The primary cannot respond until it has received all acks for prior updates.
Chain replication

Primary

Head  $f+1$ replicas  Tail
Chain replication

update

Head

$ f + 1 $ replicas

Tail

query

reply
Chain replication

Head

update

$\mathit{f + 1}$ replicas

Tail
Chain replication

Head

$f + 1$ replicas

Tail

update
Chain replication

Head \rightarrow f + 1 \text{ replicas} \rightarrow \text{Tail}

update \rightarrow \text{reply}
Chain replication

Tail can respond immediately, without waiting for the new update.
CONSISTENCY

Is the server’s response correct?

(are all the server’s responses consistent with each other?)
**Consistency**

Consistency is a **property** of the execution; a constraint on the values of the reads and writes returned by the server.
Monotonic read consistency

If a client reads the value of a data item \( x \), any successive read operation on \( x \) by that client will always return that same value or a more recent value.

Are these runs monotonic read consistent?

\[
W_1(x,3) \quad R_1(x)=4 \quad W_2(x,4) \quad R_2(x)=4
\]

\[
R_1(x)=1 \quad R_1(y)=1 \quad W_2(y,4) \quad R_1(x)=4
\]

.... \[
R_1(x)=1 \quad R_1(y)=1 \quad W_2(y,4) \quad R_1(x)=4
\]
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
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.

1)  W(x,3)  \[\rightarrow\]  W(x,5)  \[\rightarrow\]  R(x)=3  \[\rightarrow\]  S

2)  W(x,3)  \[\rightarrow\]  W(x,5)  \[\rightarrow\]  R(x)=3  \[\rightarrow\]  S
ADMINISTRIVIA

Problem sets
• PS2 is out; due on 10/12, before class
  • Individual work only
• PS1 grades are out. Graded papers will go out soon™.

Presentations
• Presentation assignments posted on Piazza
• Presentation schedule posted on course webpage

Research project
• Group declaration due tomorrow
• Topic declarations due next Thursday 10/8
Presentations

First, you should always make a script for your presentation, before you start making slides. This helps you organize your thoughts and present them clearly to your audience. The script should be at the high level, a kind of summary of the presentation with about one or two sentences per slide. Also, you should avoid having lots of text on one slide, as this is guaranteed to put your audience to a deep, dreamless slumber. Where most presentations fail is that their authors, convinced they are producing some kind of stand-alone document, put everything they want to say onto their slides, in great big chunky blocks of text. While speaking, your voice should not be a flat monotonic drone, but you should try to change inflection often, so as to avoid putting your audience to sleep. And, of course, you should never try to read aloud the text written in your slides. If you find yourself doing that during your practice talks, it means there’s something wrong with the presentation. Unless of course you are trying to make a point, as I am doing right now :)
PRESENTATIONS (FOR REAL THIS TIME)

• Motivation, motivation, motivation!
• Keep it simple
  • Give the high-level intuition
• Avoid the “wall of text”
• Speak normally, with changes to your inflection
• Practice, practice, practice!
# Consensus

<table>
<thead>
<tr>
<th>Property</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Validity</strong></td>
<td>If all processes that propose a value propose ( v ), then all correct processes eventually decide ( v )</td>
</tr>
<tr>
<td><strong>Agreement</strong></td>
<td>If a correct process decides ( v ), then all correct processes eventually decide ( v )</td>
</tr>
<tr>
<td><strong>Integrity</strong></td>
<td>Every correct process decides at most one value, and if it decides ( v ), then some process must have proposed ( v )</td>
</tr>
<tr>
<td><strong>Termination</strong></td>
<td>Every correct process eventually decides some value</td>
</tr>
</tbody>
</table>
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. $V := V \cup S_j$

$\text{decide}(\cdot)$ 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
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.
<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proposers</td>
<td>A proposer is a process that has a value to propose</td>
</tr>
<tr>
<td>Acceptors</td>
<td>Acceptors are the processes that ultimately choose which proposed value will be decided</td>
</tr>
<tr>
<td>Learners</td>
<td>A learner only cares about learning which value was decided</td>
</tr>
</tbody>
</table>

(2f+1)
How the game is played

- **Election:** Proposers first try to get a majority of acceptors to follow them.

- **Legislating:** After acquiring a majority, a proposer can *try* to enforce her value, by getting acceptors to accept it, but…

- **Playing nice:** If an elected proposer finds that some previous value has been proposed, she proposes that value instead.

- **Winning the game:** Once a majority of acceptors have accepted a value, the value is *chosen/decided*
Greetings, peasants! I am your fearless leader! Grant me your blessing!

My first decree is: The value should be 12

We are with you, oh wise leader!

Sounds good to me!
HOW IT IS SUPPOSED TO WORK

Proposer

Acceptors

IAmLeader YouAreLeader Decree

Learner

Accept
Dealing with multiple proposers

Greetings, peasants! I am your fearless leader! Grant me your blessing!

We are with you, oh mighty leader!

We are with you, oh wise leader!

Greetings, peasants! I am your fearless leader #2! Grant me your blessing!

We are with you, oh wise leader #2!
DEALING WITH MULTIPLE PROPOSERS

- I swear I won’t follow an earlier leader!
- And, btw, here is my current accepted value (if any) by leader x.

Proposer

Acceptors

IAmLeader #1  YouAreLeader  Decree

Learner

Accept
I swear I won’t follow an earlier leader!
And, btw, here is my current accepted value (if any) by leader x.
DEALING WITH MULTIPLE PROPOSERS

- I swear I won’t follow an earlier leader!
- And, btw, here is my current accepted value (if any) by leader x.

Proposer #1

IAmLeader #1  YouAreLeader  Decree

Acceptors

Proposer #2

IAmLeader #2  YouAreLeader
THE CRUCIAL YouAreLeader MESSAGE

Proposer #1

Acceptors

I swear I won’t follow an earlier leader!
And, btw, here is my current accepted value (if any) by leader x.

1. Wait for a majority of YouAreLeader messages before proceeding.
2. If none of them contain a previously accepted value, propose your own. Otherwise, propose the value of the most recent leader.
THE CRUCIAL YouAreLeader MESSAGE

1. Wait for a majority of YouAreLeader messages before proceeding.
2. If none of them contain a previously accepted value, propose your own. Otherwise, propose the value of the most recent leader.

Important
By consulting a majority, the new leader makes sure she cannot have missed a chosen value

(a value must be accepted by a majority to be chosen, and any two majorities overlap!)