SCC 311

An Introduction to Distributed Systems

Overview of the Session

- What is a distributed system?
- Examples of distributed systems
- Why distributed systems?
  - ➔ Historical distributed systems
  - ➔ Recent trends in distributed systems

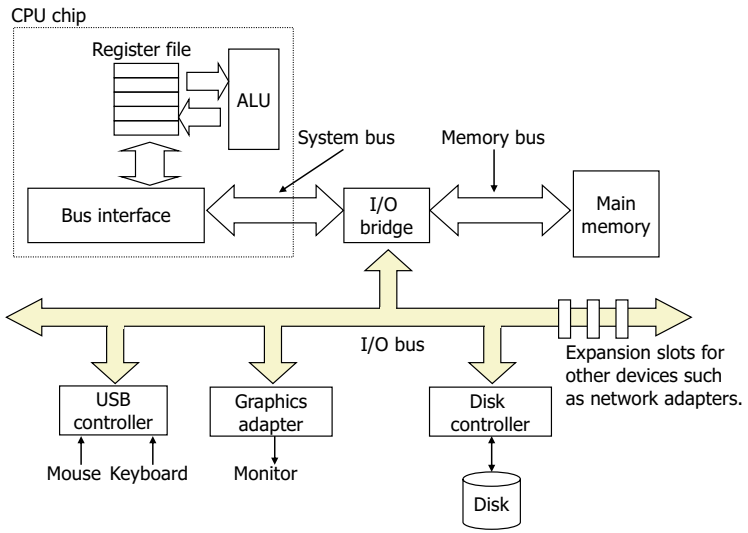
What is a System?

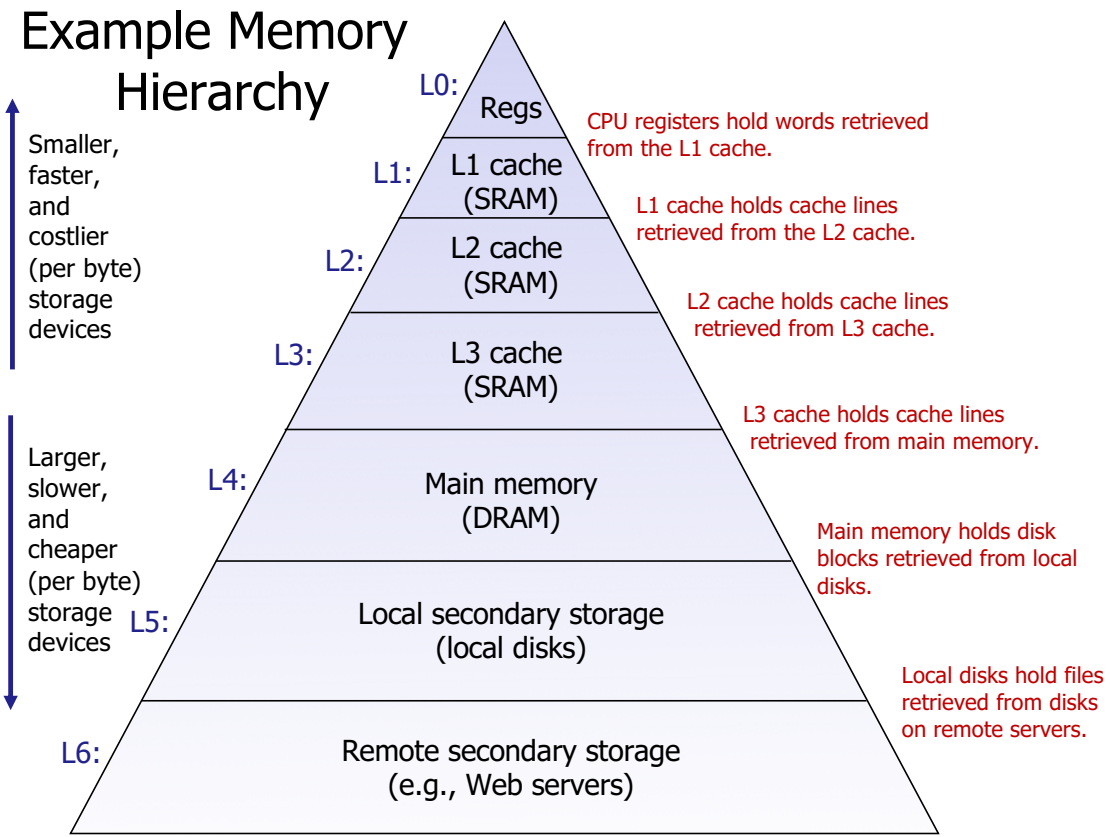
- A collection of hardware and software.
- “A computer system consists of hardware and systems software that work together to run application programs.”

[Bryant and O'Hallaron]

Tour of a Computer System

- Registers, L1/L2/L3 caches, RAM, local disk, remote disk
  - A memory hierarchy
- CPU running instructions





Tour of a Computer System

- Registers, L1/L2/L3 caches, RAM, Disk
  - memory hierarchy
- CPU with multiple cores
- **Operating system**
  - **Files, threads, …**
- **Assembly code**
- **Compilers**
- **Programs**

What is a Distributed System?

"a collection of independent computers that appears to its users as a single coherent system"

[Tanenbaum and van Steen]

"one in which hardware or software components located at networked computers communicate and coordinate their actions only by passing messages"

[Coulouris, Dollimore, Kindberg]

"one that stops you getting work done when a machine you've never even heard of crashes"

[Lamport]

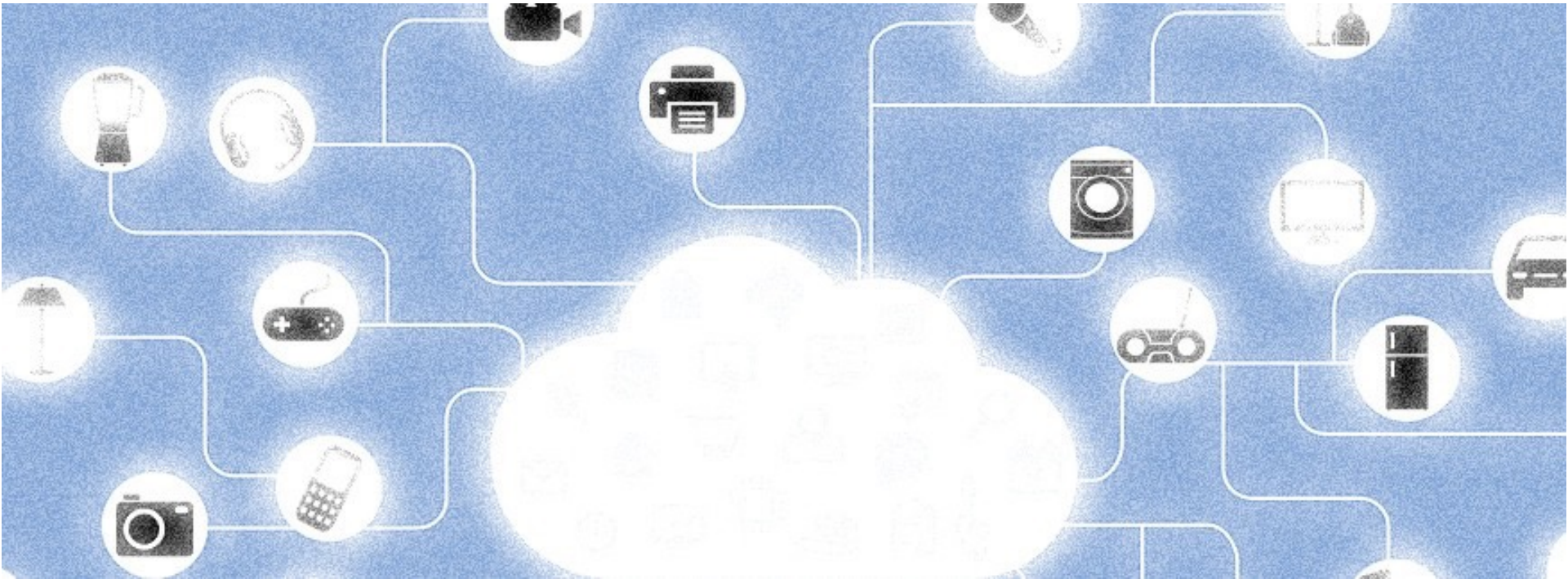
What is a Distributed System?

"a system designed to *support* the development of applications and services which can *exploit* a physical architecture consisting of multiple *autonomous* processing elements that *do not share primary memory* but cooperate by sending *asynchronous* messages over a communications network"

[Blair and Stefani, 1998]

What is a Distributed System?

In its broadest definition, a distributed system is one that comprises more than one computer with the goal of reaching a level of performance and/or providing a service that is quite difficult or infeasible to do on a single computer.



Examples of Distributed Systems

- **Financial trading**
  - ➔ Support execution of trading transactions
  - ➔ Dissemination and processing of events

- **Web search**
  - ➔ The web consists of over 6 billion web pages across 8.2 million servers with an average lifetime of about 2 months.
  - ➔ Modern web engines serve over 3.5 billion searches per day
    - › **Major distributed systems challenges**

- **Massively multiplayer online games**
  - ➔ Online games (e.g. Fortnite, Among Us) support large numbers of users viewing *and* changing a common world
  - ➔ Need for very low latency coordination to support gameplay

Why Distributed Systems?

- Because the world is distributed
  - → You want to book a hotel in Sydney, but you are in Lancaster
  - → You want to be able to retrieve money from any ATM in the world, but your bank is in London
  - → An airplane has 1 cockpit, 2 wings, 4 engines, 10k sensors, etc.
    - 0 Similarly railway networks, and other distributed transport systems
- Because problems rarely hit two different places at the same time
  - → As a company having only one database server is a bad idea
  - → Having two in the same room is better, but still risky
- Because joining forces increases performance, availability, etc.
  - → High Performance Computing, replicated web servers, etc.

Why Distributed Systems?

acmqueue

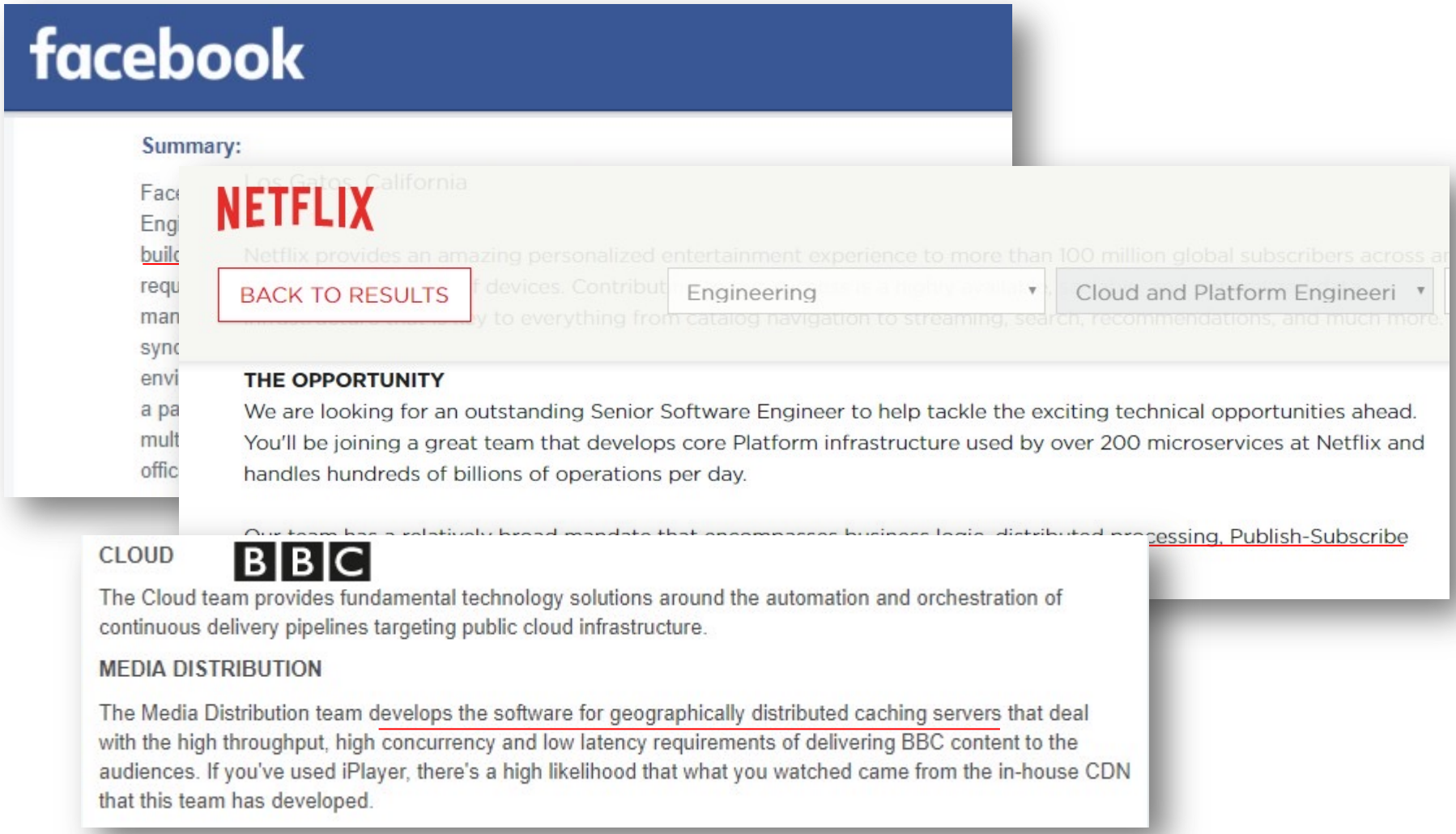
There's Just No Getting around It: You're Building a Distributed System

Building a distributed system requires a methodical approach to requirements

Mark Cavage

Distributed systems are difficult to understand, design, build, and operate. They introduce exponentially more variables into a design than a single machine does, making the root cause of an application problem much harder to discover. It should be said that if an application does not have meaningful SLAs (service-level agreements) and can tolerate extended downtime and/or performance degradation, then the barrier to entry is greatly reduced. Most modern applications, however, have an expectation of resiliency from their users, and SLAs are typically measured by "the number of nines" (e.g., 99.9 or 99.99 percent availability per month). Each additional 9 becomes harder and harder to achieve.

Is this important?



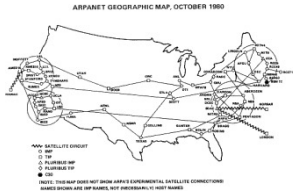
A Short History of DSs

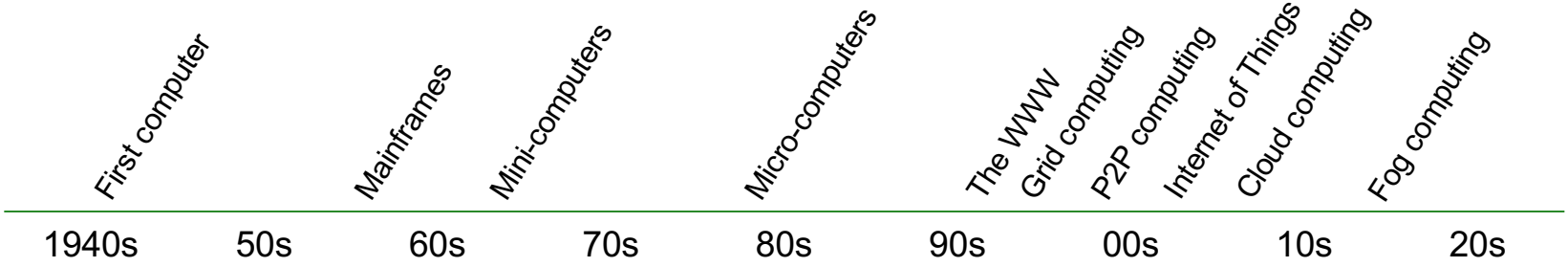
- Distributed *Computer* Systems are largely concerned with:
  - ➔ Data processing/ management/ presentation ("computing" side)
  - ➔ Communication/ coordination ("distributed side")
- Those concerns existed well before computers were invented
  - ➔ Ancient civilisations needed efficient communication:
    - cf. the Postal Service of the Persian Empire (6th century BC), the Roman roads (many still visible today), etc.
    - assuming a good infrastructure (roads, horses, staging posts)
    - max message speed: ~ 200 miles/day in the Persian system

  - ➔ Delays impose distributed organisations
    - Persian and Roman empires extended over 1000s of miles
  - ➔ Trust / secrecy / reliability issues
    - Am I sure Governor X is doing what he says he is?
- This all has not really changed!
  Things have only sped up.

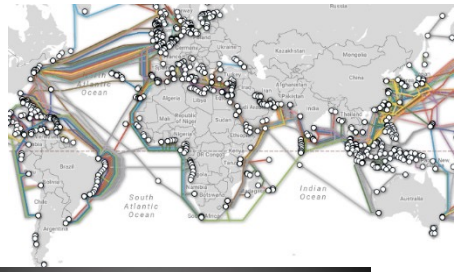
A Short History of DSs

- Computers are far more recent
  - ➔ The first "modern" computers appeared just after WWII
  - ➔ They were slow, bulky, and incredibly expensive
    - The ENIAC (1945), used by the US army: 30 tons, 170 m^2 footprint, 180 kilowatts, 18,000 vacuum tubes, and 5,000 additions/second (5KHz),
    - Price: $500,000 (in US$ of the time, would be > $5,000,000 today)
- And distributed computing is even more recent
  - ➔ For a long time, only very few computers were around anyway
  - ➔ No practical technology to connect them
  - ➔ This all changed in the 80's:
    - The rise of the **micro-computers** (PC, Mac, etc.)
    - The launch of the **"Internet"** (1982, TCP/IP), after 10 years of development

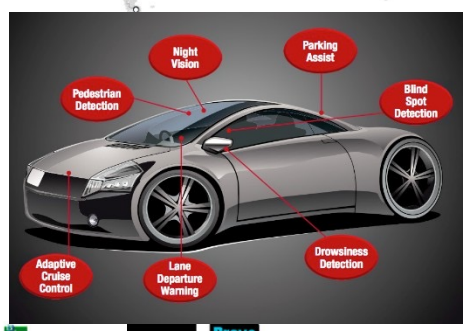




Recent Trends



- Pervasive networking and the modern Internet
  - ➔ Highly heterogeneous networking technologies
  - ➔ Fusion of services (telephony, broadcasting…)



- Mobile and ubiquitous computing
  - ➔ Personal devices and embedded systems

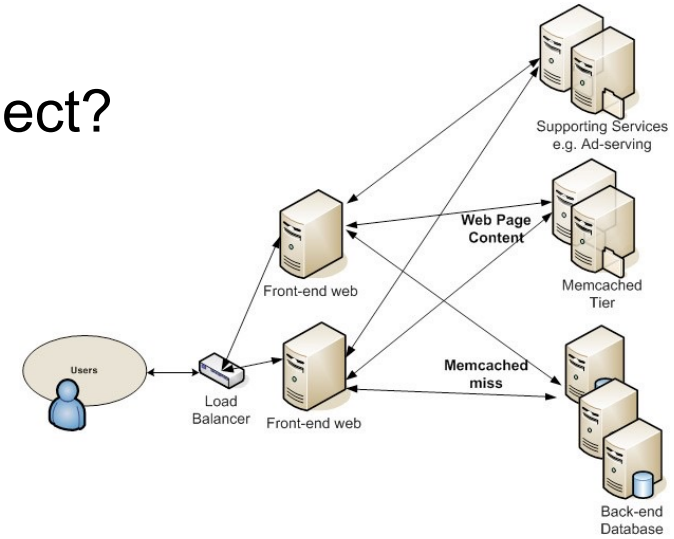
- Distributed multimedia systems
  - ➔ IPTV and Voice over IP services

- Distributed computing as a utility
  - ➔ Grid, Cloud, and Fog computing

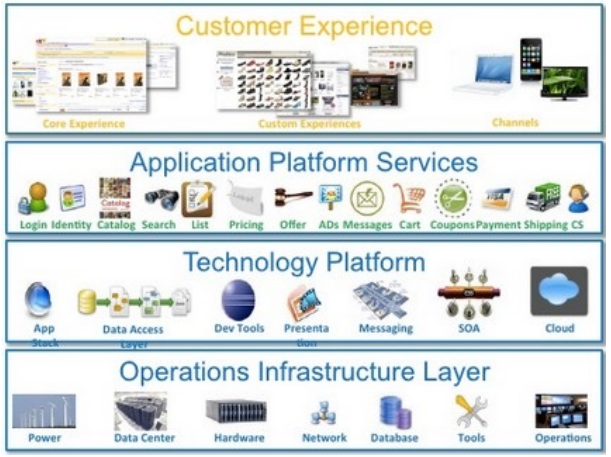
- The pace of change is accelerating

Interactions

- What sort of interactions do we expect?

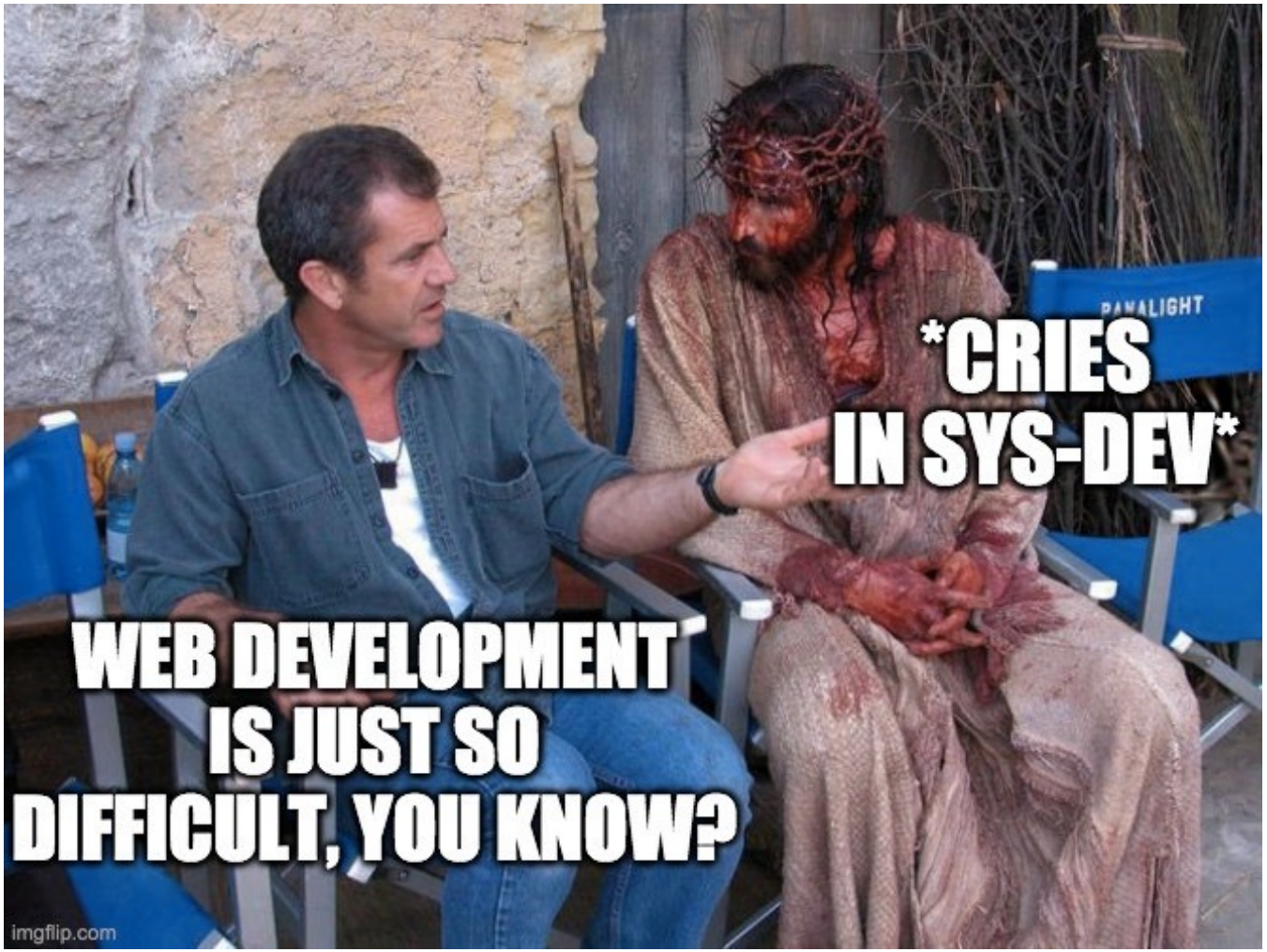


- Web
  - ➔ Client: “I want this thing”
  - ➔ Server: “Here’s the thing”
- eBay
  - ➔ Alice: “I bid £10”
  - ➔ Bob: “I bid £12”
  - ➔ eBay: “Alice wins. Bob was late”



- More on design patterns later (week 4).

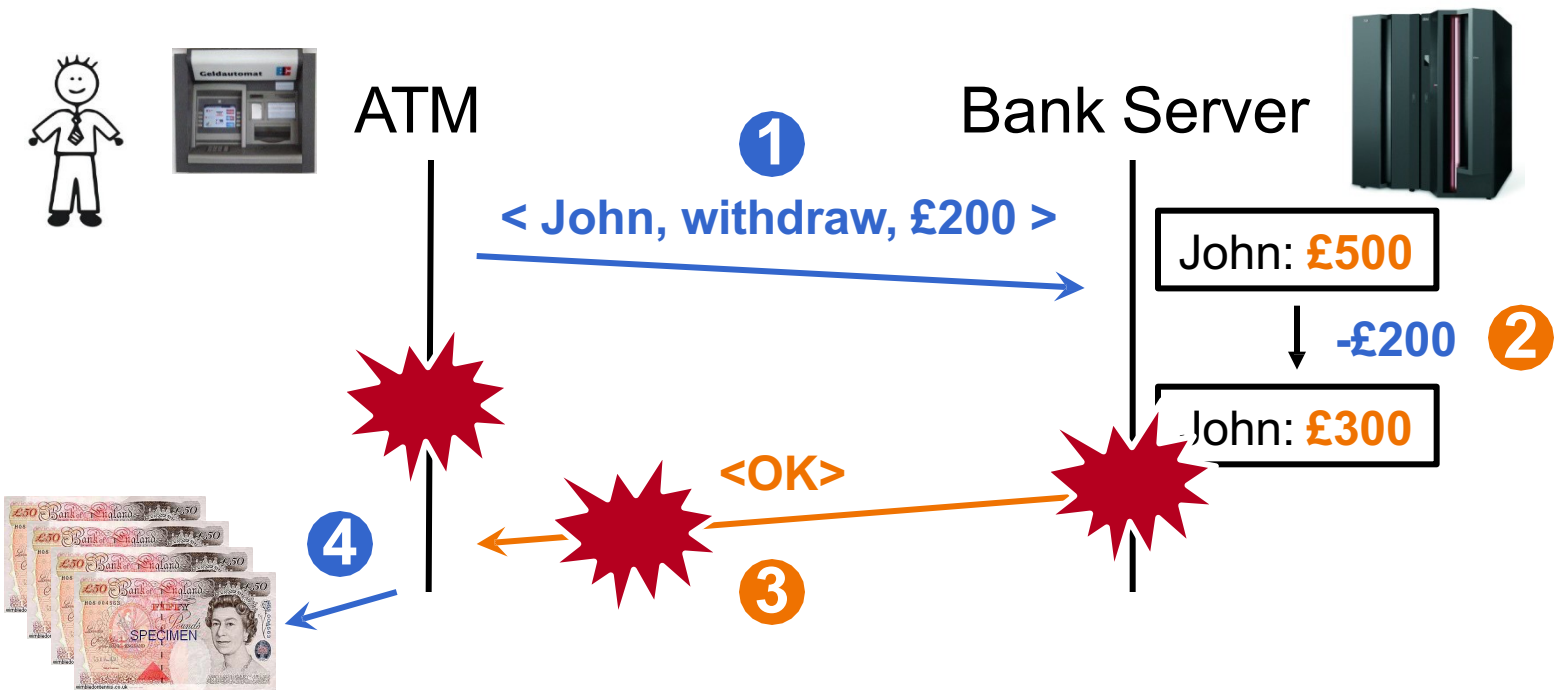
Is this easy?


CRIES IN SYS-DEV
WEB DEVELOPMENT IS JUST SO DIFFICULT, YOU KNOW?
imgflip.com

But why is this so hard?

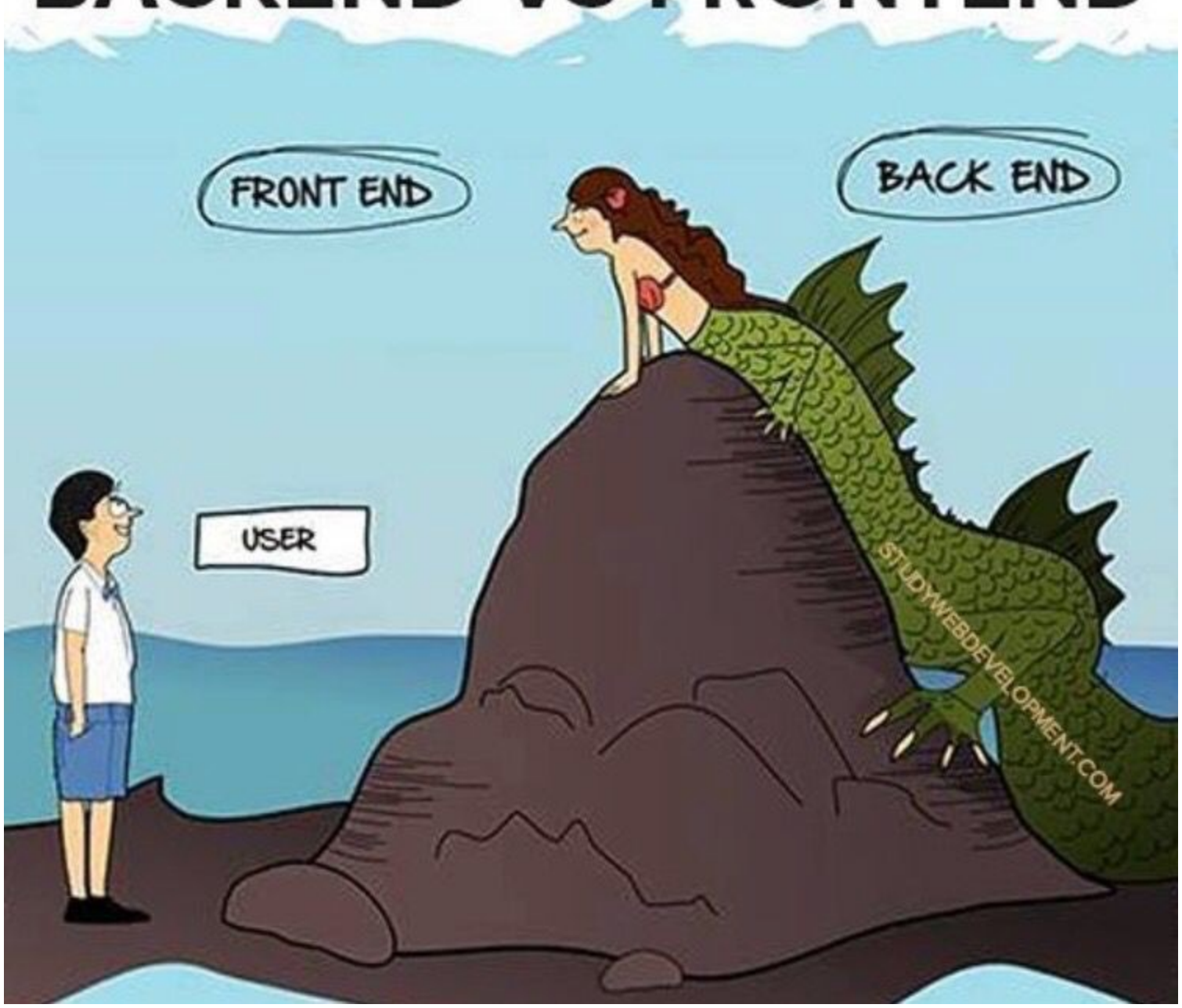
- A bank asks you to program their new ATM software
  - ➔ Central bank computer (server) stores account information
  - ➔ Remote ATMs authenticate customers and deliver money
- A first version of the program
  - ➔ ATM: (ignoring authentication and security issues)
    1. Ask customer how much money s/he wants
    2. Send message with **<customer ID, withdraw, amount>** to bank server
    3. Wait for bank server answer: **<OK>** or **<refused>**
    4. If **<OK>** give money to customer, else display error message
  - ➔ Central Server:
    1. Wait for messages from ATM: **<customer ID, withdraw, amount>**
    2. If enough money withdraw money, send **<OK>,** else send **<refused>**

But why is this so hard?



- But ...
  - ➔ What if the bank server crashes just after 2 and before 3?
  - ➔ What if the <OK> message gets lost? Takes too long to arrive?
  - ➔ What if the ATM crashes after 1, but before 4?
- More on challenges in coming lectures.

BACKEND VS FRONTEND



Expected Learning Outcomes

At the end of this session, you should be able to:

- Define distributed computer systems
- Explain why distribution is needed
- Know a few everyday examples of distributed systems
- Understand how they evolved into their current form

Additional Reading

- **Required: CDKB, chapter 1 sections 1.1 -- 1.4**
  - ➔ also chapter 3 for revision
- TvS, chapter 1
- Computer Systems: A Programmer's Perspective, by Bryant and O'Hallaron, chapter 1
- Mark Cavage, “There's Just No Getting around It: You're Building a Distributed System”, ACM Queue, Vol. 11 No. 4, April 2013. http://queue.acm.org/detail.cfm?id=2482856