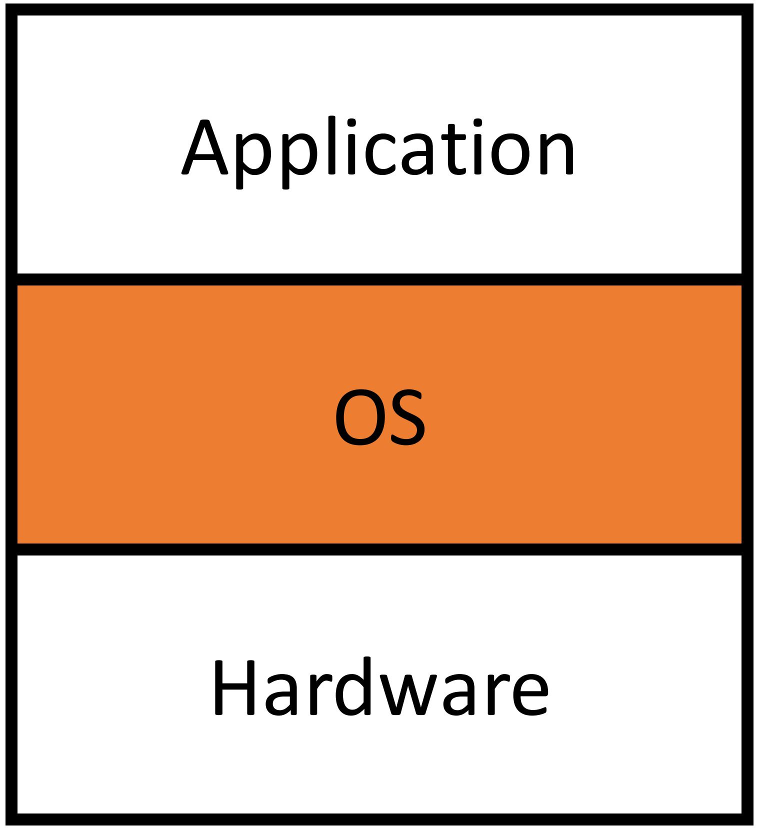
What is a System?

- Provides an interface to underlying resources
- Mediates access to shared resources
- Isolates applications
- Abstracts complexity
- Abstracts differences in implementation

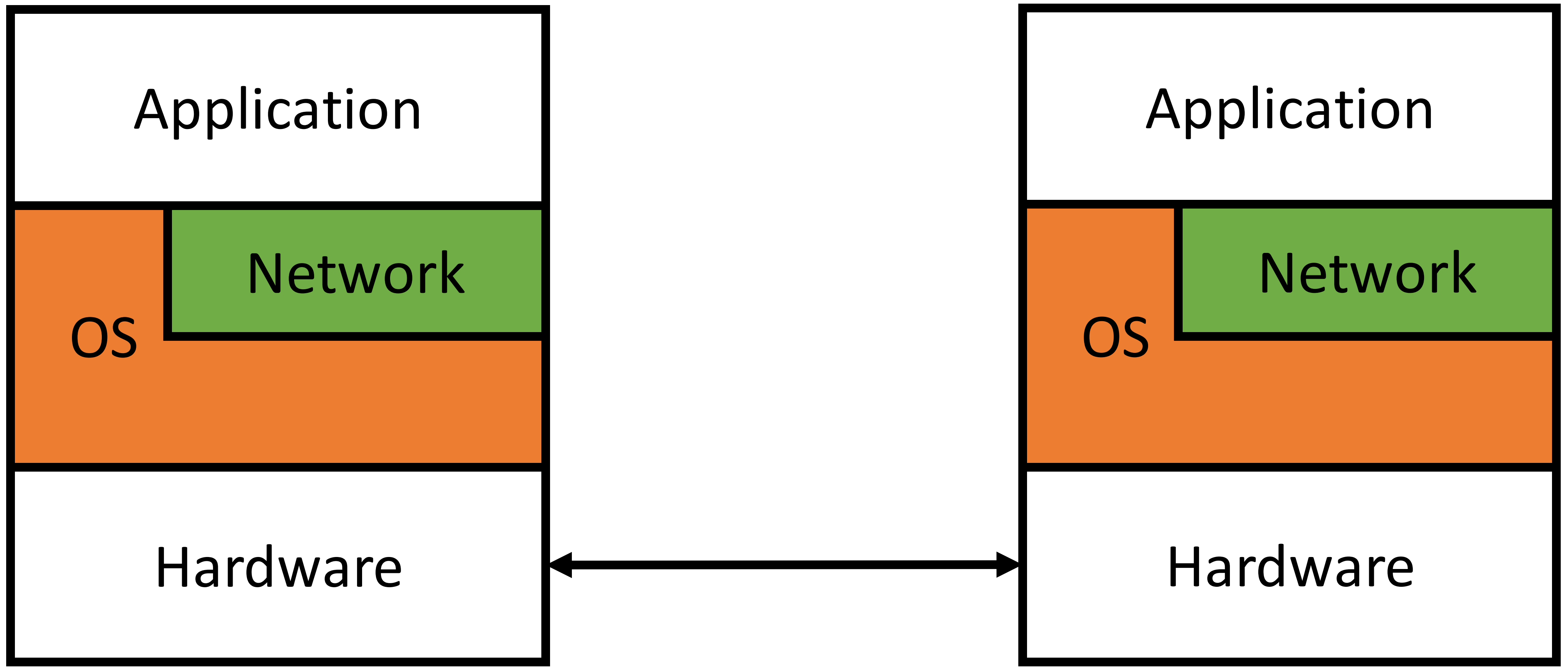
Example System: OS Kernel

- Interface: system calls
- Underlying resources: hardware (CPU, memory, network, disk)
- Isolation: Firefox, terminal, zoom, … don’t worry about each other
- Abstraction: Collection of system calls
  - Instead of specific protocols for using specific devices
  - Don’t need to rewrite Firefox to display on new monitors, or save to new disks, or …

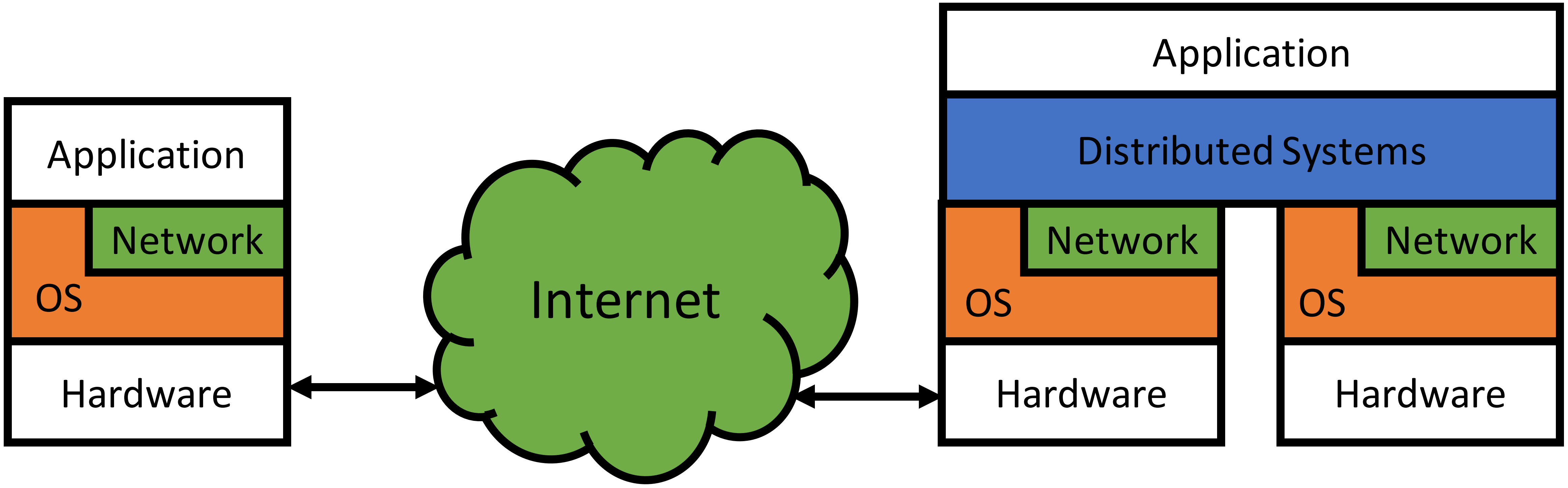
Systems Stack (terminal)



Systems Stack (Firefox)

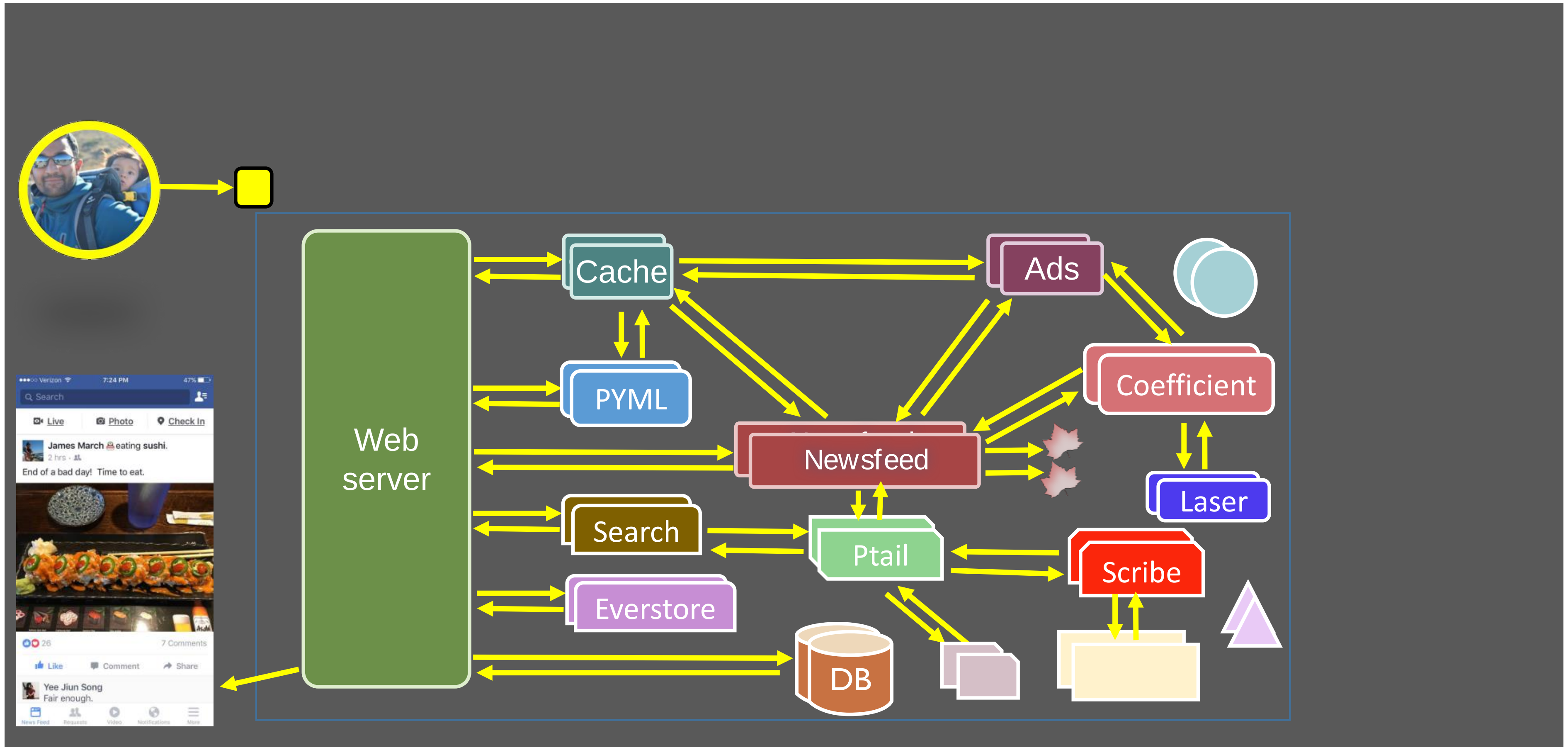


Systems Stack (Firefox to Google)



So Many Systems…

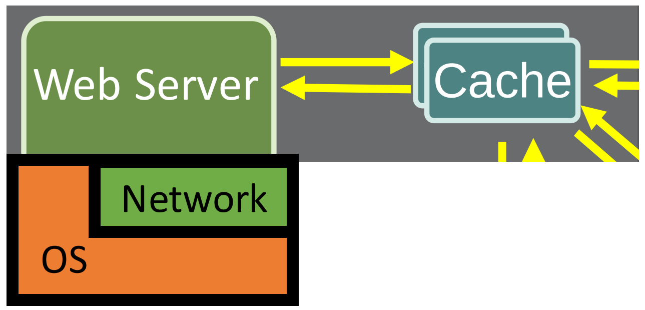
Each user request touches hundreds of systems



[Slide from Kaushik Veeraraghavan Talk’s on Kraken at OSDI 2016]

Systems Are Everywhere!

- People use applications
  - Applications are built on systems
  - On systems on systems on systems…
- If you’re building applications
  - Useful to understanding underlying systems
    - What could be causing X?
    - Why can’t they do Y?
    - What can I trust Z to do or not?
- If you’re building systems ☺
  - That’s what this is all about!
  - Useful to understanding your underlying systems



Why Are Systems Challenging? Part-1a

- Correctness
  - Incorrect system => incorrect applications
  - Correctly implement interface's guarantees
- Performance
  - Slow system => slow applications
  - Make system fast enough
- Security
  - Insecure system => insecure applications
  - Build security into the system

Why Are Systems Challenging? Part-1b

- Distributed storage system that keeps data forever (e.g., videos)

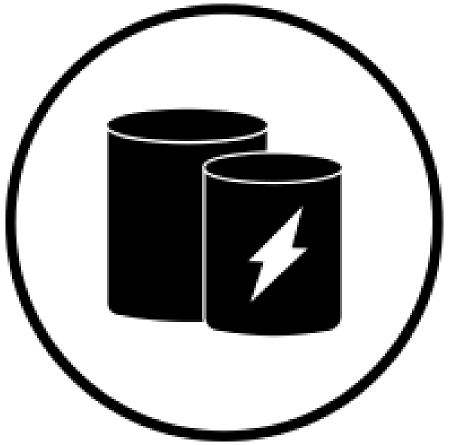
- Correctness
  - Accurately retain data forever. Really delete data on deletes.
- Performance
  - Fast and highly concurrent.
- Security
  - Only allow authorized users to retrieve data

Why Are Systems Challenging? Part-2a

- How general should an interface be?
  - More general => supports more application-level functionality
  - Less general => easier to implement, easier correctness, better performance, easier security
- How portable should an interface be?
  - More portable => supports more underlying resources
  - Less portable => …
- Design tradeoffs!

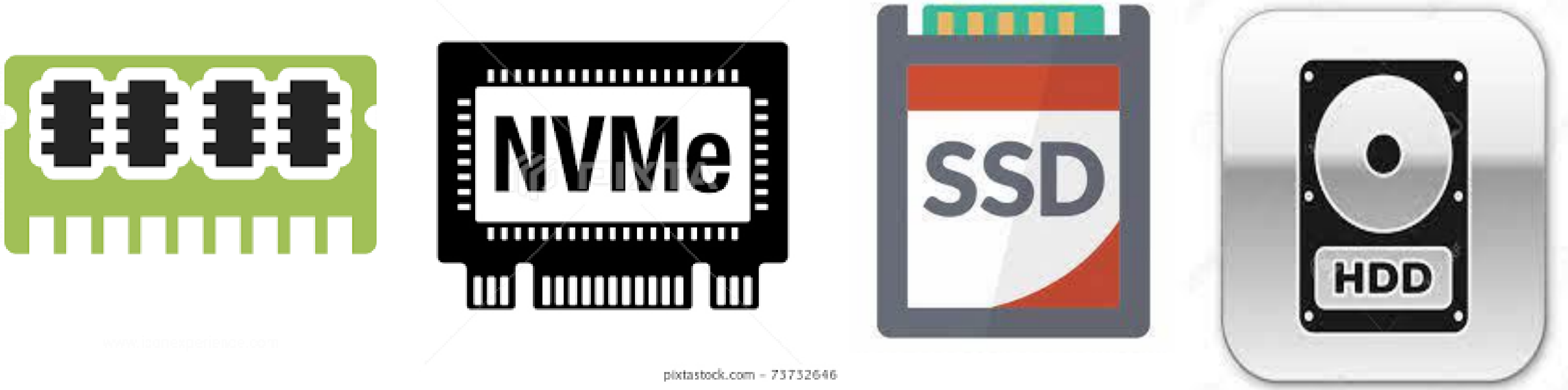
Why Are Systems Challenging? Part-2b

- Distributed cache that provides fast access to popular data
- How **general** should an interface be?
  - Read(key)
  - Write(key, value)
  - Read_transaction(<keys>)
  - Write_transaction(<keys>)
  - Read_and_write_transaction(<read_keys>, <write_keys>)
  - …
- Design tradeoffs!



Why Are Systems Challenging? Part-2c

- Distributed cache that provides fast access to popular data
- How **portable** should an interface be?

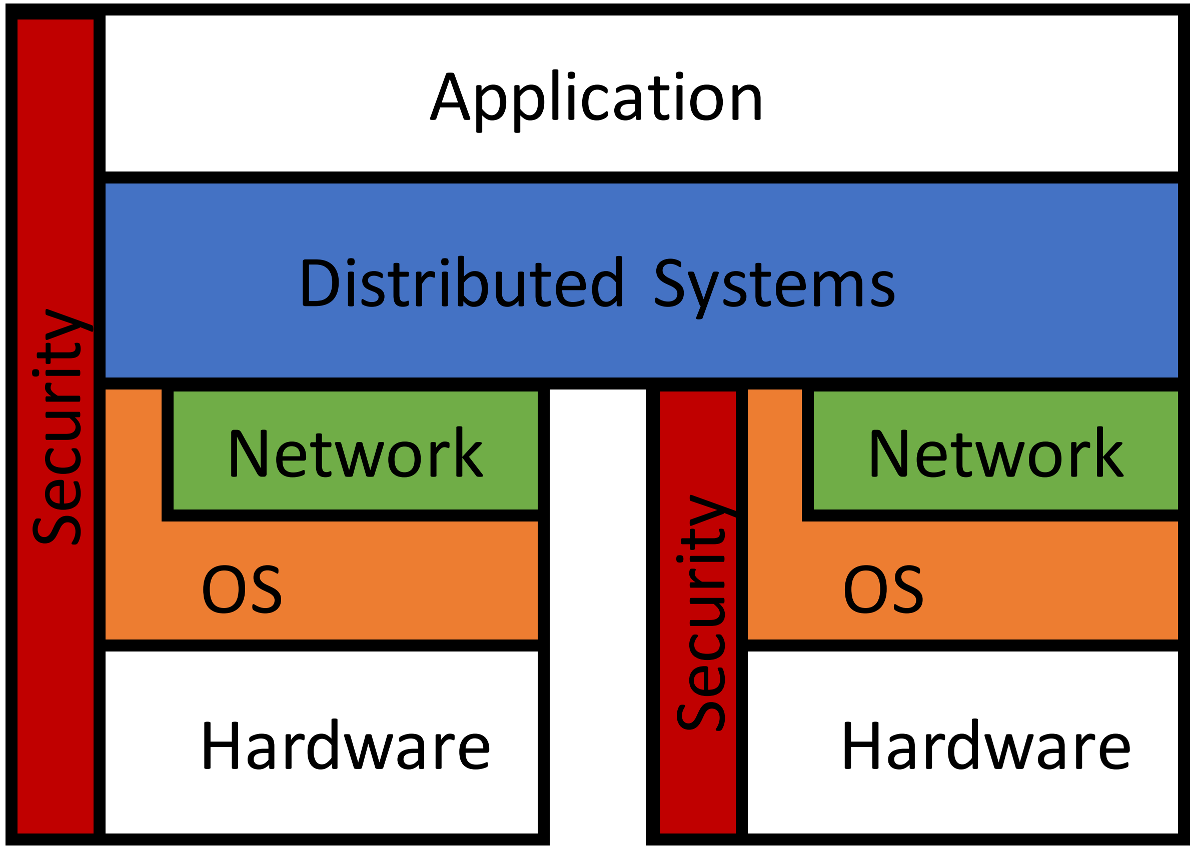


- Design tradeoffs!

General vs Portable Interfaces

- Cache A:
  - Read, Write on DRAM, SSD, NVM, HDD
- Cache B:
  - Read, Write, Read Transaction, Write Transaction on SSD
- Which cache is more general? More portable?
- PL Example: Javascript vs Assembly?

Systems We Will Cover In This Class



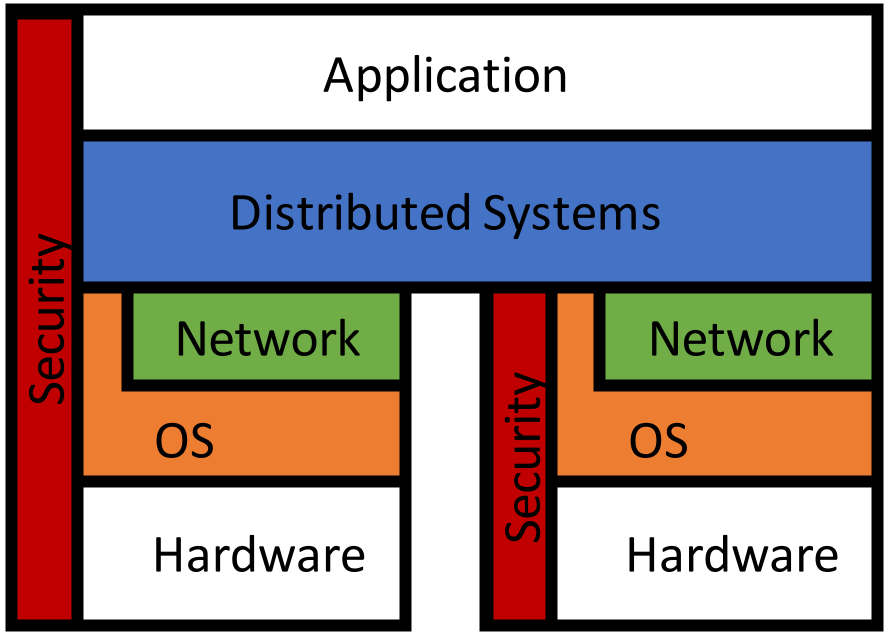
- Distributed Systems
- Networking
- Operating Systems
- Security

Why Do I Love Systems?!

- I don't need to be that smart:
  - Systems are about getting the design right. After that it's easy.
- Art of reasoning about tradeoffs: e.g., Safety vs. Performance
- Try to anticipate what applications that don't yet exist might need
- Multiplicative impact: improving systems improves all apps built on them

Summary

- Systems abstract underlying resources
- Systems are **everywhere**
- Systems are challenging and interesting and cool
- This class is about systems: details next lecture



Mini-Logistics: More in the Next Lecture

- Lectures
  - In person
  - Slides available online
- Precepts on Thursdays and Fridays
  - Bring your laptop to precept!
- Install software
  - Create your GitHub account
  - Go language
  - Editor (e.g., Emacs, Vim, or VS Code)
- Ed discussions