Intro: What is a System?

COS 316: Principles of Computer System Design

Lecture 1

Wyatt Lloyd

- Today: Systems!

- Next time: Course Overview, Syllabus, …

Example Systems

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

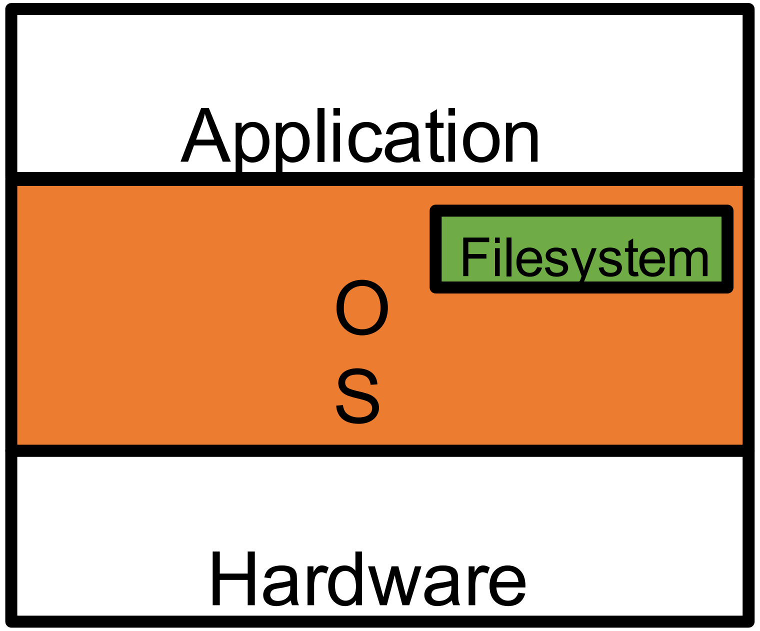
What is a System?

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

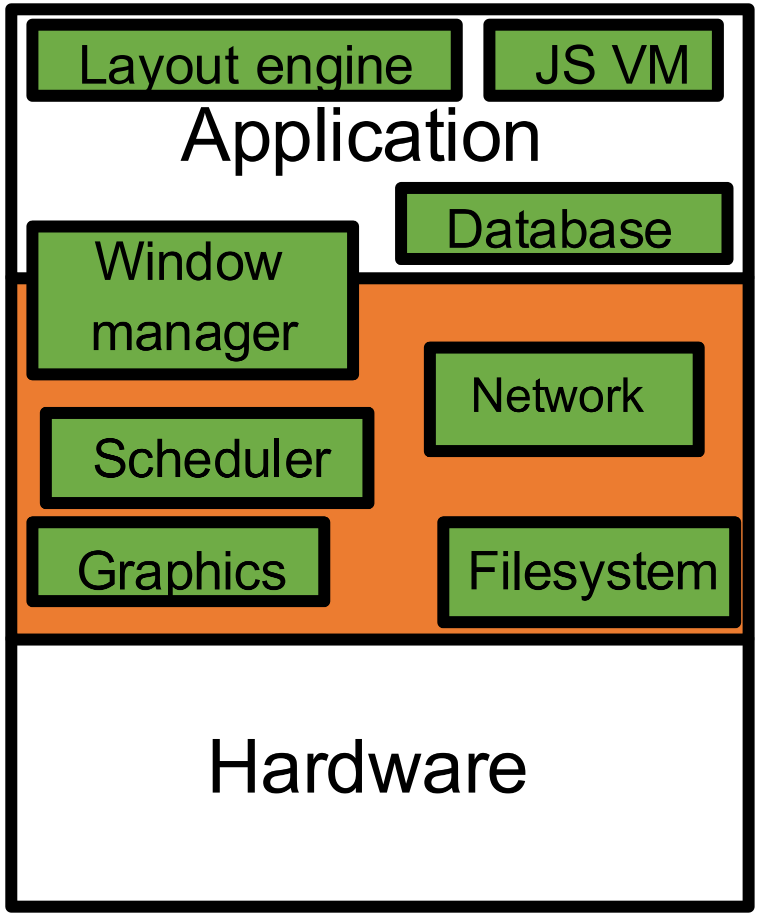
Example System: OS Kernel

- 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 …

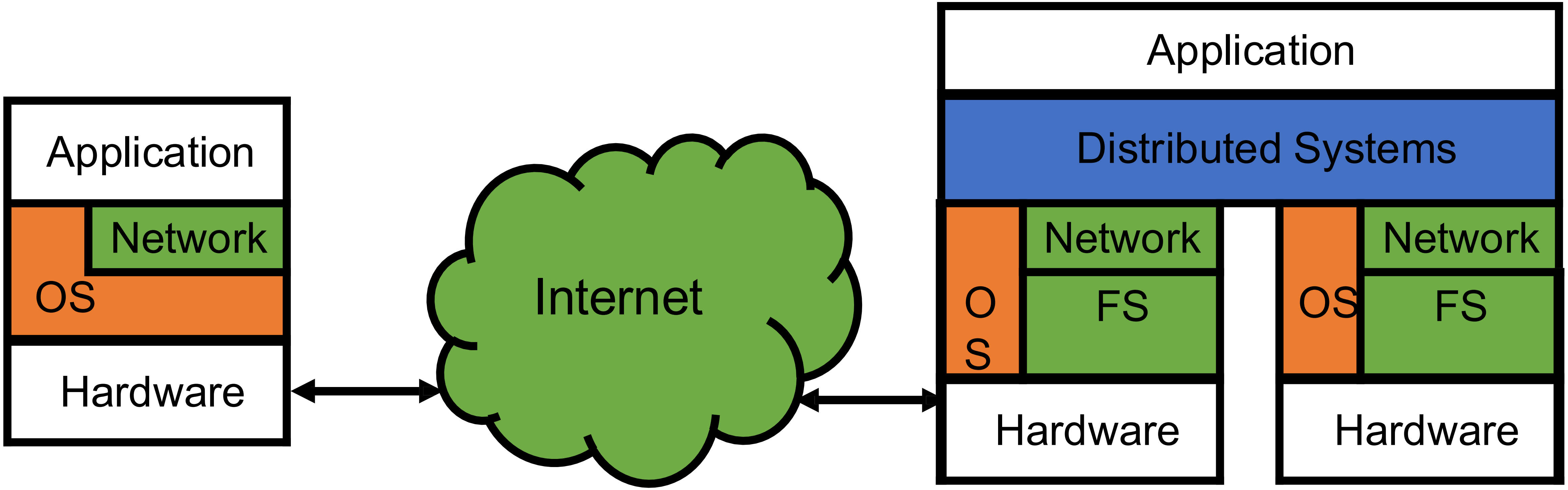
Systems Stack (terminal)



Systems Stack (Firefox)

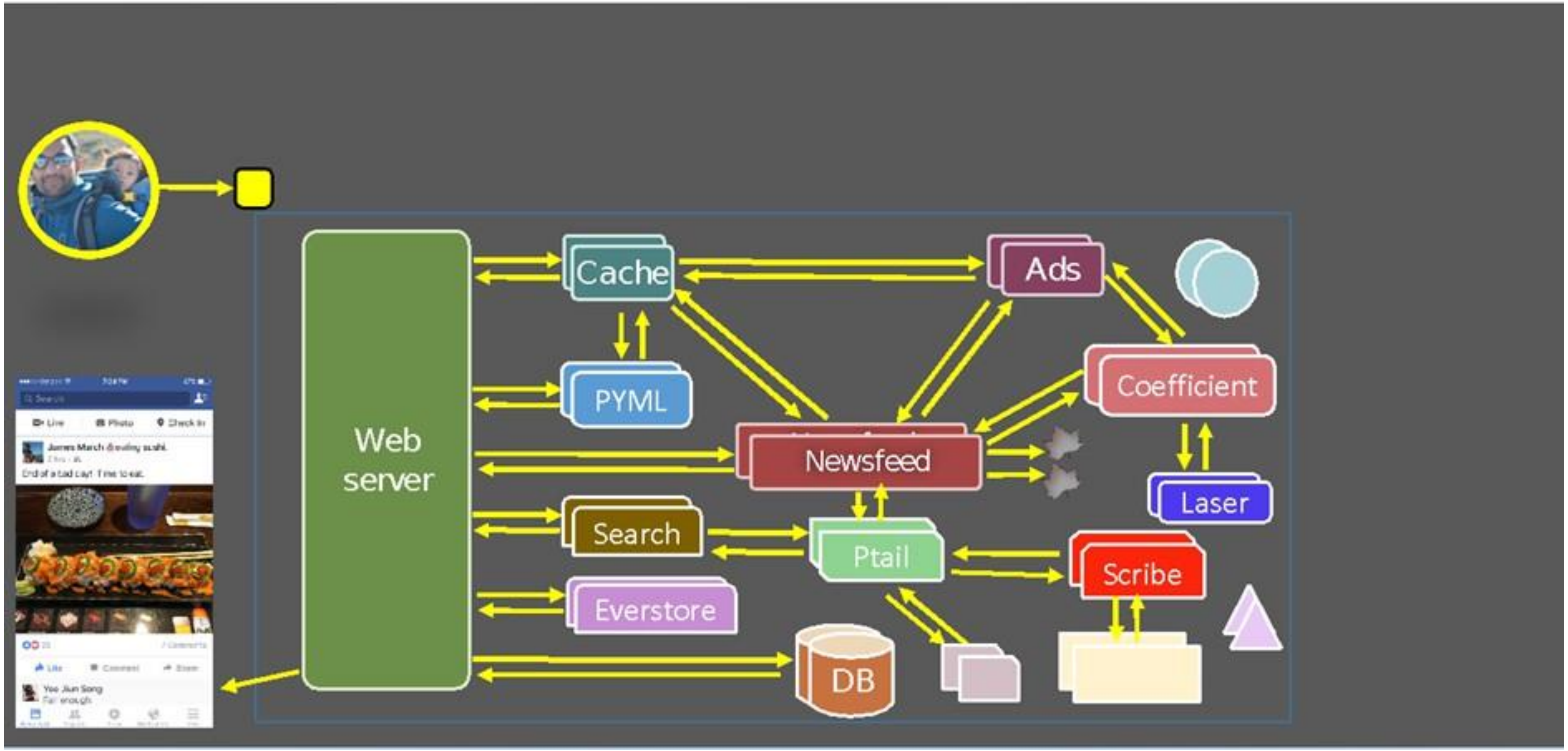


Systems Stack (Firefox to Wikipedia)



So Many Systems…

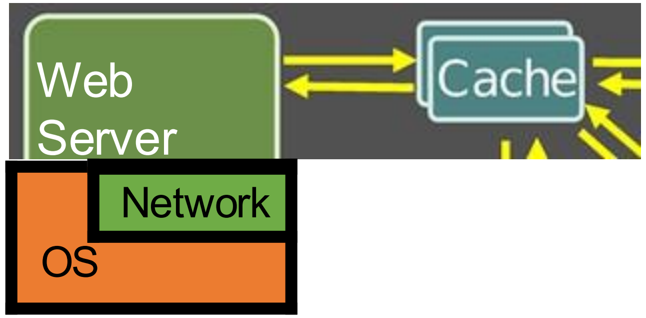
Each user request touches hundreds of systems



[Slide from Kaushik Veeraraghavan Talk’s on Kraken at OSDI ‘16]

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 do we build systems?

- Sharing: Mediates access to shared resources
- Portability: Abstract differences in underlying implementations
- Safety: Isolate resources and other applications from faulty apps
- Abstraction: Make complex resources easier to use

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>,<values>)
  - Read_and_write_transaction(<read_keys>, <write_keys>,<values>)
  - …

- Design tradeoffs!

Why Are Systems Challenging? Part-2c

- Distributed cache that provides fast access to popular data

- How portable should an interface be?
  - Cache in DRAM
  - Cache on SSD
  - Cache on NVM
  - Cache on HDD
  - …

- Design tradeoffs!

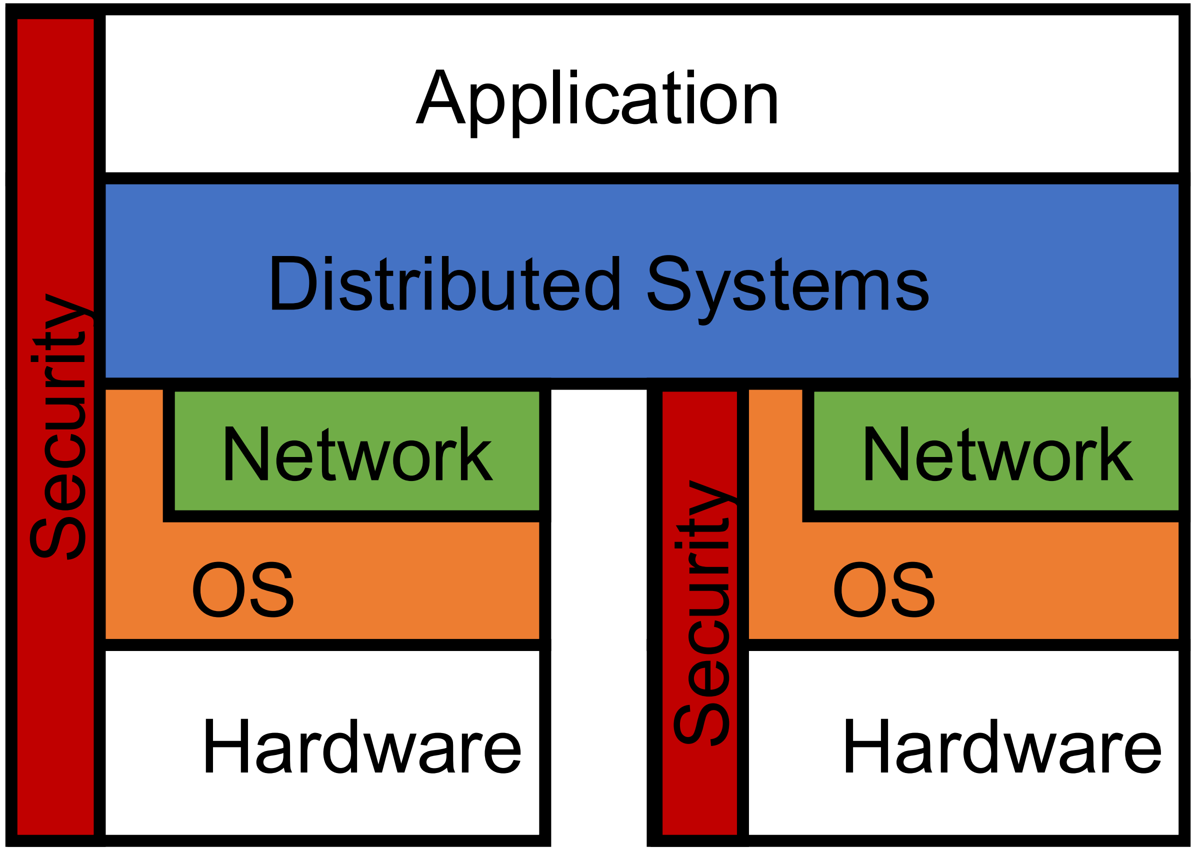
General vs Portable Interfaces

- 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?

Systems We Will Cover In This Class



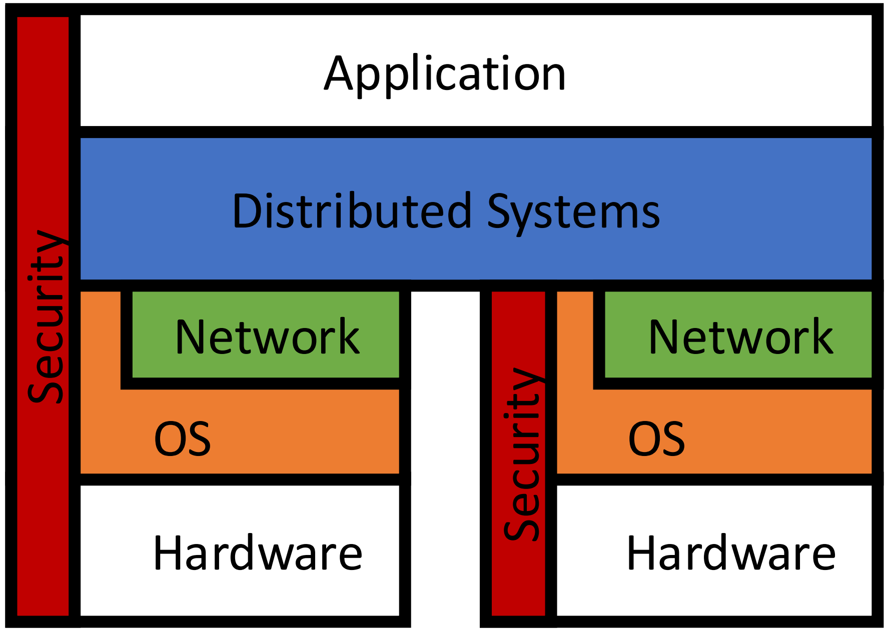
- Distributed Systems
- Networking
- Operating Systems
- Security

Why Do I Love Systems?!

- Work on the “hard” problems, so applications don’t have to
- Correctness as a puzzle: reason through all corner cases
- Performance is a different type of puzzle:
    - Where are bottlenecks, how to speed them up?
- Art of reasoning about tradeoffs: e.g., Interface vs. Performance
- 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


Application
Distributed Systems
Security
Network
OS
Hardware
Security
Network
OS
Hardware