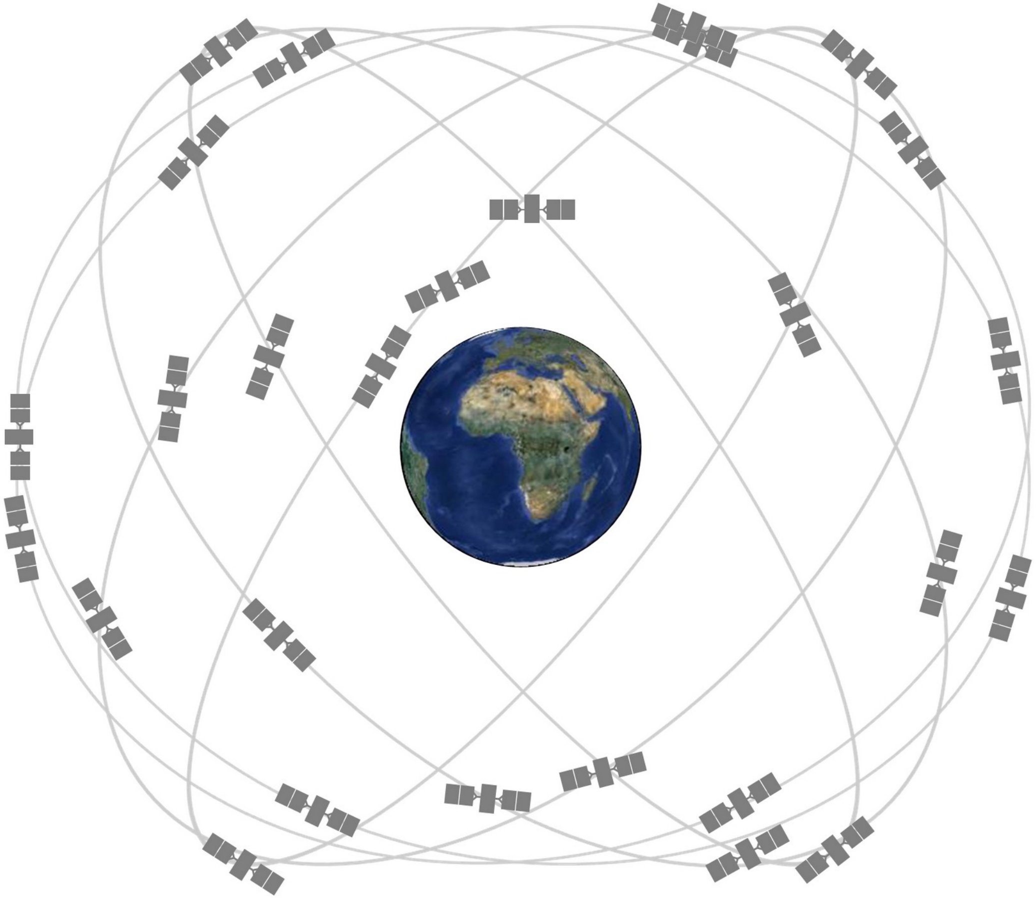
An Introduction to Distributed Systems and Concurrency

CSE 536 Spring 2024

jedimaestro@asu.edu

GPS (Global Positioning System)



GPS facts

- Altitude of satellites is *approx.* 12,550 miles
- Moving about 7,000 miles per hour
  - At the equator, earth spins at about 1,000 miles per hour
- GPS signals reach earth in about 1/15th of a second
  - Going about 670,616,629 miles per hour
    - Every 1000 mph is about 0.000149116% error
      - Who cares?

GPS facts

- Altitude of satellites is *approx.* 12,550 miles
- Moving about 7,000 miles per hour
  - At the equator, earth spins at about 1,000 miles per hour
- GPS signals reach earth in about $1/15^{th}$ of a second
  - Going about 670,616,629 miles per hour
    - Every 1000 mph is about 0.000149116% error
      - Who cares?

0.000149116% of 12,550 miles is about 100 feet!

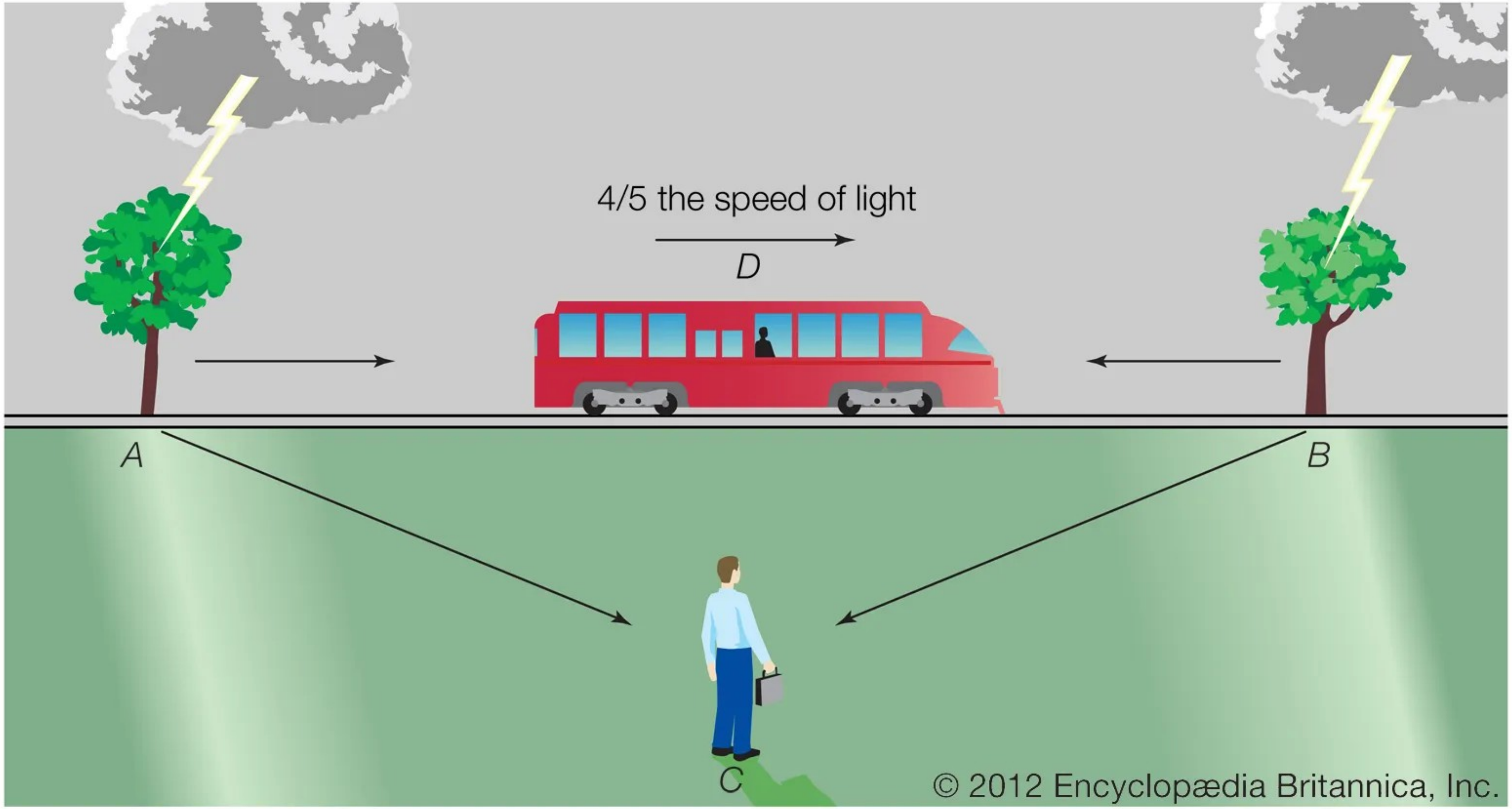
GPS corrections for velocities of satellites and Earth’s spin?

- If we’re spinning 1000mph in one direction and the GPS satellite is going 7000mph in the other direction, *vs.*…
- we’re spinning 1000mph and the satellite is going in the same direction at 7000mph (or orthogonal? an angle?)
- 1000 feet is a lot of error, compared to Coor classrooms 1000 feet away is like Biodesign B
- Do you think the software in your phone accounts for these errors?

GPS corrections

- No correction for the velocities of satellites and the spin of the Earth
  - Einstein's theory of relativity
- Two corrections to the timers aboard the satellites combine for (slowed by a net of 38 microseconds per day)
  - Special relativity → satellites moving faster → time dillation→ time is 7 microseconds per day slower
  - General relativity → satellites farther from Earth's gravity → time is 45 microseconds per day faster

https://www.britannica.com/science/relativity/Special-relativity

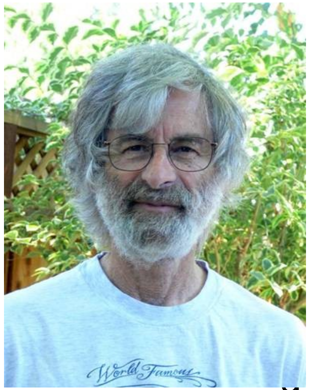


The universe is weird...

- Even the best clocks (*e.g.*, atomic clocks) get out of synch
  - *E.g.*, because of elevation
- Events are only only partially ordered
  - Events are relative to an observer
- Our networks and NIDS systems exist in this weird universe
  - Even if we ignored packet loss and variable network delay, assumed every computer/device had an atomic clock, and accounted for the rough elevation of devices there is no way for a NIDS to know for certain if, *e.g.*, a message was received before a connection timed out on one end of a network flow.

Leslie Lamport

- Microsoft Research
- "winner of the 2013 Turing Award for imposing clear, well-defined coherence on the seemingly chaotic behavior of distributed computing systems"
- Also… LaTeX, Lamport signatures, temporal logic, ...
- https://en.wikipedia.org/wiki/Leslie_Lamport



http://lamport.org/

Communications of the ACM, Volume 21, Issue 7 • July 1978

Operating Systems

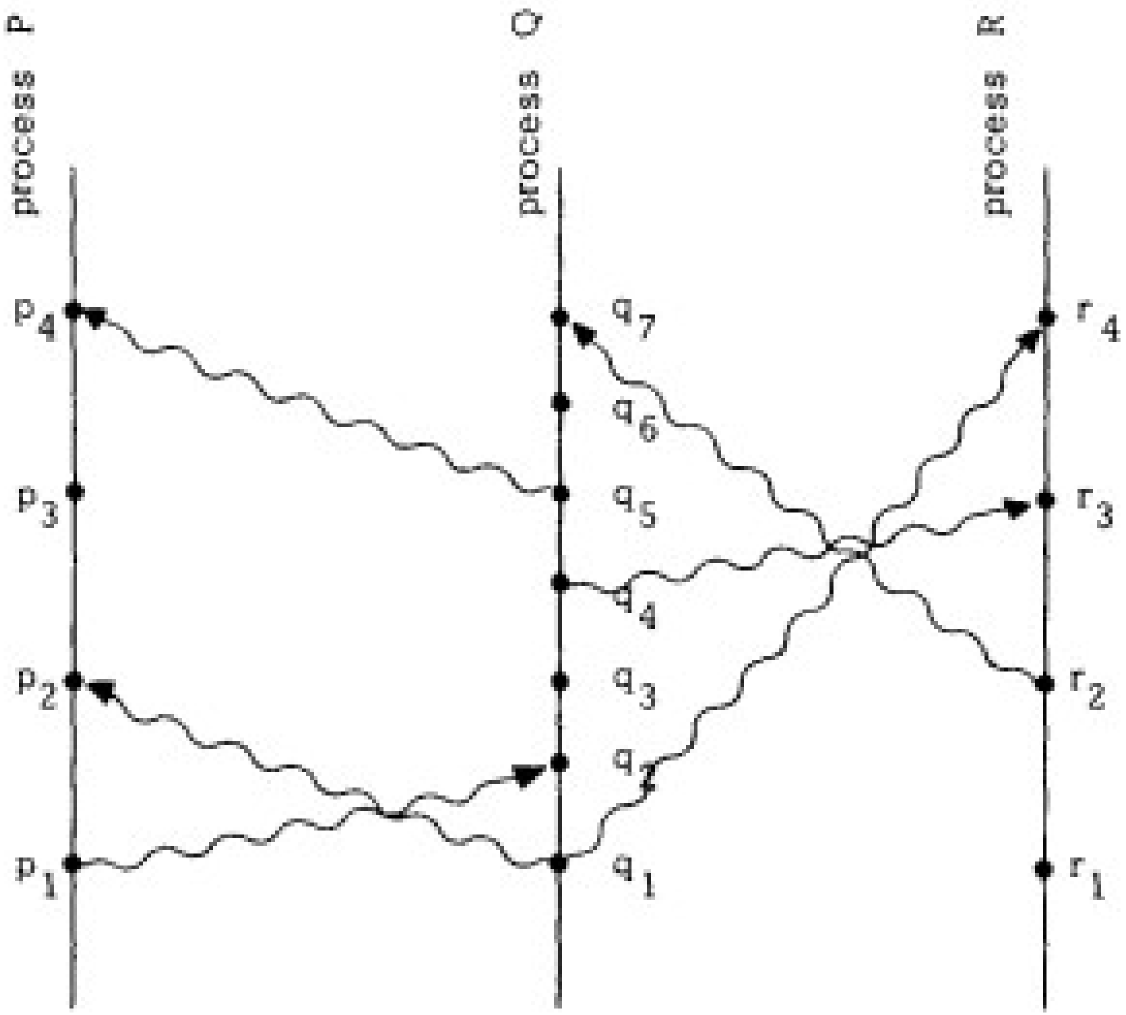
R. Stockton Gaines
Editor

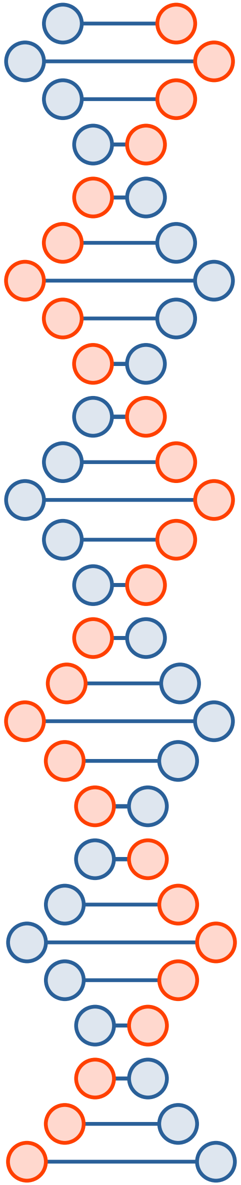
Time, Clocks, and the Ordering of Events in a Distributed System

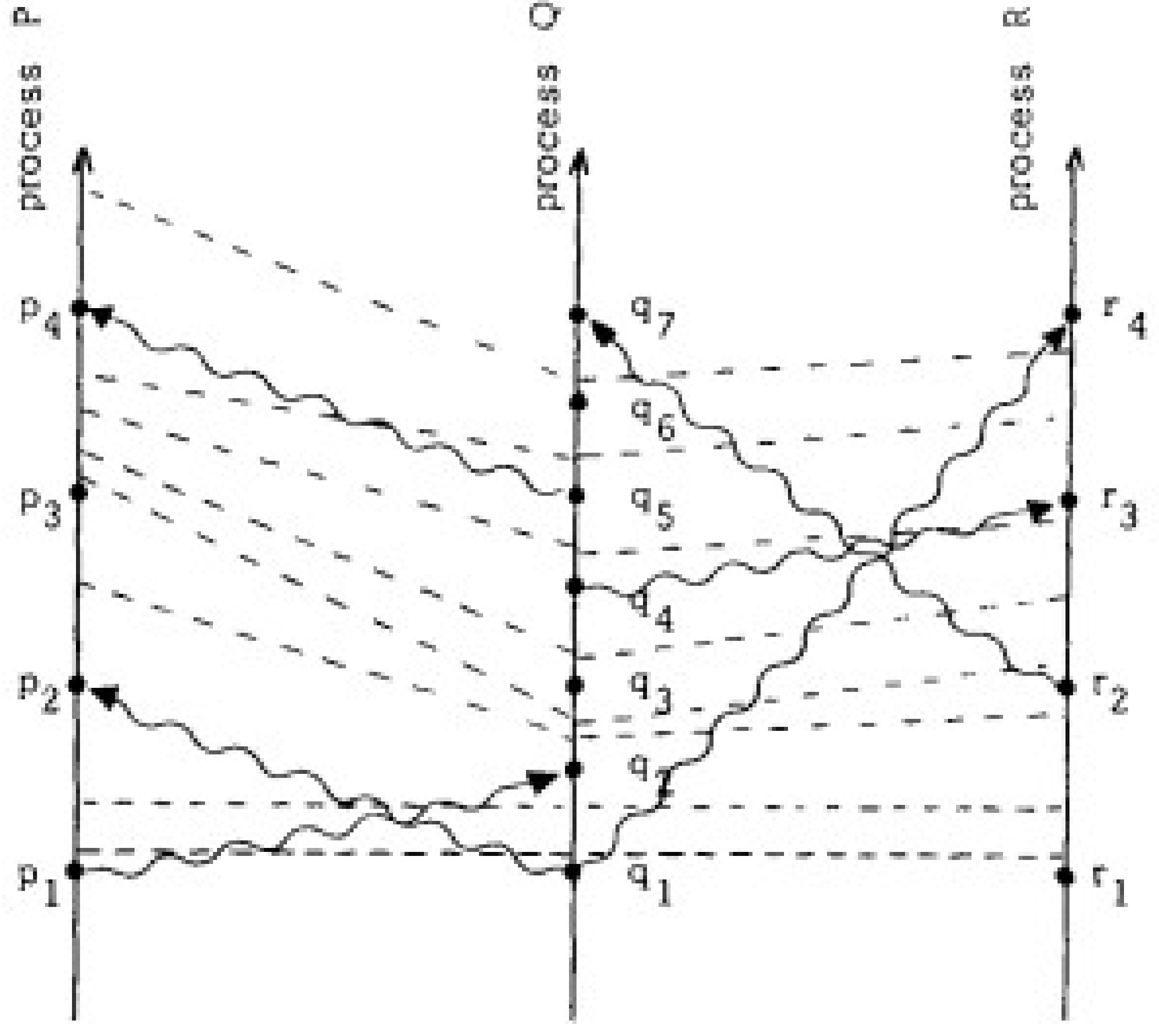
Leslie Lamport
Massachusetts Computer Associates, Inc.

The concept of one event happening before another in a distributed system is examined, and is shown to define a partial ordering of the events. A distributed algorithm is given for synchronizing a system of logical clocks which can be used to totally order the events. The use of the total ordering is illustrated with a method for solving synchronization problems. The algorithm is then specialized for synchronizing physical clocks, and a bound is derived on how far out of synchrony the clocks can become.

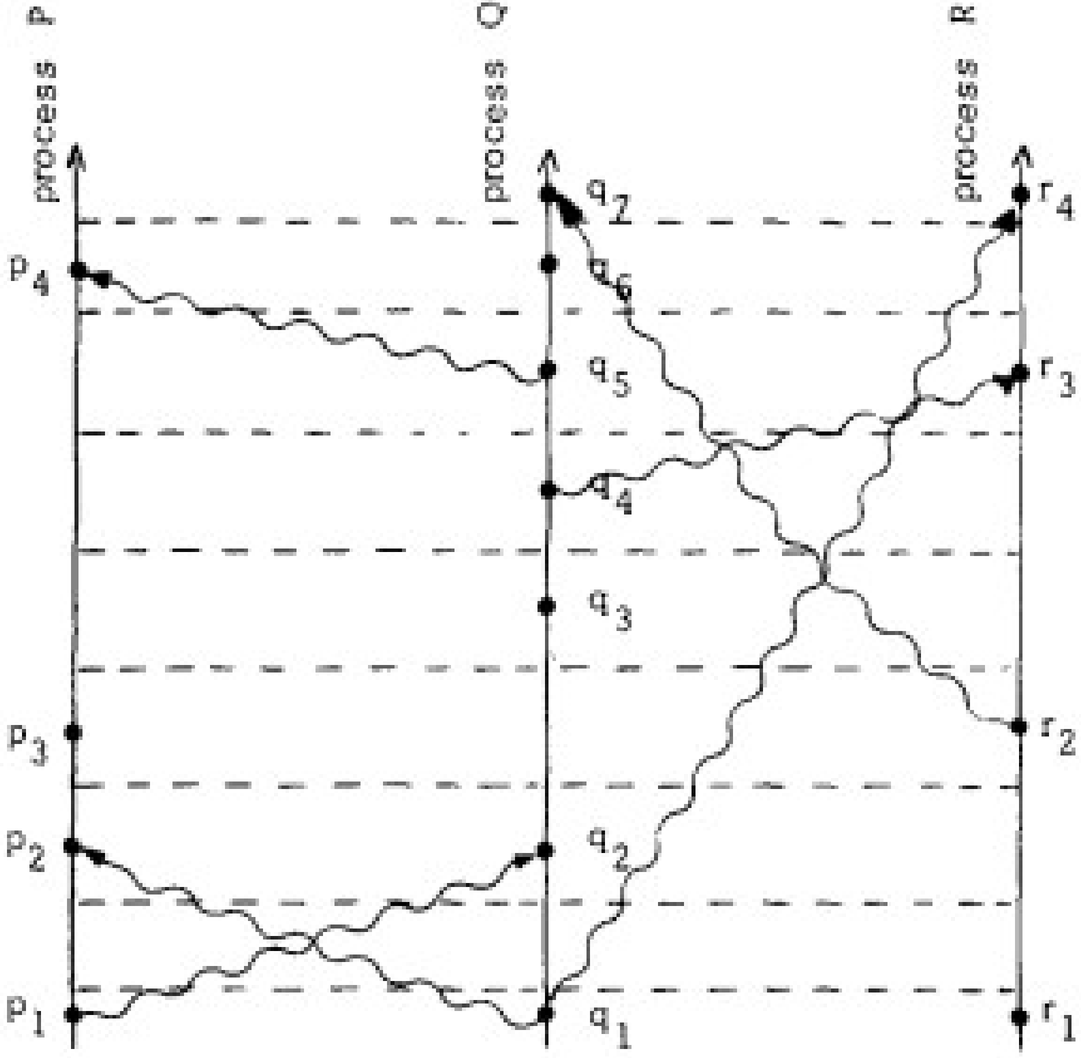
Key Words and Phrases: distributed systems, computer networks, clock synchronization, multiprocess systems


process P
process Q
process R
p4
p3
p2
p1
q7
q6
q5
q4
q3
q2
q1
r4
r3
r2
r1





process P
process Q
process R
p1
p2
p3
p4
q1
q2
q3
q4
q5
q6
q7
r1
r2
r3
r4



process P
process Q
process R
p_4
p_3
p_2
p_1
q_7
q_6
q_5
q_4
q_3
q_2
q_1
r_4
r_3
r_2
r_1

In a distributed system, it is sometimes impossible to say that one of two events occurred first. The relation “happened before” is therefore only a partial ordering of the events in the system. We have found that problems often arise because people are not fully aware of this fact and its implications.

Another way of viewing the definition is to say that $a \rightarrow b$ means that it is possible for event a to causally affect event b. Two events are concurrent if neither can causally affect the other. For example, events p_3 and q_3 of Figure 1 are concurrent. Even though we have drawn the diagram to imply that q_3 occurs at an earlier physical time than p_3, process P cannot know what process Q did at q_3 until it receives the message at p_4. (Before event p_4, P could at most know what Q was *planning* to do at q_3.)

This definition will appear quite natural to the reader familiar with the invariant space-time formulation of special relativity, as described for example in [1] or the first chapter of [2]. In relativity, the ordering of events is defined in terms of messages that *could* be sent. However, we have taken the more pragmatic approach of only considering messages that actually *are* sent. We should be able to determine if a system performed correctly by knowing only those events which *did* occur, without knowing which events *could* have occurred.

"Correctness" is defined as a partial ordering in distributed systems

A protocol must enforce ordering for correctness

What kinds of things can go wrong?

Race conditions (shared memory)

- <u>Thread #1</u>

  x := x + 1

  Move x into Register

  Add 1 to Register

  Move Register into x

- <u>Thread #2</u>

  x := x + 1

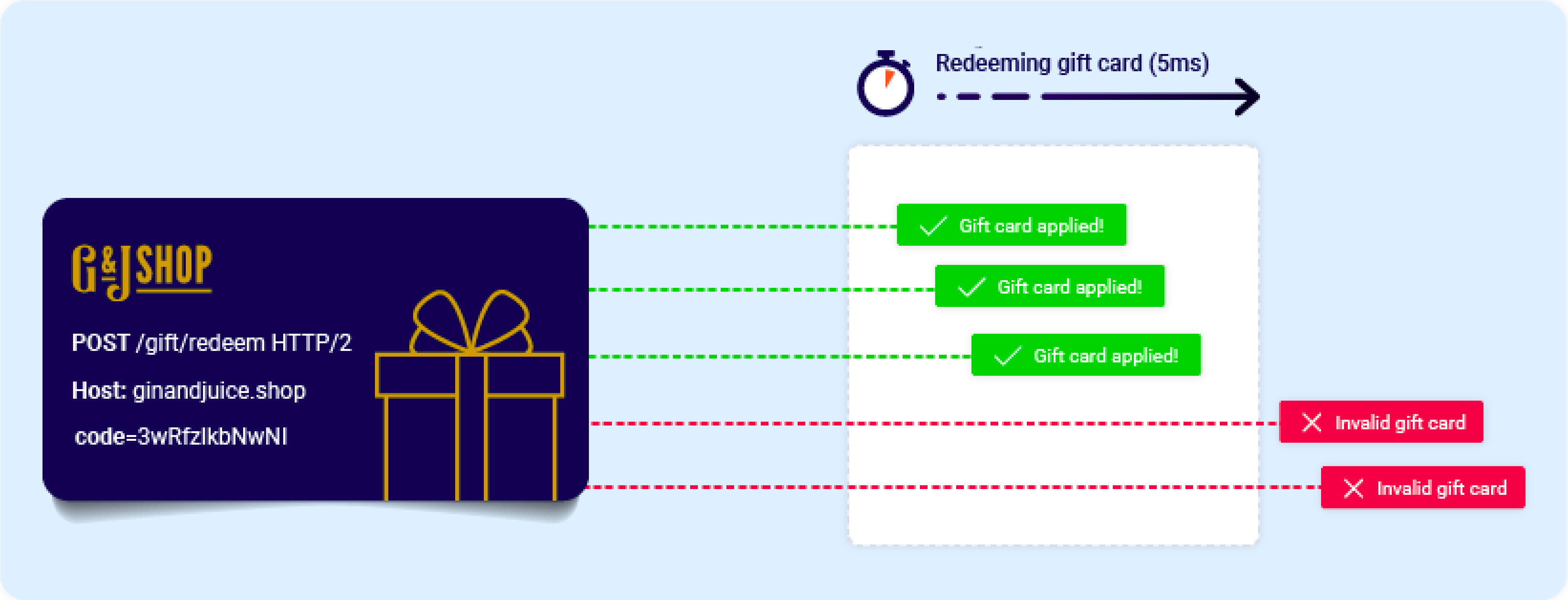
  Move x into Register

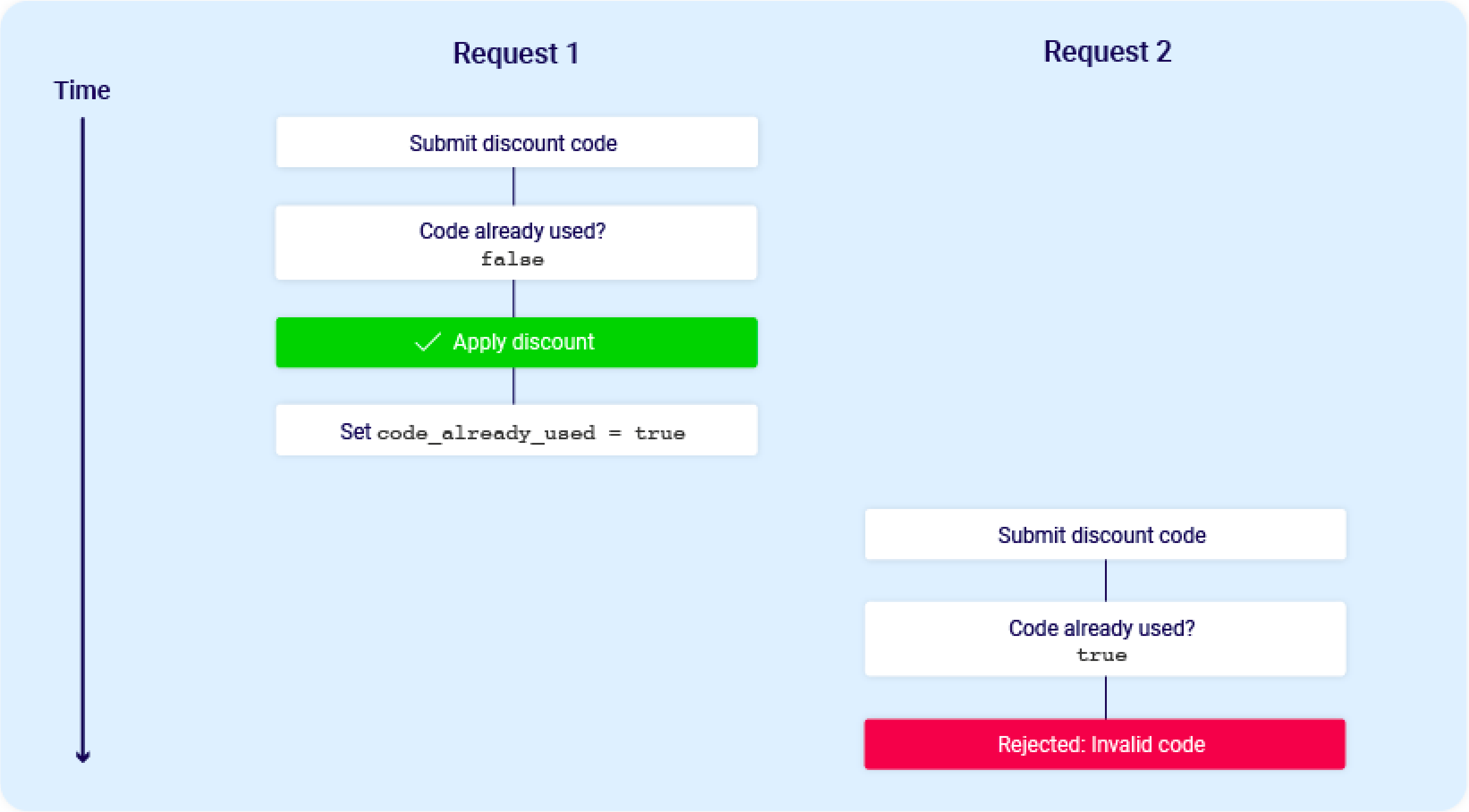
  Add 1 to Register

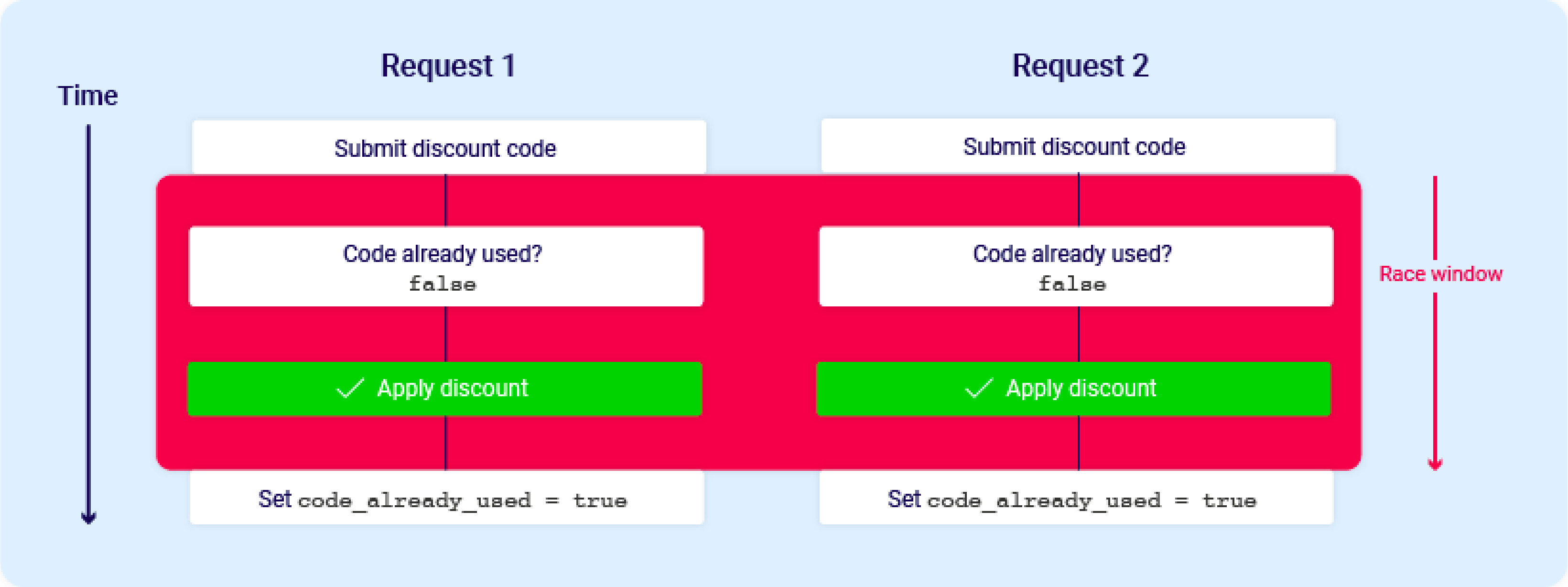
  Move Register into x

An example from a more obviously distributed system…

https://portswigger.net/web-security/race-conditions


Redeeming gift card (5ms)
G&J SHOP
POST /gift/redeem HTTP/2
Host: ginandjuice.shop
code=3wRfzIkbNwNI
Gift card applied!
Gift card applied!
Gift card applied!
Invalid gift card
Invalid gift card


Request 1
Request 2
Time
Submit discount code
Code already used?
false
Apply discount
Set code_already_used = true
Submit discount code
Code already used?
true
Rejected: Invalid code


Request 1
Request 2
Time
Submit discount code
Submit discount code
Code already used?
false
Code already used?
false
Race window
Apply discount
Apply discount
Set code_already_used = true
Set code_already_used = true

The Last Stand 2

94 | 28M | ONLINE SAVE | ZOMBIE | FLASH | DEFENSE | ACTION

Solutions

- Locks (next slide)
- Semaphores (later in the semester)
- Mutual exclusion (later in the semester)
- Transactions (later in the semester)

Locks

- <u>Thread #1</u>

  lock(L)

  x := x + 1

  unlock(L)

  Lock L

  Move x into Register

  Add 1 to Register

  Move Register into x

  Unlock L

- <u>Thread #2</u>

  lock(L)

  x := x + 1

  unlock(L)

  Lock L

  Move x into Register

  Add 1 to Register

  Move Register into x

  Unlock L

Solutions (like locks) introduce their own problems...

- Deadlock
- Priority inversion

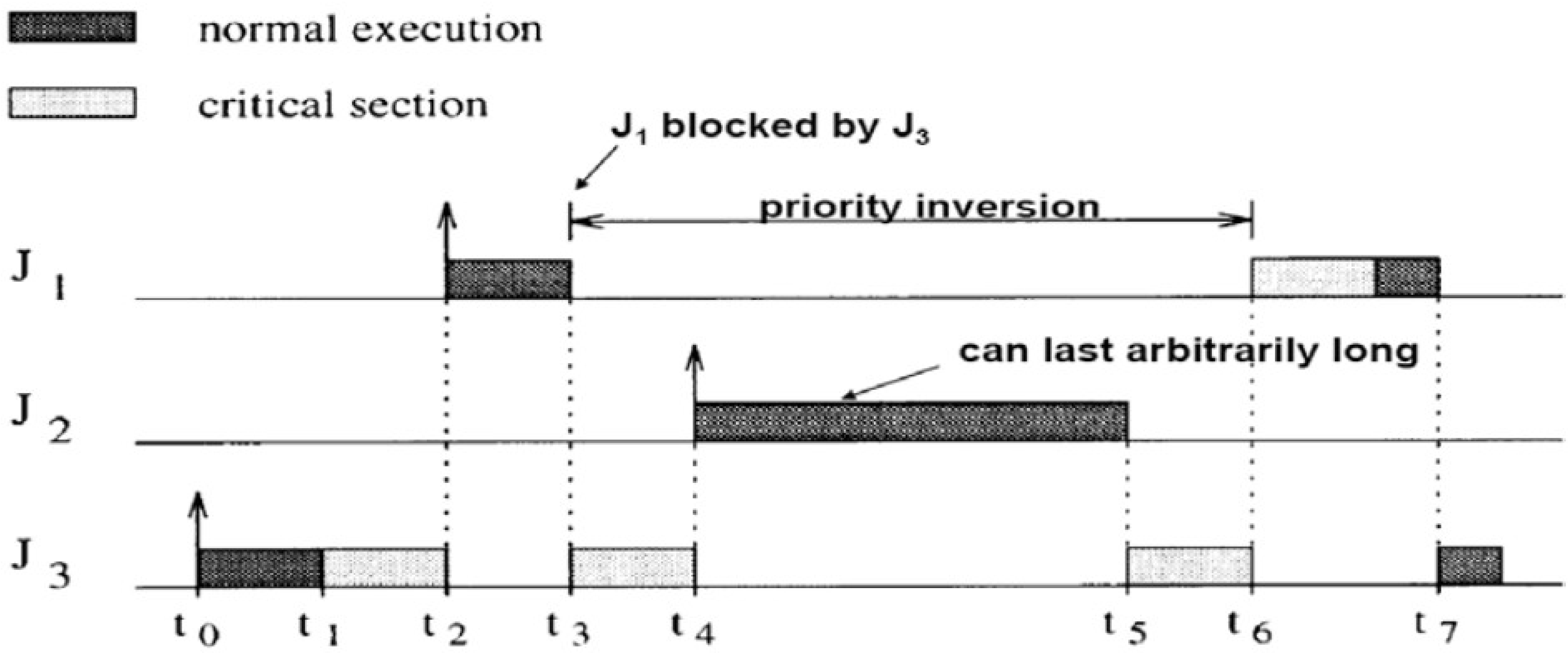
Deadlock

```
lock(L1)
lock(L2)
x := x + 1
unlock(L2)
unlock(L1)
```

```
lock(L2)
lock(L1)
x := x + 1
unlock(L1)
unlock(L2)
```

Priority inversion

https://www.cse.chalmers.se/~risat/Report_MarsPathFinder.pdf


normal execution
critical section
J1 blocked by J3
priority inversion
J 1
can last arbitrarily long
J 2
J 3
t 0
t 1
t 2
t 3
t 4
t 5
t 6
t 7