EECS 591
Distributed Systems

Manos Kapritsos
Fall 2021
CONSISTENCY

Is the server’s response correct?

(are all the server’s responses consistent with each other?)
Consistency is a property of the execution; a constraint on the values of the reads and writes returned by the server.
Causal consistency

All processes see causally related events in the same order.

A student removes advisor from friends list and then posts Spring Break photos

The advisor should not be able to see the pictures
**Serializability**

A *concurrent* execution of transactions is equivalent to one that executes the transactions serially in *some sequential order*.

Are these runs serializable?

1) \( T_1 : W(x,3) \)
   \( T_2 : W(x,5) \)
   \( T_3 : R(x)=3 \)

2) \( T_1 : W(x,3) \)
   \( T_2 : [W(x,5), R(x)=3] \)
LINEARIZABILITY

Same as serializability, but the sequential order must preserve the **real-time** constraints of non-overlapping operations.
**Administrivia**

**Implementation project**
- Going out after class
- Groups of 2 (no need to declare)

**Deadlines for the coming month**
- Declare project topic: 10/8
- Problem set #2: 10/11
- Midterm exam: **moved to 10/27**
- Implementation project: 10/25
- Presentation slides: 11/2
## Consensus

| **Validity** | If all processes that propose a value propose \( v \), then all correct processes eventually decide \( v \) |
| **Agreement** | If a correct process decides \( v \), then all correct processes eventually decide \( v \) |
| **Integrity** | Every correct process decides at most one value, and if it decides \( v \), then some process must have proposed \( v \) |
| **Termination** | Every correct process eventually decides some value |
THE ALGORITHM

Process $p_i$:

Initially $V = \{v_i\}$

To execute $\text{propose}(v_i)$:

round $k, 1 \leq k \leq f + 1$

1. Send $\{v \in V: p_i \text{ has not already sent } v\}$ to all
2. for all $j, 0 \leq j \leq n + 1, j \neq i$, do
3. receive $S_j$ from $p_j$
4. $V := V \cup S_j$

$\text{decide}(\ )$ occurs as follows:

5. if $k = f + 1$
6. decide $\min(V)$
Good news

Our algorithm implementing consensus in a synchronous setting is correct! That is, it is both safe and live.
BAD NEWS

The FLP result:
There is no protocol that solves consensus in an asynchronous system where one process may crash

Fischer, Lynch, Paterson 1985
The intuition

In an asynchronous setting, a process cannot tell the difference between a crashed process and one whose messages take long to arrive.

How long should the process wait before deciding?

- It can’t wait forever: that would violate liveness.
- If it gives up on a process, but it turns out that process is just slow, that would violate safety.
GETTING AROUND THE IMPOSSIBILITY RESULT OF FLP

You can’t be both safe and live in the presence of asynchrony

The FLP result

Fine, then I’ll just be safe! I will only be live when the network behaves synchronously.
ENTER PAXOS
Abstract

The Paxos algorithm, when presented in plain English, is very simple.