Distributed Systems

CSCI 5673
1/8/2026
Mark Zhao

College of Engineering & Applied Science
UNIVERSITY OF COLORADO **BOULDER**

What is a distributed system?

collection of devices

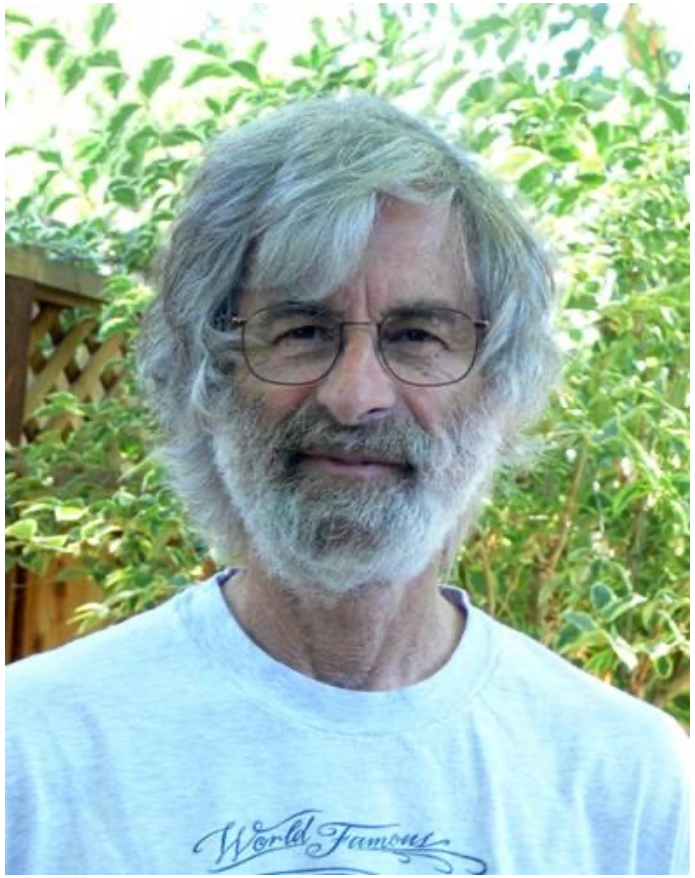
- collaborating on a task

Internet ⇒ networked

- Autonomous

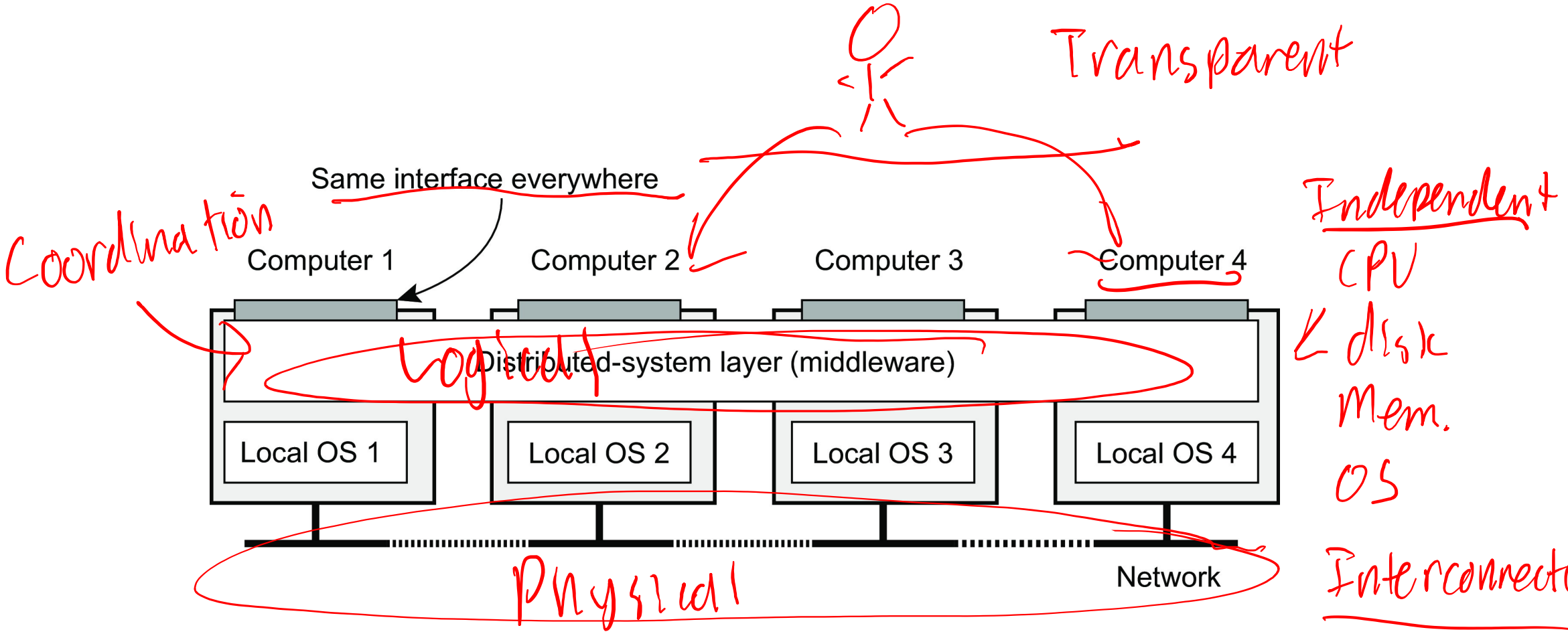
(For the most part)

What is a distributed system?



"… a system in which the failure of a computer you didn't even know existed can render your own computer unusable." – Leslie Lamport

What is a distributed system?



What is a distributed system?

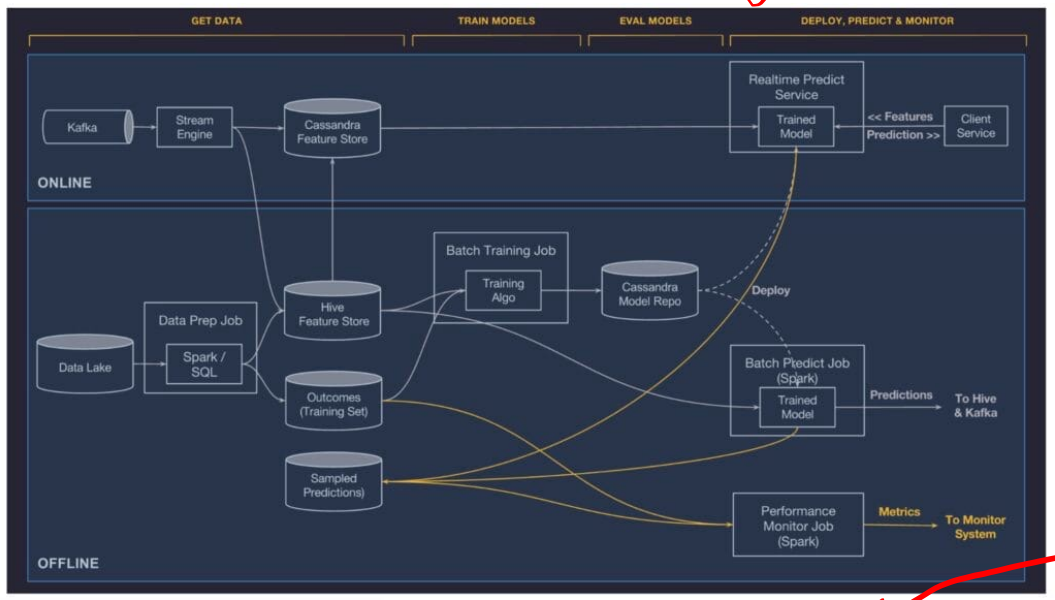
*A collection of **independent** computing components, **interconnected** via a network, that are capable of **collaborating** on a task.*

- **Independent**
  - Own set of compute resources
- **Interconnected**
  - Communicate over a network to exchange information
- **Collaborative**:
  - No centralized control
  - **Hide** the fact that processes/resources are physically distributed, possibly globally

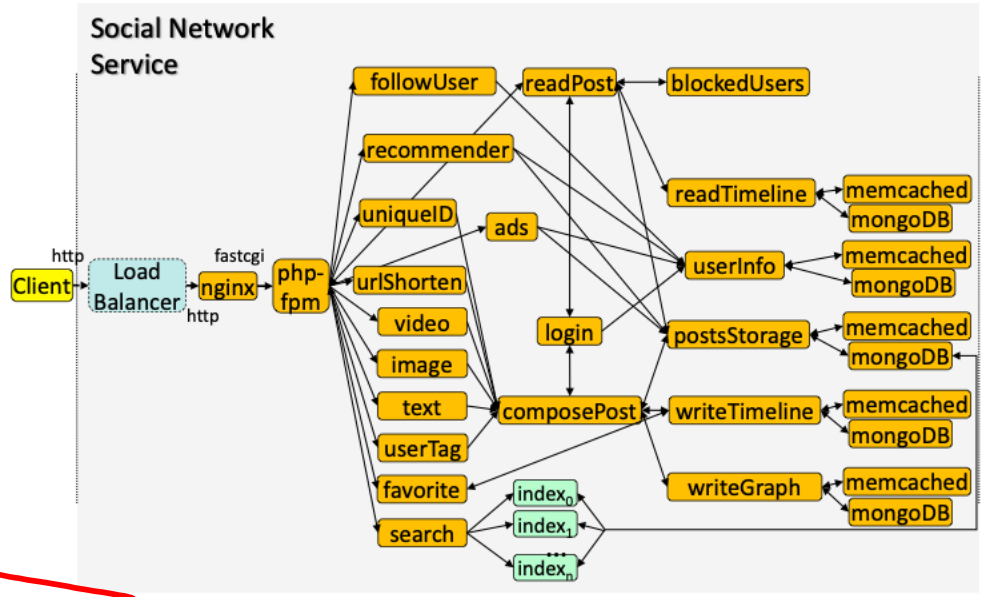
What is a distributed system?

Databases
→ Transactions

Uber's ML System



Microservice



UK

Facebook

US

Lower Latency
→ CDNs

https://www.uber.com/blog/michelangelo-machine-learning-platform/

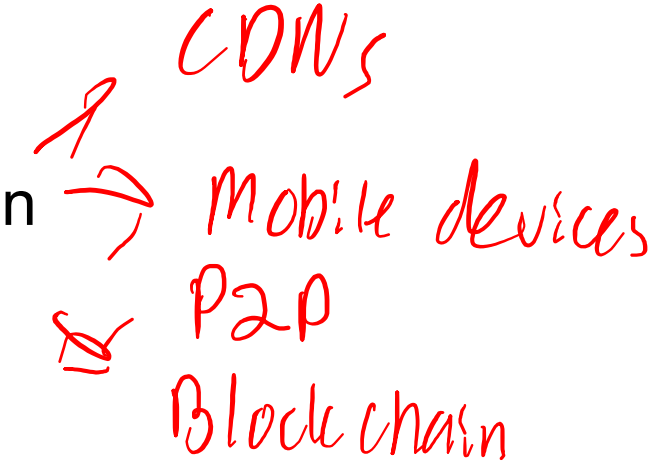
https://www.csl.cornell.edu/~delimitrou/papers/2019.asplos.microservices.pdf

Why do we want to distribute systems?

- Low Latency (CDNs)
- Parallelism / Scalability
  - 10x resources → 10x more throughput / tasks
- Reliability / Availability
  - ⇒ If one node fails, we can still make progress
- Features / Modularity / Extensibility

Why do we want to distribute systems?

- Scalability
  - Maintain performance as users/data/… grows
  - How? Increase compute/memory/storage/network/… capacity via parallelism
- Incremental Growth
  - Gradually add/remove components as needed
- Fault tolerance
  - Keep going even if one/few nodes fail
- Distribution
  - Match physical placement of components to application needs



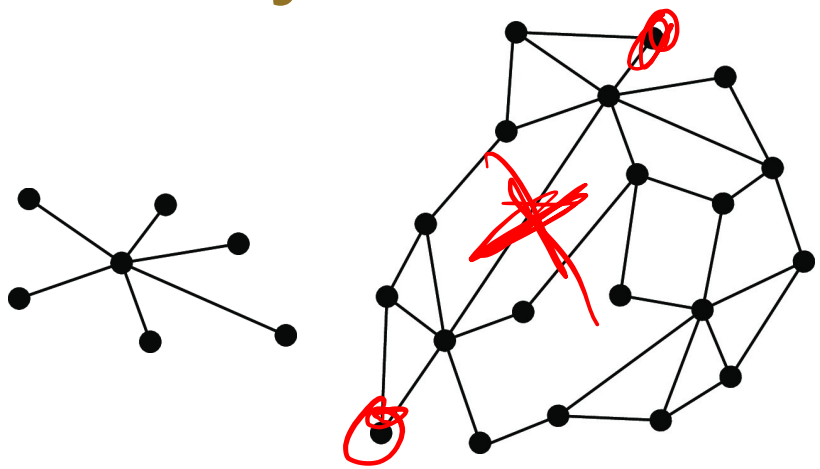
Why do we want to distribute systems?

- Security
  - Isolate sensitive components from others
  - Modularity, principle of least privilege
  - Mutually distrusting parties ← blockchains
- Heterogeneity
  - Adapt to different HW, devices, etc.
- Mobility
  - Mobile and “Ubiquitous” computing

Big goal: Hide the complexity of distribution from applications

Why do we **not** want to distribute systems?

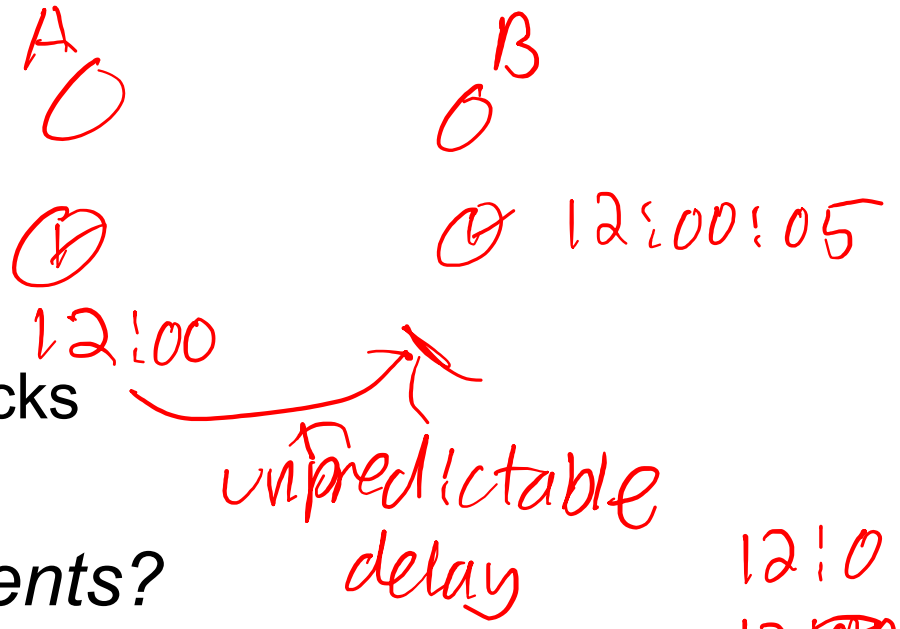
- **Complexity**
  - System architecture much more involved
  - Administrative complexity
- **Performance bottlenecks**
  - ”Critical path” involves many more steps
- **Inconsistencies**
  - What if two nodes disagree?
- **Partial failures**
  - What if systems/networks fail?
- **Security**
  - How do we ensure components are trusted?



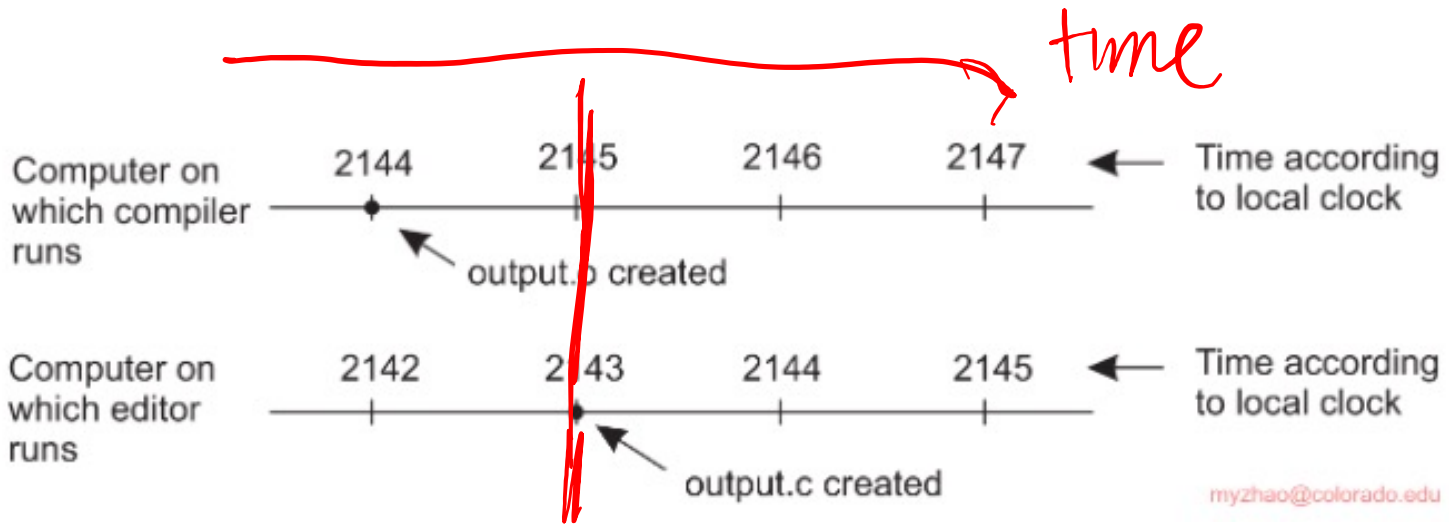
(Some) Limitations of distributed systems

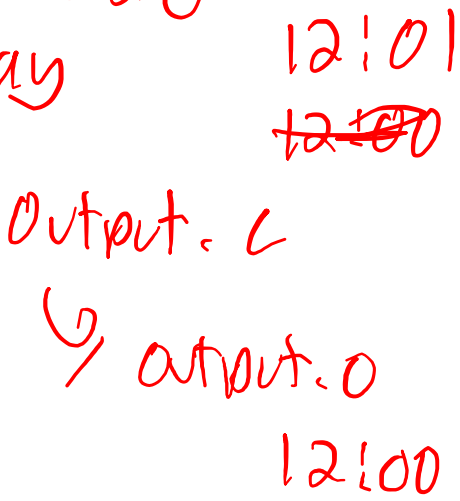
Absence of a common global clock

- No notion of a shared "global" time
  - Each component has its own clock
  - Impossible to perfectly synchronize all local clocks



- *Challenge: What's the temporal order of events?*



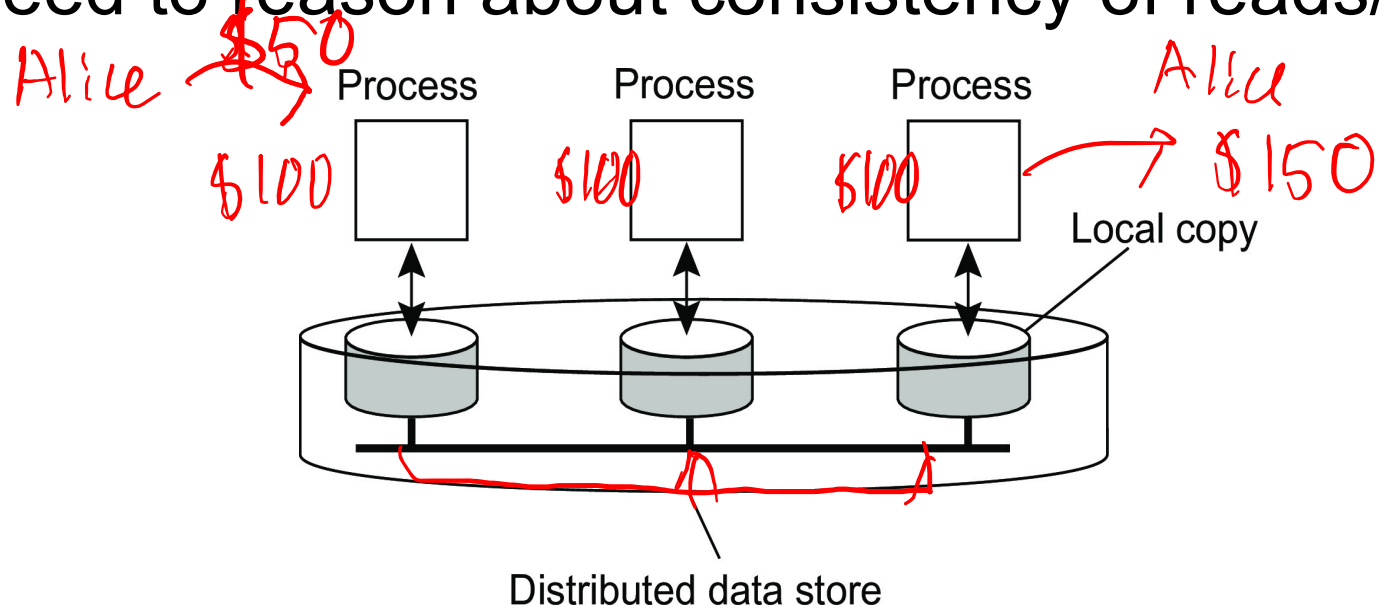


(Some) Limitations of distributed systems

Absence of shared memory

- Notion of a common shared memory (RAM, disk, …) does not exist
- Each component may have its own memory

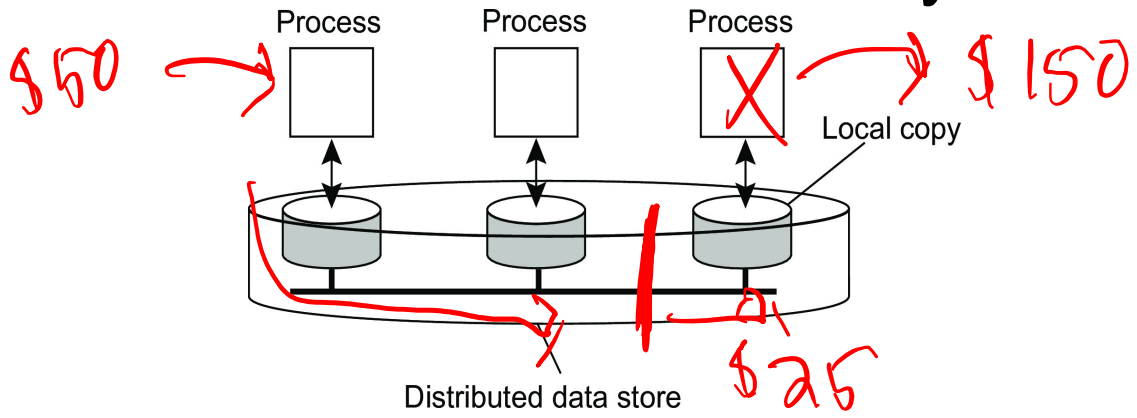
- *Challenge:* Need to reason about consistency of reads/writes across replicas



(Some) Limitations of distributed systems

Absence of any guarantees in reliability

- Real-world servers/networks can “break” at any point
  - No guarantees that processes are always alive
  - No guarantees that network messages ever arrive
    - …and if they do, no guarantees on *when* they arrive
  - No guarantees that messages arrive intact
    - What if a malicious actor or just pure chance causes a message to say the “wrong” thing
- *Challenge:* How do we still let the overall system make progress?



Why study distributed systems?

- These challenges come up when building pretty much any modern application
  - Web apps, AI systems, mobile apps, …
- Understanding how to reason about and solve these issues will help across a broad set of challenges
  - System design in industry – what tools / techniques to use to deploy your product
  - Active research area – lots of unsolved problems!
- It’s interesting!
  - Hard problems with powerful solutions

Teaching staff

- Mark Zhao
  - Assistant professor, computer science
  - https://basil.colorado.edu
- Rezwan Rahman
  - PhD candidate, systems
- Doncey Albin
  - PhD candidate, robotics
- Charith Purushotham
  - MS CS

Course structure

- Course website is the main source of truth for schedule/logistics/etc.
  - https://basil.colorado.edu/courses/sp26/csci5673/about/
- Canvas
  - Course announcements, assignment releases, lecture slides
- Gradescope
  - Assignment submission
- Piazza
  - Course Q&A and discussion forum

Course structure

- Homeworks and programming assignments (40%)
  - ~4 homework assignments based on concepts from lectures
  - ~4 lab assignments that get you to build a large-scale distributed system from scratch
- Exams (40%)
  - Midterm in class on 3/12
  - Final exam during finals week
- Project (20%)
  - Open-ended project building and deploying a distributed system
  - Intermediate project updates to build up to a full “paper”
  - In groups of up to 3, start forming project groups!

Prereqs and Materials

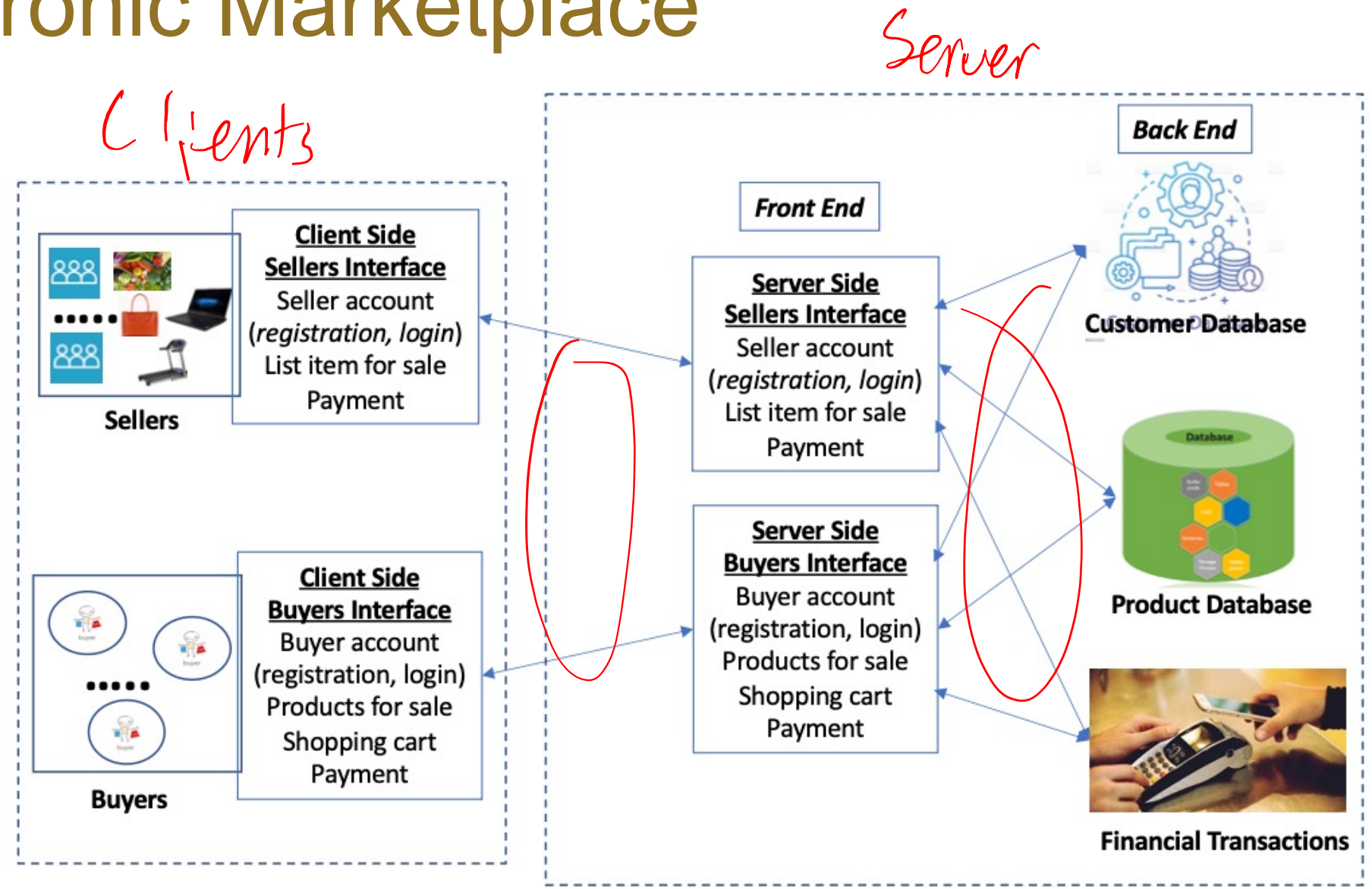
- Course in Operating Systems (CSCI 3753)
  - Processes, threads, synchronization, memory management, file systems, virtualization, …
- Network Systems (CSCI 4273/5273)
  - TCP/UDP, HTTP, DNS, Socket API and programming, wide area networks, …

- Readings published on course schedule
  - Distributed Systems by van Steen and Tanenbaum (available online)
  - Designing Data-Intensive Applications by Kleppmann (available online)
  - Selected papers from literature (mostly around large-scale, distributed systems deployed at Google/Amazon/Meta/…)

Programming assignments / course outline

- Your job is to build up a distributed marketplace system
  - E.g., Amazon, eBay, etc.
- Sellers can list items for sale
- Buyers can view and purchase items for sale

Electronic Marketplace

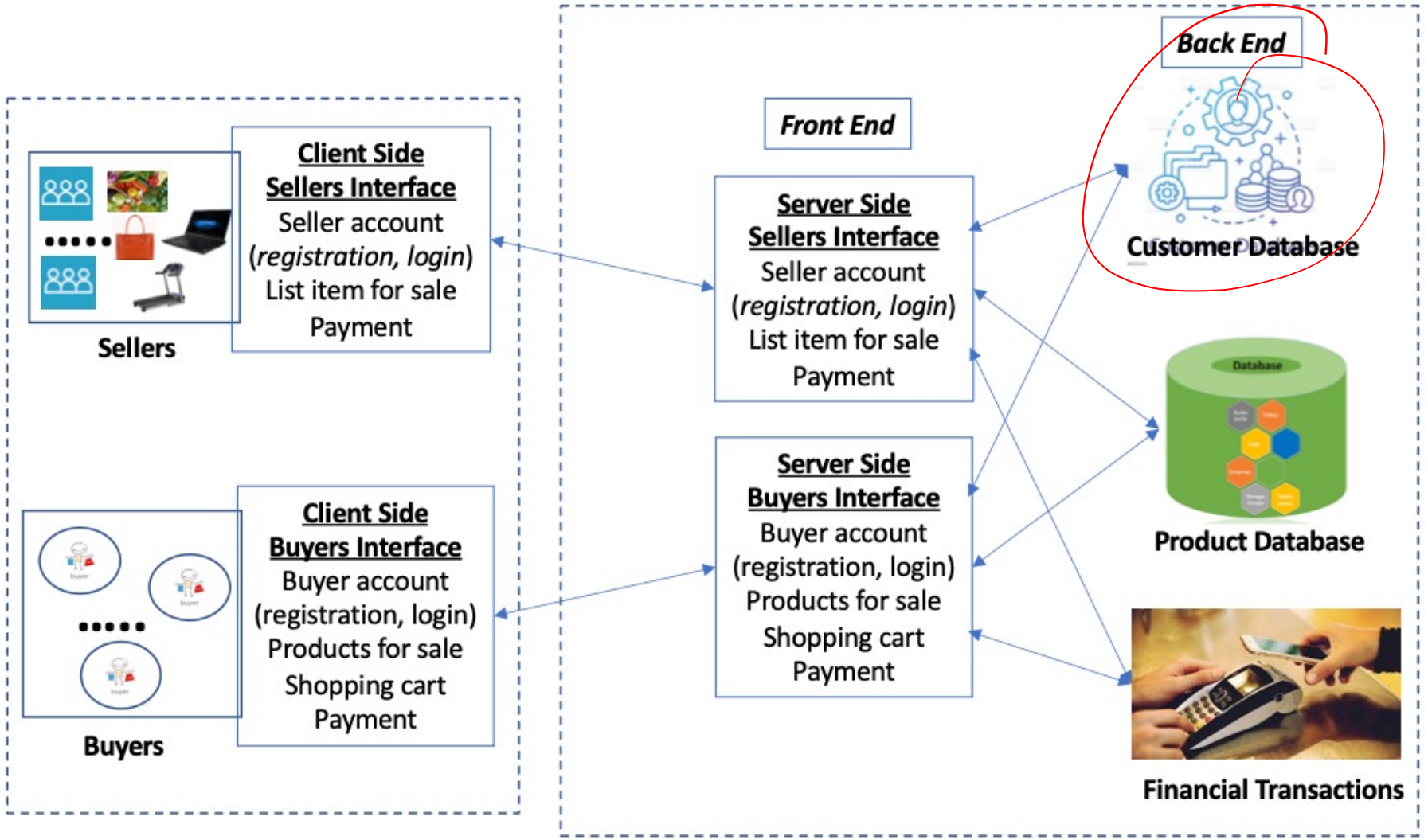


Interprocess Communication

- TCP/IP
  - Foundation of communication across distributed processes
  - Typically, too “low-level” to use directly

- Remote Procedure Calls (RPC)
  - Goal: Provide easy-to-program client/server communication
  - Review local procedure calls
  - Stub functions, parameter passing, server binding, data representation

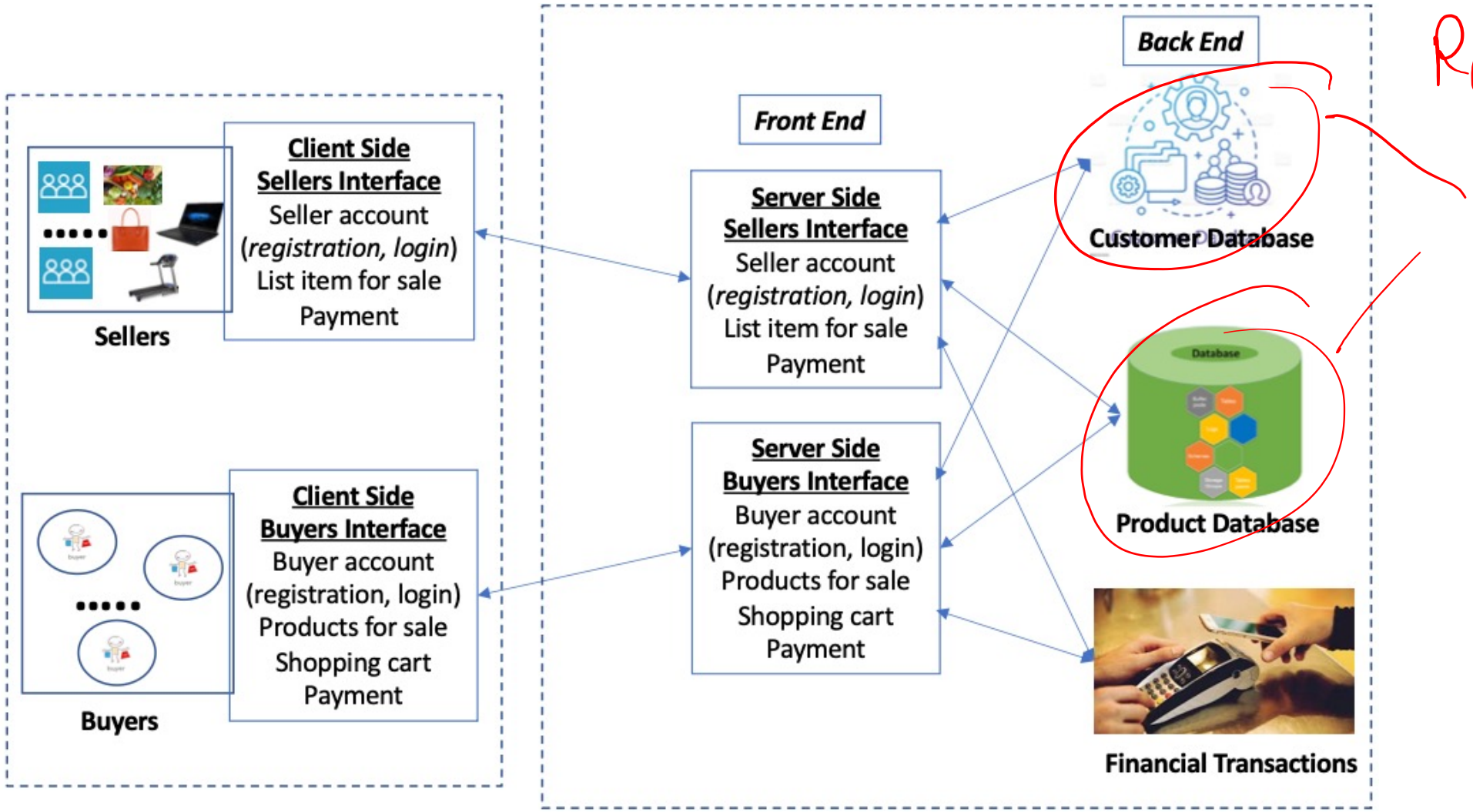
Electronic Marketplace



System models and failure models

- What do we care about in a distributed system?
  - What goals are we trying to achieve?
  - What assumptions can/can’t we make?
  - What are some typical architectural styles?
- What can go wrong?
  - How can processes misbehave?
  - How can the network misbehave?
  - What can/can’t we guarantee?

Electronic Marketplace



Replication

Managing replication

- Clock synchronization
  - How do we get clocks to agree as best as we can?

- Event ordering
  - How do we reason about the *relative* order of events across distributed processes?
  - Logical clocks
  - Vector clocks

Managing replication

- Communication and coordination
  - How do distributed replicas talk to each other?
  - Group communication systems
  - Leader election
- Consensus
  - What if things go wrong? How do we still make progress?
  - Byzantine Generals problem
  - Replicated state machines
  - Paxos, Raft
- Consistency and CAP
  - How do we reason about consistency across replicas?
  - CAP theorem – consistency, availability, and partition tolerance

Large-scale distributed systems

- How do we actually apply these principles into practice?
- How do we build a distributed locking and coordination service?
  - ZooKeeper, Chubby
- How do we process petabytes of data?
  - MapReduce, Spark
- How do we store petabytes of data?
  - Google File System, Meta’s Tectonic File System

Large-scale distributed systems

- How do we build data abstractions on top of distributed storage/processing systems?
  - Amazon's Dynamo KV store
  - Cassandra database management system
  - Google's Spanner – globally distributed database

- Other topics
  - Peer-to-peer systems
  - "Serverless" computing
  - Machine learning systems