Introduction

CS503: Operating systems, Spring 2019

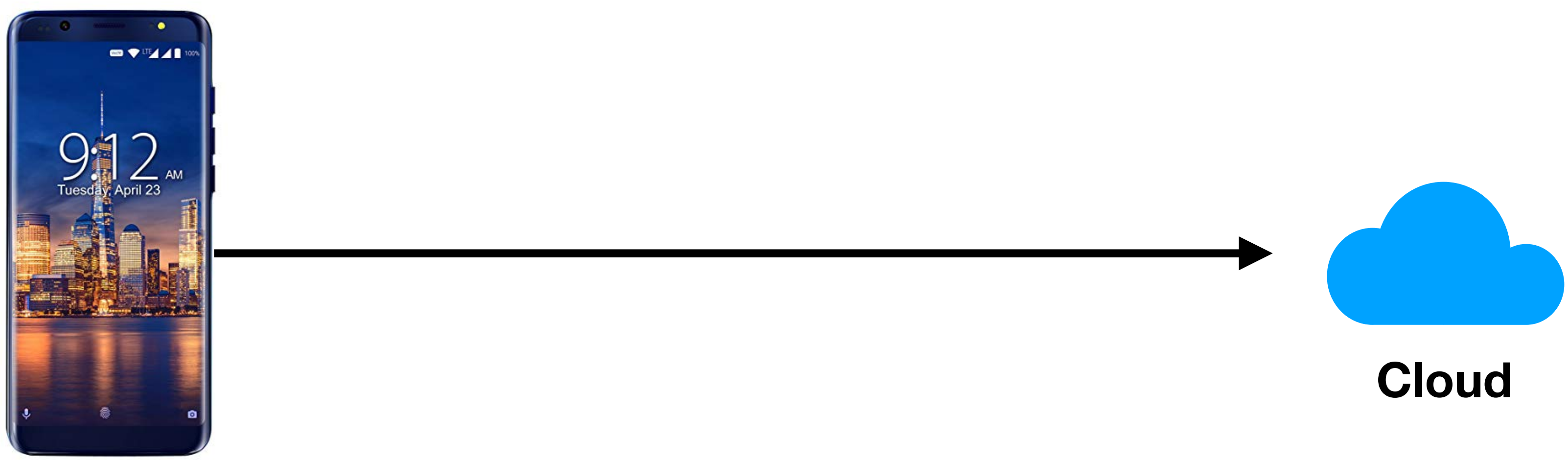
Pedro Fonseca
Department of Computer Science
Purdue University

Hi everyone!

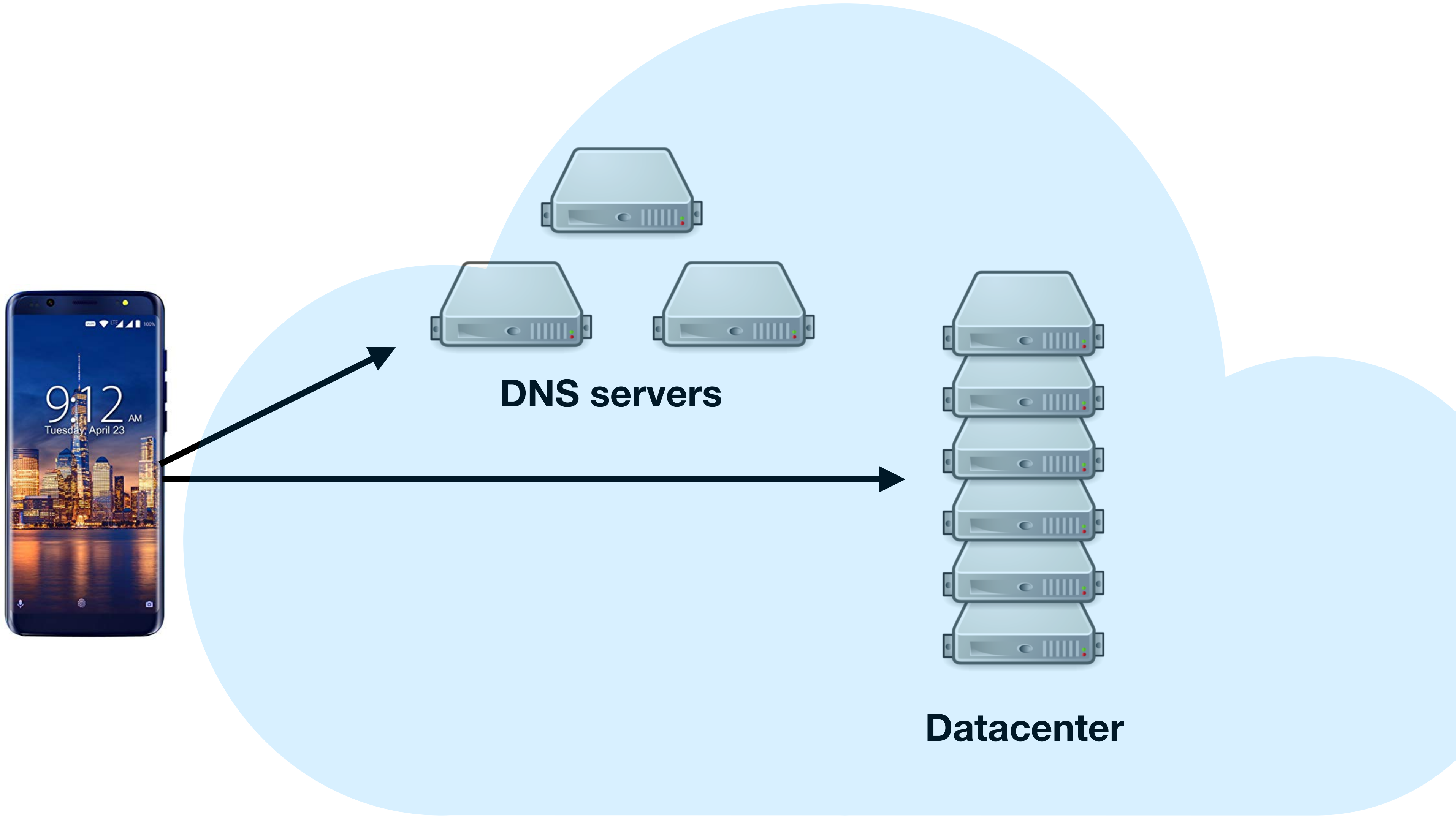
- Campus students and online students!
  - Fancy room, with video & audio recording for online students
- Please do **ask questions**, interactiveness is important

Finding a photo…today

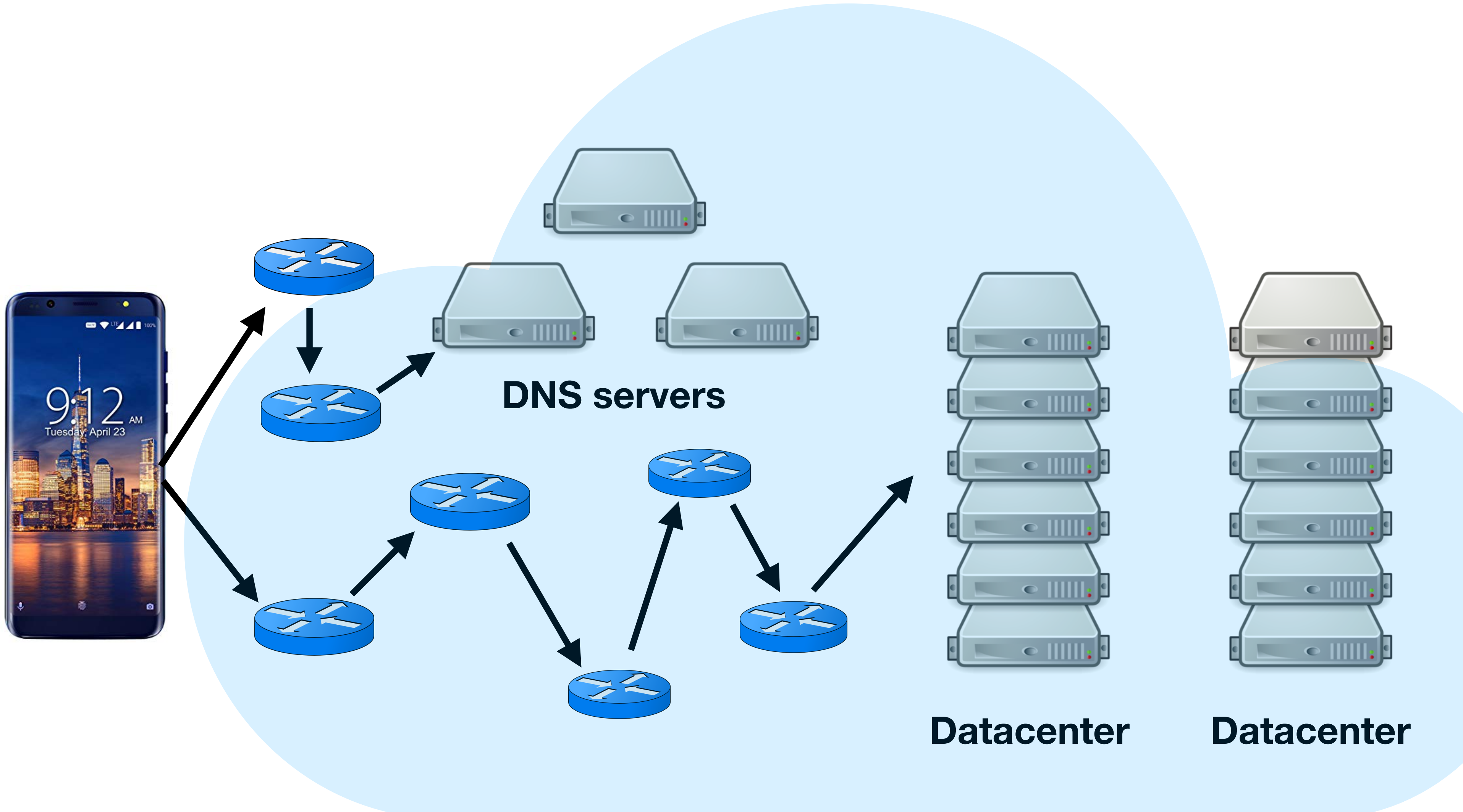
Finding a photo…today



Finding a photo…today

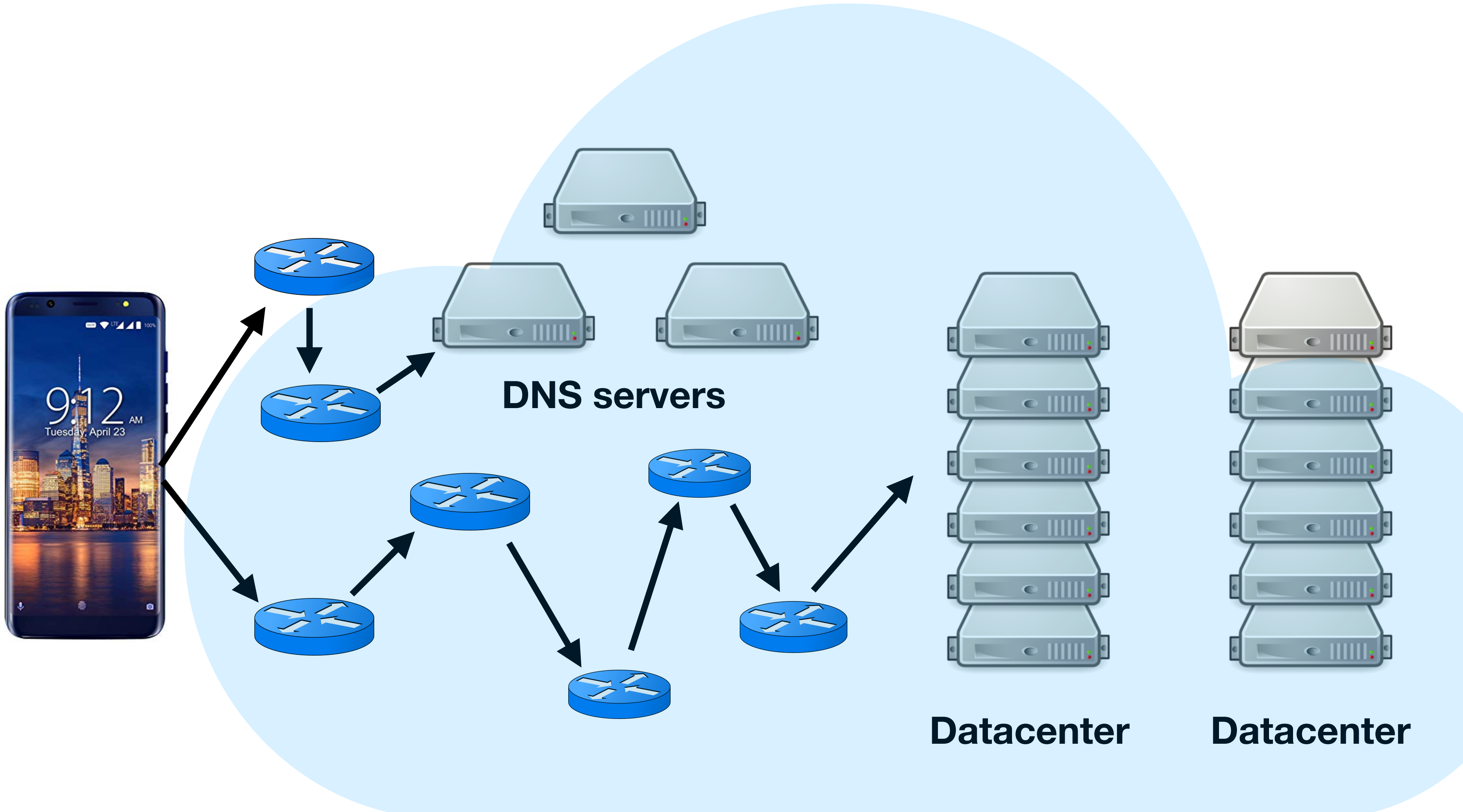


Finding a photo…today



Incredibly complex!!!

Finding a photo…today



Incredibly complex!!!

Finding a photo…today

Abstraction
Management of shared resources
Indirection
Concurrency
Atomicity
Protection
Naming
Security
Reliability
Scheduling
Fairness
Performance

Many of these problems have been first found in the context of operating systems

Why take CS503?

- Some of you are going to **write or modify operating systems**
- Some of you will do **research on operating systems**
- Some of you (All?) will **write applications** that interact with operating systems
- Some of you will **leverage operating systems concepts** in other contexts

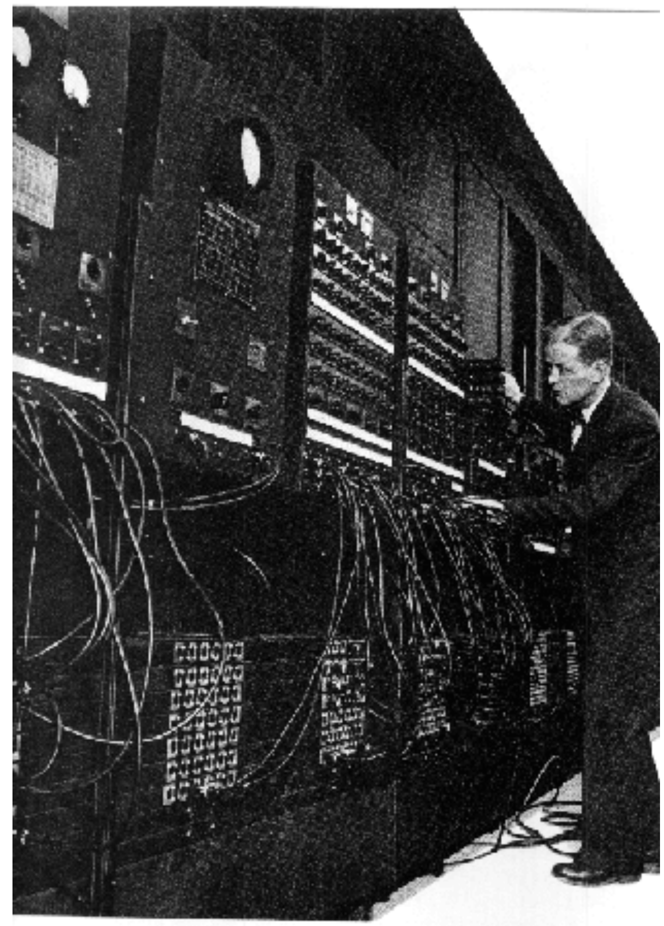
Goals for today

- What makes operating systems so exiting?
- Examples of operating system designs
- How does this class operate?
- What is an operating system?

A very brief history of operating systems

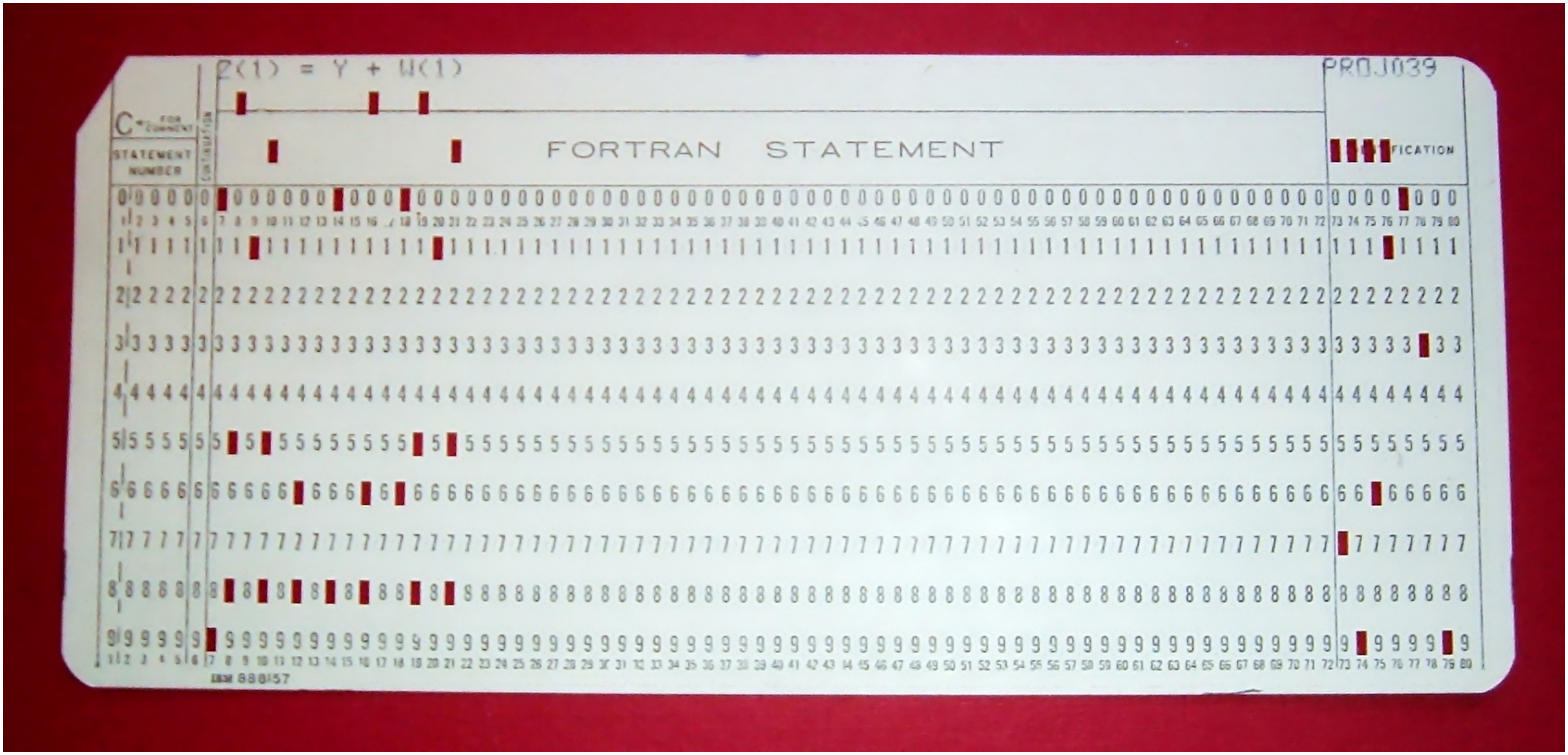
40’s - 50’s: no OS

- Machines very expensive
- No high-level PL and no OS
- Programmed machines with patch cables or punched-cards
- IO through punched-cards



ENIAC Computer
1945-1955

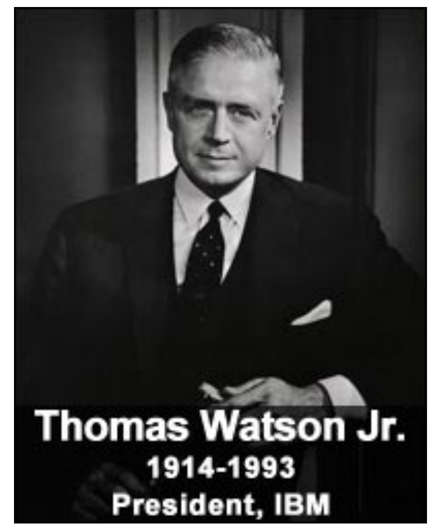
Punched-cards



"By 1937... IBM had 32 presses at work in Endicott, N.Y., printing, cutting and stacking 5 to 10 million punched cards every day."
—"IBM Archive: Endicott card manufacturing"

How many computers do we need?

- "*I think there is a world market for about five computers.”* — Thomas J Watson, IBM's president, early 1940s
  - Watson’s legendary misjudgment did not prove fatal to IBM



50's: batch processing

- The run queue is a deck of cards
- Not practical, slow
- One program at a time, no computation during IO
- OS functions: IO handler and loader

60’s: multiprogrammed batch systems

- When job waits for IO, switch to another job
- New OS functions:
  - Job scheduling policies
  - Memory management
  - Handle asynchronous IO devices
- Computer still not interactive

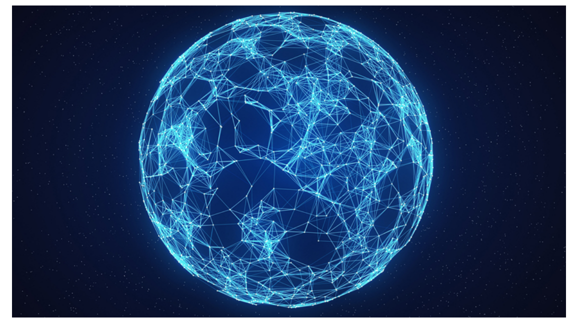
60’s-70’s: timesharing machines

- Jobs are quickly switched to give appearance of dedicated machine
- New OS functions: preemption, advanced scheduling policies

80’s: Personal computers

- Entire machine is inexpensive
- One user, one machine
- OS functions were simplified (e.g., DOS):
  - Single user, no timesharing, no virtual memory

Other recent developments….

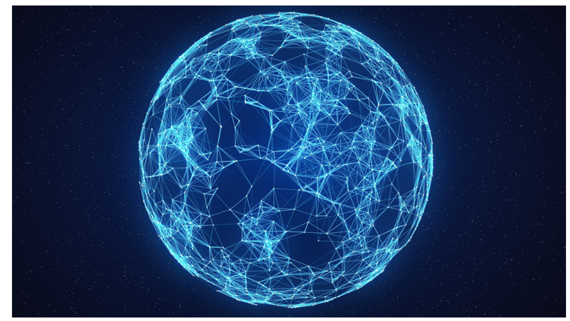


- Internet:
  - Network stack became part of every OS
  - Very efficient to support high-throughput and low latencies

- Computers are ubiquitous and come in all forms

Other recent developments….

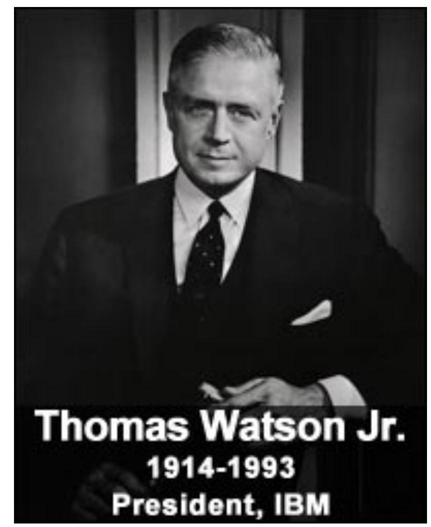
Internet of Things (IoT)



- Constrained devices
  (e.g., limited memory, hw protection)
- May not use a “normal” desktop OSs
- Reliability and security are critical

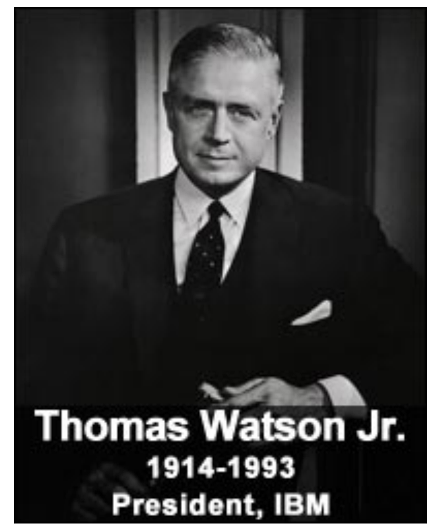
How many computers do we need?

- "*I think there is a world market for about five computers.”* — Thomas J Watson, IBM's president, early 1940s
    - Watson’s legendary misjudgment did not prove fatal to IBM



How many ~~computers~~ do we need?
datacenters

- *"I think there is a world market for about five computers."* — Thomas J Watson, IBM's president, early 1940s
    - Watson's legendary misjudgment did not prove fatal to IBM



Progression over the decades

- Hardware:
  - Vacuum Tubes (1950s) - one bit on the size of a thumb;
  - Transistors (1950s and 1960s) - one bit on the size of a fingernail;
  - Integrated Circuits (1960s and 70s) - thousands of bits on the size of a hand
  - Silicon computer chips (1970s and on) - millions of bits on the size of a finger nail.
- Ease of use and program:
  - Almost impossible to use except by very patient geniuses (1950s);
  - Programmable by highly trained people only (1960s and 1970s);
  - Useable by just about anyone (1980s and on).
- Complexity
- Reliability

Multics OS, an ambitious OS

- Multics OS (MIT + GE, 1969-):
  - Some say very successful, others very unsuccessful
  - Influential (Linux, Windows)
- Notable properties and functions:
  - Files and memory are treated identically
  - Dynamic linking
  - Engineered to be secured from the ground

We’ve come a long way!

- Operating systems were at the core of the revolution

- The **problems** uncovered and the **principles** developed remain fundamental
    - And applicable to many other contexts

OS History

- *The UNIX Operating System*, AT&T Archives, at YouTube
- *The UNIX Time-Sharing System*, Dennis M. Ritchie and Ken L. Thompson, Bell System Technical Journal 57(6)
- *The Evolution of the Unix Time-sharing System*, Dennis M. Ritchie
- *A Narrative History of BSD*, Kirk McKusick, at YouTube
- *From L3 to seL4: What Have We Learnt in 20 Years of L4 Microkernels?*, Kevin Elphinstone and Gernot Heiser
- *SOSP History Day*, October 4, 2015

http://sigops.org/s/conferences/sosp/2015/history/

SOSP History Day

October 4, 2015 — Monterey, California, USA

Introduction

Fifty Years of Operating Systems
Peter Denning
FOREWORD

Overview of the Day
Jeanna Matthews
ACM & VIDEO — SLIDES

The Founding of the SOSP Conferences
Jack Dennis
ACM & VIDEO — SLIDES

9am - 10:30am

Perspectives on OS Foundations
Peter Denning
ABSTRACT — ACM & VIDEO — SLIDES

Perspectives on Protection and Security
Butler Lampson
ACM & VIDEO — SLIDES

Perspectives on System Languages and Abstraction
Barbara Liskov
ACM & VIDEO — SLIDES

11am - 12:00pm

Evolution of File and Memory Management
Mahadev Satyanarayanan (Satya)
ABSTRACT — ACM & VIDEO — SLIDES

Reflections on the History of Operating Systems

Class operation

Staff

- Instructor: Pedro Fonseca, pfonseca@purdue.edu
  - MPI-SWS -> UW -> Purdue
  - Research interests: Operating systems, hypervisors, distributed systems
- TA: Basavesh Shivakumar, bammanag@purdue.edu
  - Research: Operating systems and concurrency

Acknowledgements

- Course slides heavily based on content by:
  - Douglas Comer
  - Dongyan Xu
  - Byoungyoung Lee

Resources

- Online resources:
    - Web page: Anchor
    - Piazza: Online discussion, slides, announcements
    - Blackboard: Grades
- PSOs
- Office hours
- Ask questions

Textbook

- Required textbook:
  - **Operating System Design: The XINU Approach**, Second Edition, Feb 18, 2015, by Douglas Comer
- Other recommended resources:
  - **Operating Systems: Principles and Practice**, 2nd Edition, by Thomas Anderson and Michael Dahlin
  - **Operating Systems: Three Easy Pieces**, by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau

Class pre-requisites

- Comfortable writing **C code** (e.g., know well how pointers and memory management works)
- Understand basic data structures (lists, arrays, etc.)
- Comfortable dealing with **large code bases** (e.g., 10-50k LOCs)
- Courses:
  - CS250 (Computer Architecture), CS251 (Data Structures), CS252 (Systems Programming), CS240 (C Programming), or equivalent

What to expect?

- **Code intensive** (lots of time invested understanding the code base, writing your own code and debugging)
- Understand and deal with low-level **architecture details** (e.g, assembly, CPU registers, memory regions, physical memory addresses)
- Lectures will be code intensive as well

Lecture Format

- Help you understand important and hard OS concepts, design, and implementation
- Lectures do not cover everything
  - Not all questions in exams are from lectures
  - But many (80%?) of them are :)
- Your responsibility
  - Attend lectures
  - Read code, textbook, lecture notes
  - Labs, quizzes, exams
  - Periodically check web page
  - Read/participate in Piazza discussion

PSOs

- Not mandatory, but attendance is highly encouraged
- Labs will be explained in the PSOs
- Schedule:
  - PSO 1: 9:30am - 11:20am, Wed @ HAAS 257
  - PSO 2: 9:30am - 11:20am, Fri @ HAAS 257
  - **No PSOs in the first two weeks of class**
- Online students: Special instructions

Grading

- Tentative grading criteria:
    - Midterm exam: 20%
    - Final exam: 25%
    - Labs: 50%
    - Quizzes: 5%

Exams and quizzes

- Exams and quizzes are close-book, close-note
- Online students:
    - Proctored exams
    - Blackboard quizzes

Late days for labs

- There is **no partial credit for late assignments**
- Granted three grace days (24-hour periods) that can be used for any laboratory any time during the semester
  - The three days can be applied to a single assignment (e.g., a lab) or one day can be applied to each of three assignments
  - Grace days must be used in **increments of one full day**
- Once your three grace days have been used, no further exceptions will be made
- Grace days cannot be used to extend the due date beyond the last day of regular classes

Grading policy for labs

- Projects will be done **individually**
  - General discussion allowed
  - Must be on your own when coding starts
- Don’t copy the one you found online
  - Will check your submissions against online solutions and other (including previous) submissions

Academic Integrity Policy

- Academic integrity: lifeline of education
- Your projects, quizzes and exams must be your own - we have a zero tolerance policy towards cheating of any kind and any student who cheats will get:
  - **Zero grade** for the project/quiz/exam at first offense
  - **F grade for the course** thereafter
- Both the cheater and the person(s) who aided the cheater will be held responsible for the cheating

Course organization

1. OS overview
2. Computer architecture overview
3. Process scheduling and management
4. Process synchronization
5. Memory management
6. Interrupt processing
7. Device drivers
8. File system

Your feedback

- Your feedback is very welcome!
  - The earlier, the better :)
- Feedback about the class organization, lectures, PSOs, etc.
- Might help you and your colleagues

What is the operating system?

What is the operating system?

- The software system that manages hardware resources for users and their applications

- Many other definitions

Types of OSs

- OSs may be invisible to “users”
- We’re going to focus on general-purpose operating systems (XINU, Linux, Windows, Android)

The three roles of (many) OSs

- **Referee**
  - Manages resources shared between different applications/users

- **Illusionist**
  - Provide an abstraction of physical hardware to simplify application design

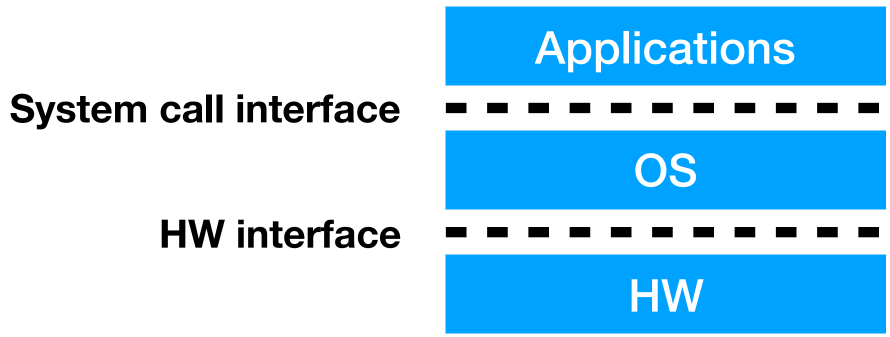
- **Glue**
  - Provides common services to facilitate sharing between different applications

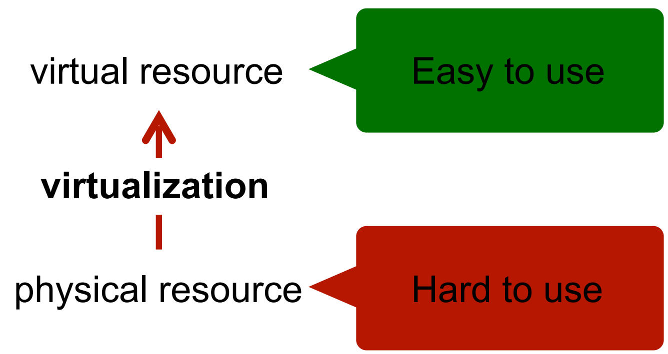
OS: The referee

- Sharing is central in most system uses
- Sharing raises critical challenges:
  - Resource allocation
  - Isolation
  - Communication
- OS: balances needs, separates conflicts, facilitates sharing

OS: The illusionist

- *Virtualization* provides the illusion of resources that are not physically present
- A few examples of challenges:
    - HW provides only a small amount of RAM
    - Different HW works in slightly different ways
- OS: virtualizes resources to simplify application development





OS: The glue

- How to make applications easily share information?
  - OS: provides the communication primitives
- E.g., files, messages, and copy&paste
- A minor role compared with the other two roles
  - Most of the OS complexity arrises

Other systems with similar challenges and designs

- Cloud computing
- Web browsers
- Media players
- Multi-player games
- Multi-user databases
- The internet

Questions?