Consensus:

Impossible
or
Not?

Admin: Midterm Next Week

→ During Class.

→ Open Book: Feel free to bring papers, notes, etc.

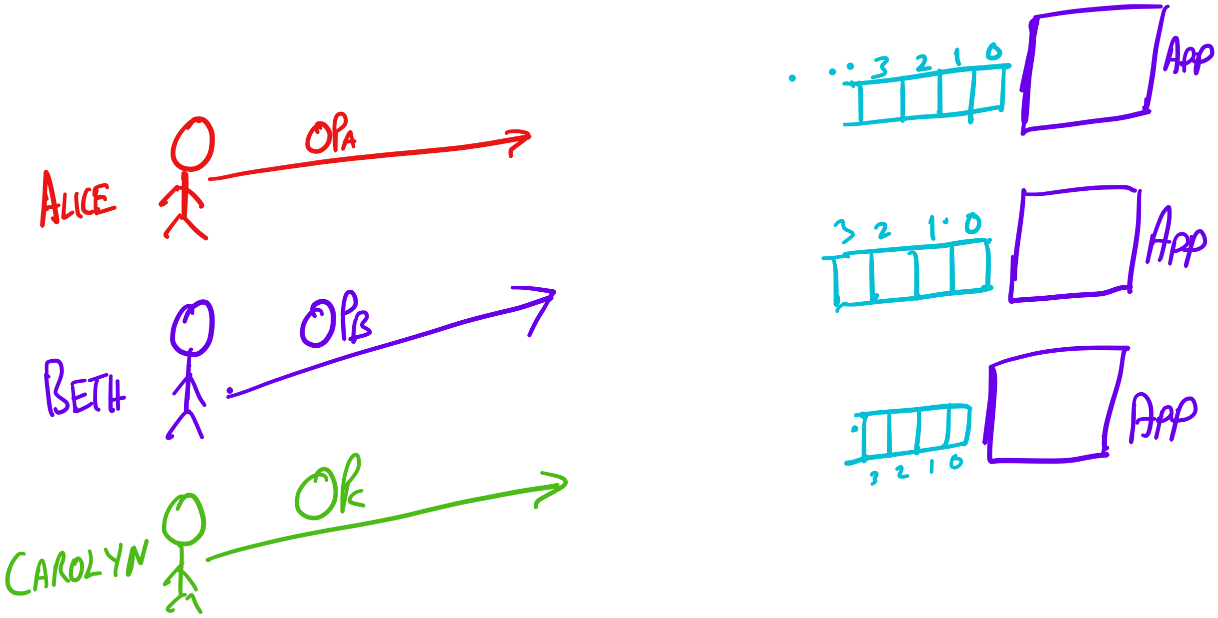
→ Focus on testing understanding of the papers & anything additional concepts covered in class
(please make sure you know what the asynchronous model entails)

→ Questions in the pre-class notes are good examples of what might be asked

Recap

Want To Build A Fault Tolerant Application

↳ Is A State Machine



Requirement

→ All Logs Agree On Order Of Operations

$\Uparrow$

All Logs $\boxed{\text{Agree}}$ On Operation At Idx 0, 1, 2, ...

↳ Consensus Algorithms

Core-Result

No Deterministic (1) Fault-Tolerant (2) Consensus Protocol (3)

For The Asynchronous Setting

4

① Deterministic What?

Where used?

② Fault Tolerant

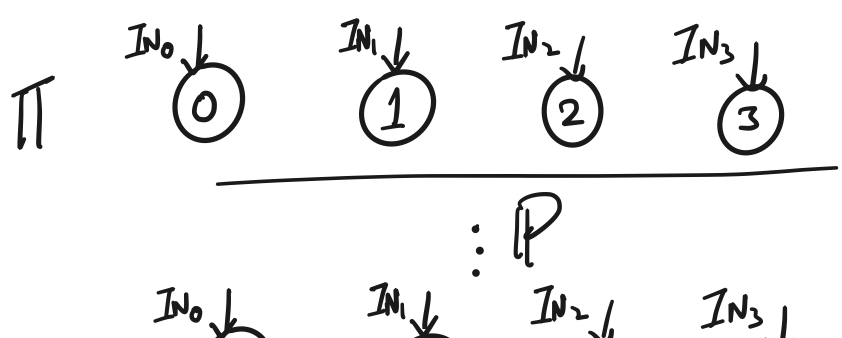
- Want to consider as many protocols as possible

↳ Choose a weak failure model

At most 1 process fails

Fail-stop failures

③ Consensus Protocol



0 → Commit(X_0)   1 → $\perp$ 2 → Commit(X_2)   3 → $\perp$

Requirements

Agreement: If process $\underbrace{P_i \text{ Decides } X_i}_{\text{Outputs Commit}(x_i)}$ & P_j Decides X_j

Then $X_i = X_j$

Validity: If Process P_i Decides X_i then $\exists j$. s.t. $\underline{In_j} = X_i$

(Equivalent To Strong Unanimity)

Termination: All Processes Eventually Decide

Terminology

$C_0 \xrightarrow[e_i]{\text{receive}(P_1, m)} C_1 \xrightarrow[e_2]{\text{receive}(P_1, m)}$ $e_2(C_0)/_2$ $e_1(C_0)/_1$

C_0: $\boxed{0 \quad 1 \quad 2 \quad 3}$ **Configuration:** State Of All Processes

Events/Steps: $\underbrace{\text{receive}(P, m)}$ Receive message $\boxed{m}$ at process p

Changes the state of process p.

$\boxed{\text{m can be } \emptyset \text{ the empty message}}$

e = receive(P_1, m)

$C_0 \xrightarrow{e} C_1$ Must be ENABLED In C_0

??

Schedule (σ): Sequence of steps.

e_0 Disjoint From e_1 : e_0 is a receive for a different process than e_1

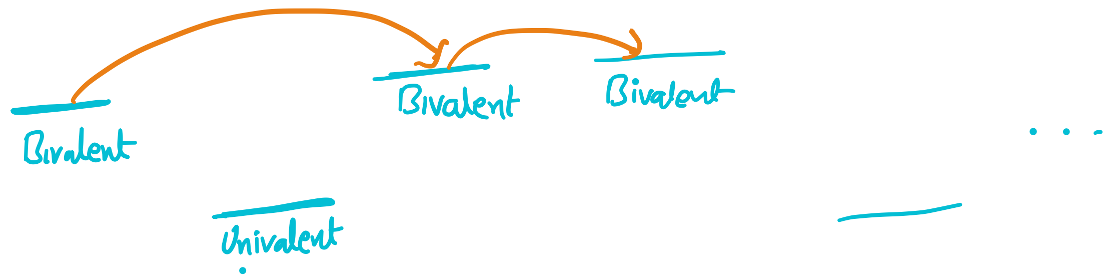
σ_0 Disjoint From σ_1

No Deterministic Fault-Tolerant Consensus Protocol For The Asynchronous Setting

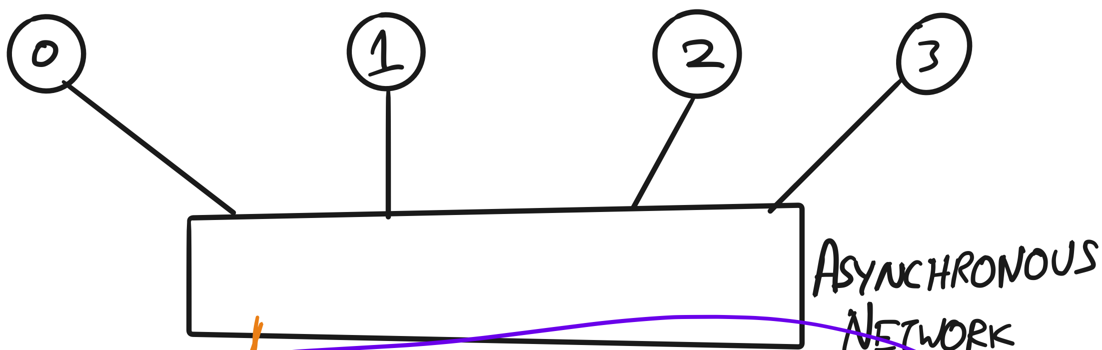
Overall Idea:

1. Focus On Binary Consensus [Inputs & Outputs Are 0 or 1]
   - Assume A Det, FT Consensus Protocol Exists $\mathbb{P}$
2. Show That $\exists$ Configurations For Which $\mathbb{P}$ Can $\underbrace{\text{Decide Both}}_{\exists \sigma_0 \ldots,\ \sigma_1 \ldots}$ 0 Or 1 [Bivalent Configs]
3. Show That $\exists$ $\underbrace{\text{Initial Configurations}}_{\text{(Inputs)}}$ That Are Bivalent

④ Show That One Can Always Find A Schedule That Goes From Bivalent C To Another Bivalent C



"Construct A Schedule"



↳ Delay + Reorder Message

Why Does This Work?

P Is 1-fault tolerant

But works in async networks

↳ Cannot Distinguish B/W Delayed Messages & Failed Process

Must eventually suspect quiet processes have failed

Also must deal with quiet processes sending messages

Diamond Lemma

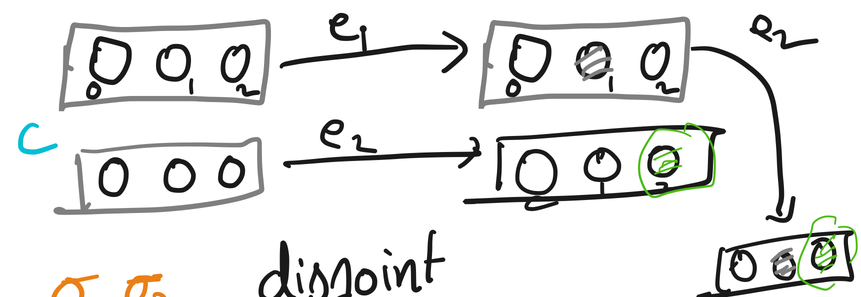
$P_1 = \cancel{0}\ 1$

$P_2 = 2$

$$
\begin{tikzcd}
 & C \arrow[dl, "e_1"'] \arrow[dr, "e_2"] & \\
C_1 \arrow[dr, "e_2"'] & & C_2 \arrow[dl, "e_1"] \\
 & C' &
\end{tikzcd}
$$

e_1, e_2 <u>disjoint</u> $\Rightarrow$ $e_1 := recv(P_1, m)$, $e_2 := recv(P_2, m')$

e_1, e_2 enabled in C_1 $\quad P_1 \neq P_2$

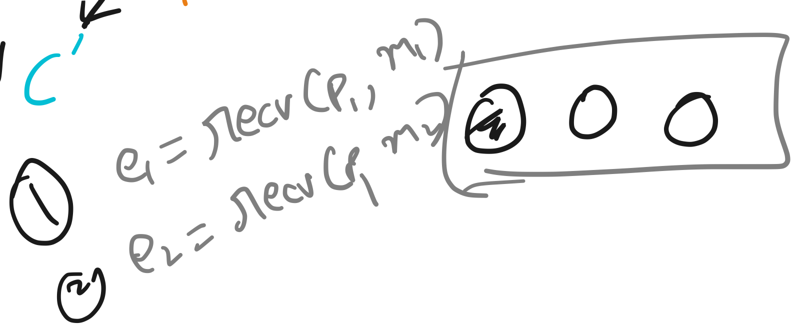


$$
\begin{tikzcd}
 & C \arrow[dl, "\sigma_1"'] \arrow[dr, "\sigma_2"] & \\
C_1 \arrow[dr, "\sigma_2"'] & & C_2 \arrow[dl, "\sigma_1"] \\
 & C' &
\end{tikzcd}
$$

σ_1, σ_2 disjoint

<u>enabled</u> in C

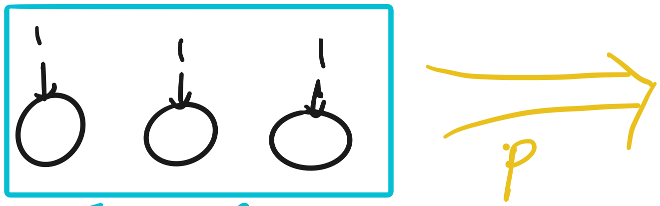
??



Valence

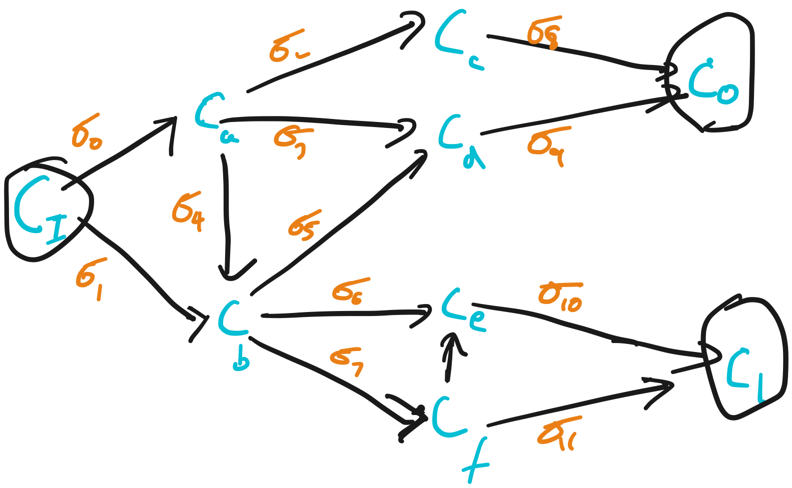
Choose 0 / Decide 0

INITIAL CONFIG



Choose 1 / Decide 1

INITIAL CONFIG



BIVALENT INITIAL CONFIGURATION

INPUT	OUTPUT
0 0 0	0
0 0 1	0
0 1 0	0
1 0 0	0
1 1 0	1
1 0 1	0
0 1 1	1
1 1 1	1

Observation

P is 1-Fault tolerant

0 0 $\boxed{0}$ 1 1 $\boxed{0}$
0 0 $\boxed{1}$ 1 1 $\boxed{1}$

1 ← $\boxed{1}$ 0 1 $\boxed{0}$ 1 1 → 0 ?
0 ← $\boxed{0}$ 0 1 $\boxed{1}$ 1 1 → 1 1

Must decide even for σ where a process takes no step

Aside: Fairness?

1. Assume $\mathbb{P}$ ✓
2. $\exists$ Initial Bivalent Config ✓
3. Show $\exists\ \sigma$ going from Bivalent Config to Bivalent Config
   - Current Bivalent Config C_0
   - Pick enabled event e
     - $e(C_0)$ is bivalent ← Done
     - $e(C_0)$ is not bivalent
       - Find enabled σ s.t.

$$C_0 \xrightarrow{\sigma} C_1 \xrightarrow{e} \underbrace{C_e}_{\text{Bivalent}}$$

Finding is hard (Not really — constructive proof of FLP)

Instead show existence.

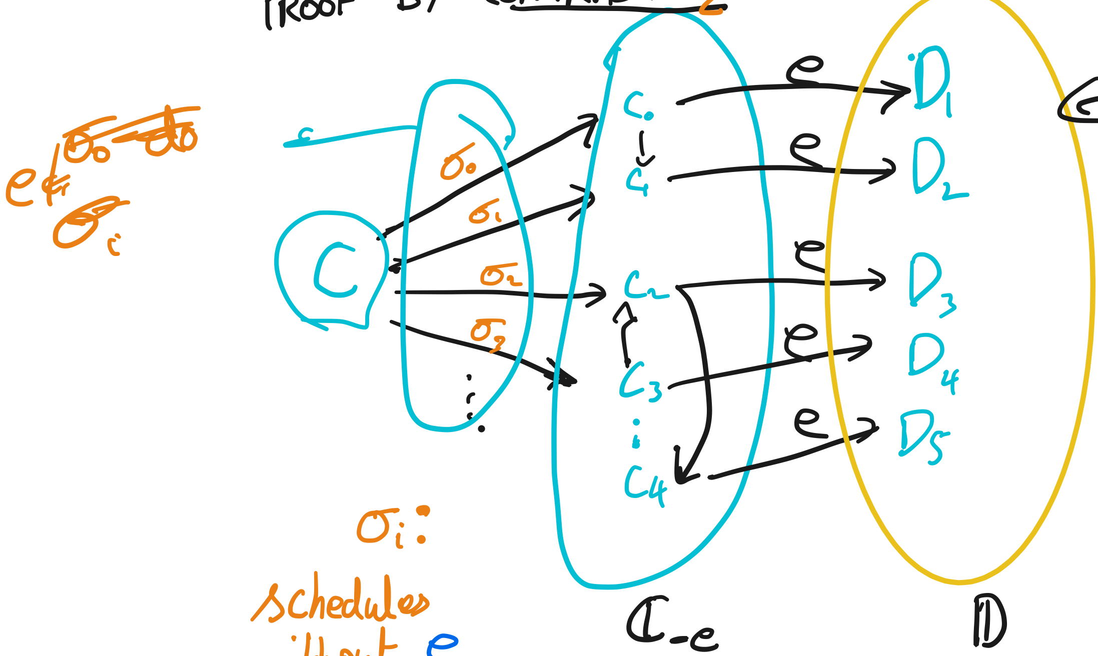
→ Given $\boxed{C}$ bivalent, enabled e, $\exists$ enabled σ

s.t. $e(\sigma(C))$ bivalent

PROOF BY CONTRADICTION ← No such σ exists

$\Downarrow$

All $\mathbb{D}$ are 0- or 1-valent



σ_i: schedules without e

What does $\mathbb{D}$ contain?

Must contain at least one 0-valent & one 1-valent configuration.

Why

C is bivalent

$\Rightarrow \exists$ enabled σ_0 s.t.

$C \xrightarrow{\sigma_0} C_0$ ← 0-valent

If $e \notin \sigma_0$ then $e(C_0) \in \mathbb{D}$

0-valent

If $e \in \sigma_0$ then

$\sigma_0 = \underbrace{e_0; e_1; e_2;}_{\sigma_{-e}}\ e;\ \underbrace{\ldots}_{\sigma_{+e}}$

Imagine same thing with

$C \xrightarrow{\sigma_1} C_1$

$\uparrow$ 1-valent

But

$e(\sigma_{-e}(C)) \in \mathcal{C}_{-e}$

↳ Can be 0-valent or $\boxed{\text{bivalent}}$ [Disallowed by assumption that $\mathcal{C}_{-e}$ has no bivalent configurations]

(4c) contains both 0- & 1-valent configurations

$\mathcal{C}_{-e}$ contains configurations C_A, C_B that are

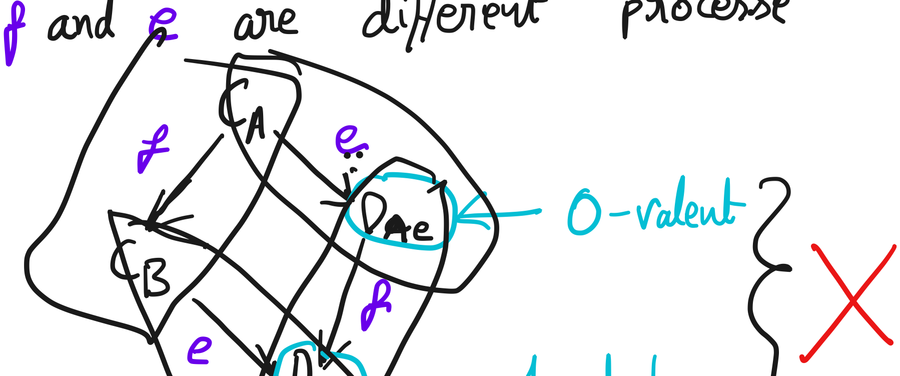
(a) Neighbors $C_A \xrightarrow{f} C_B$

(b) $e(C_A)$ is 0-valent, $e(C_B)$ is 1-valent.

How? Hand-wavy but use initial bivalent argument.

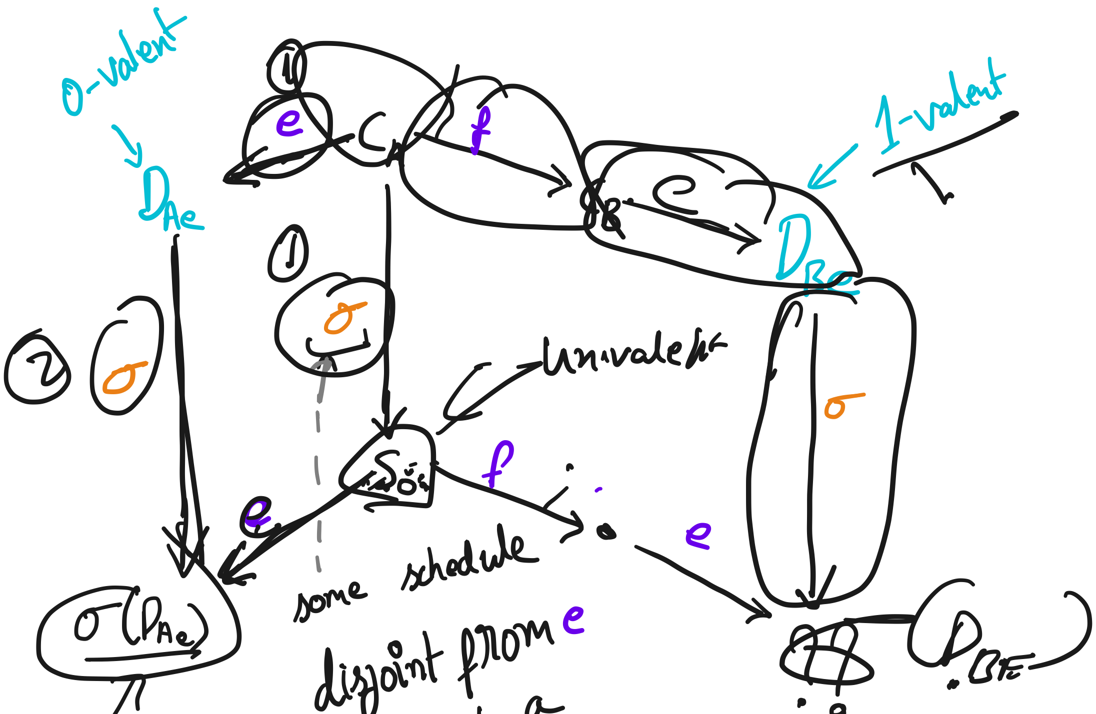
Finally: arrive at contradiction: no such f can exist

(1) If f and e are different processe



② e AND f are not disjoint

$\underbrace{e, f}_{P}$



P terminate

Must be 0-decided

disjoint from e
↳ leads to a decision

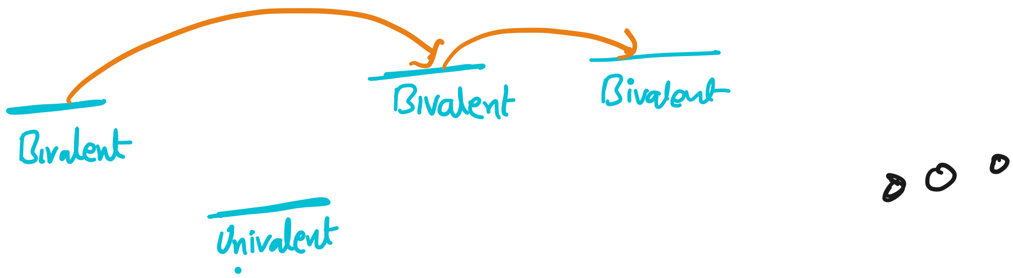
[σ disjoint from e
⇒ σ disjoint from f]

CONTRADICTION

FOUND ENABLED σ S.T.

$$C_0 \xrightarrow{\sigma} C_1 \xrightarrow{e} \underline{C_e}$$

Bivalent

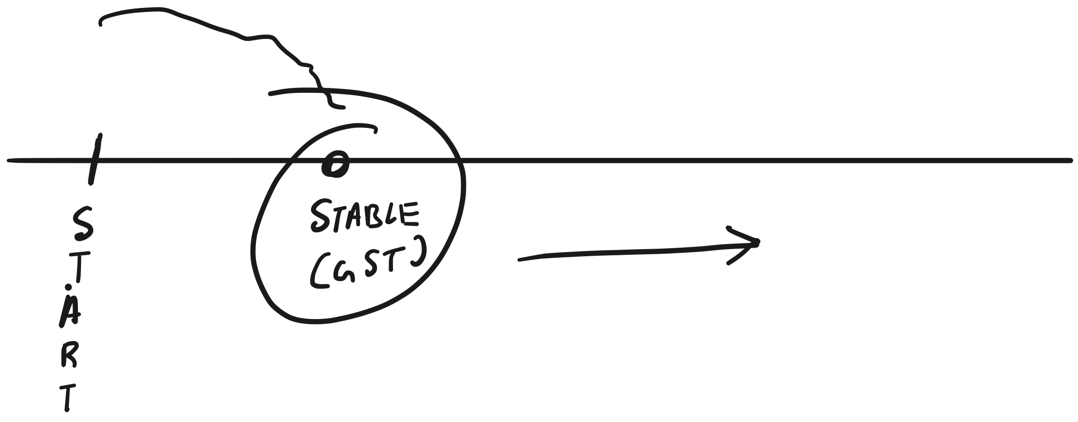


No DETERMINISTIC FAULT-TOLERANT CONSENSUS PROTOCOL FOR THE ASYNCHRONOUS SETTING

What happens if we relax assumptions?

1. Deterministic?

2. PARTIAL SYNCHRONY


START
STABLE
(GST)