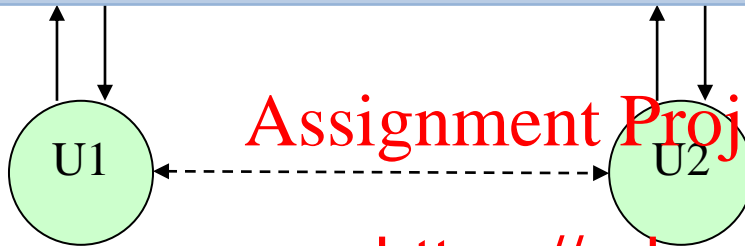


SYNCHRONISERS: ATTEMPT TO SYNCHRONISE ASYNC NETWORKS (SEE LYNCH CH 16)**DO NOT COPE WITH STOPPING (OR, EVEN WORSER, WITH BYZ) FAILURES****GlobSync (Global Synchronizer)**

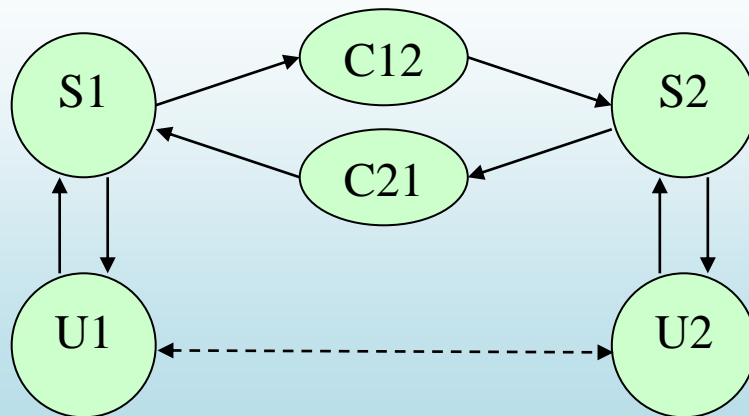
- ensures that *all* messages are sent before
- collects *all* sent messages and then delivers them to their targets

**LocSync (Local Synchronizer)**

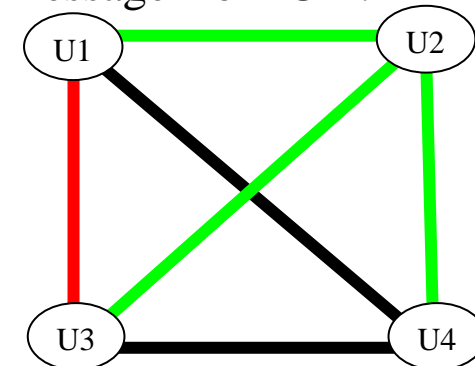
- similar to GlobSync
- but only ensures that *all neighborhood* messages are sent before receiving

SimpleSync (Simple Synchronizer)

an implementation of the LocSync



- Synchronizers need to keep the round number, explicitly or implicitly.
- If no faulty nodes are present, messages from different rounds may be interleaved (**fast**, **slow**).
- Process U3 may receive a round 2 message from U2, before receiving its round 1 message from U1 !



Assignment Project Exam Help

SYNCHRONISERS: CAN THEY HELP?

Add WeChat edu_assist_pro

- Using a simpler sync algorithm in more complex async !
 - E.g. Sync Distributed MST instead of the Async Distributed MST (GHS)

Assignment Project Exam Help

- Obtaining a faster runtime
 - E.g. could this avoid being costly in the end ?
- What if we apply a synchroniser to Distributed ?

COMMUNICATION FAILURES

Assignment Project Exam Help

Is it possible to reach a distributed agreement with (unbounded) communication failures?

Add WeChat edu_assist_pro

Formal conditions.

- **Termination:**
 - All processes eventually decide.
- **Agreement:**
 - No two processes ever decide on different values.
- **Validity:**
 - If all processes start with all messages are properly delivered, then 1 is the only possible decision value. [WEAK]
 - relaxation: allow fallback on 0 if communications fail (approx. v0 could be 0)
 - and
 - If all processes start with the same initial value 0, then 0 is the only one possible decision value. [STRONG]

We also assume that a decision once taken can be immediately acted upon (by *all processes in sync*).

Diagrams

	Initial	Decision
No comm fault	$\forall 0$	$\forall 0$
Comm faults	$\forall 0$	$\forall 0$

	Initial	Decision
No comm fault	$\forall 1$	$\forall 1$
Comm faults	$\forall 1$	$\forall 1 \text{ or } \forall 0$

	Initial	Decision
No comm fault	$\exists 0, \exists 1$	$\forall 1 \text{ or } \forall 0$
Comm faults	$\exists 0, \exists 1$	$\forall 1 \text{ or } \forall 0$

Fundamental result: Even under such relaxed conditions, no deterministic agreement is possible if unlimited communication failures are possible (but then, how do TCP and the internet work?)

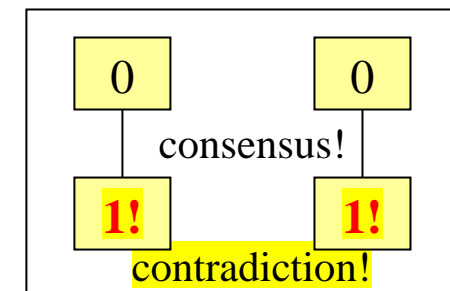
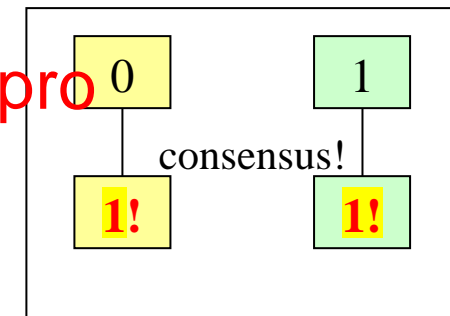
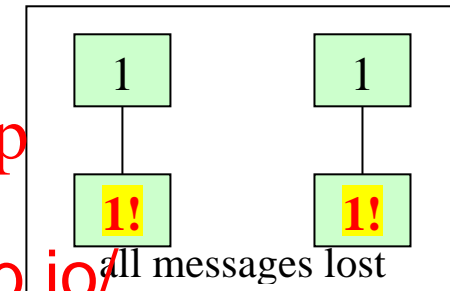
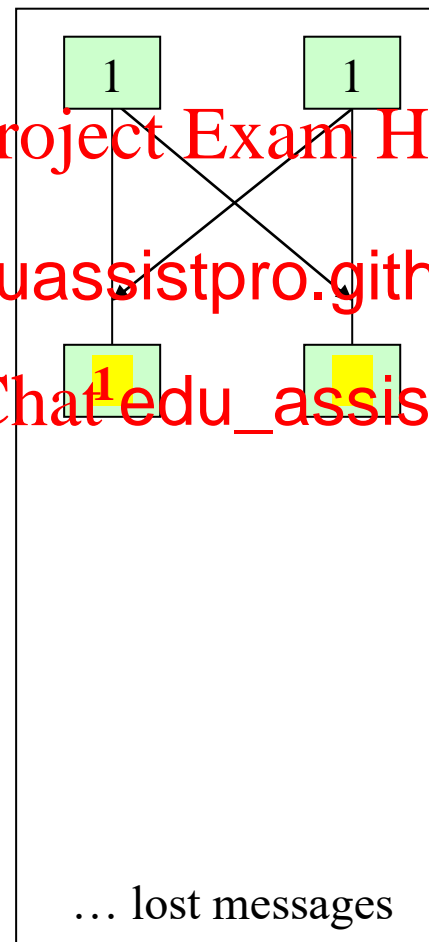
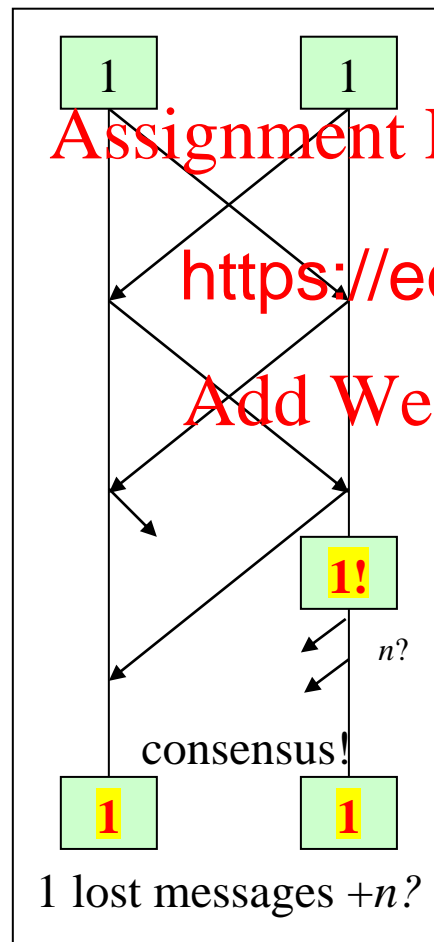
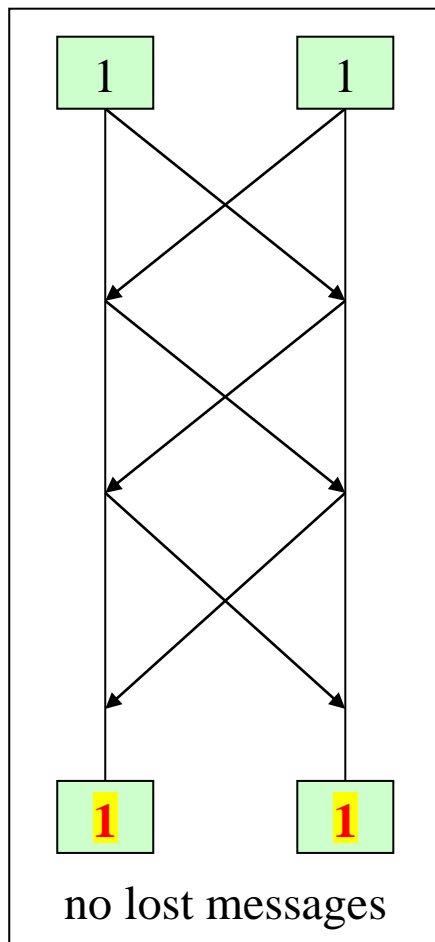
Proof: by induction, using the gi

reement and termination conditions

In our diagrams,  denotes a no-return

certain decision will inevitably follow

(even if no more messages are received)



Assignment Project Exam Help

DISTRIBUTED COMMIT

- 2 PHASE COMMIT – WEAK TERMINATION (“BLOCKING”)
- 3 PHASE COMMIT – STRONG TERMINATION (“NON-BLOCKING”)

Termination: All non-faulty processes eventually decide [STRONG]. (3PC)

or

If there are no failures then all processes eventually decide [WEAK].

Agreement: No two processes (even if faulty) ever decide on different values. (2&3PC)

Validity: If any process starts with the initial value 0, then 0 is the only one possible decision value.

initial 0 → can start as decided 0!
and If all processes start with the same initial value 1 and there are no failures, then 1 is the only one possible decision value.
 [WEAK]. (2&3PC)

Diagrams

Process	Initial	Decision
Decided, not failing	$\forall 1$	$\forall 1$ or $\forall 0^{\dagger}$
Decided, before failing		$\forall 1$ or $\forall 0^{\dagger}$
Non-decided, failed		—
Non-decided, not failing (2PC*)		—

Process	Initial	Decision
Decided, not failing	$\exists 0$	$\forall 0$
Decided, before failing		$\forall 0$
Non-decided, failed		—
Non-decided, not failing (2PC*)		—

\dagger Decision 0 is allowed only if one/more processes fail

\clubsuit The 2PC weak termination condition allows non-faulty process (with initial value 1) that never decide, that remain “blocked” (unless they receive some “magical” help).

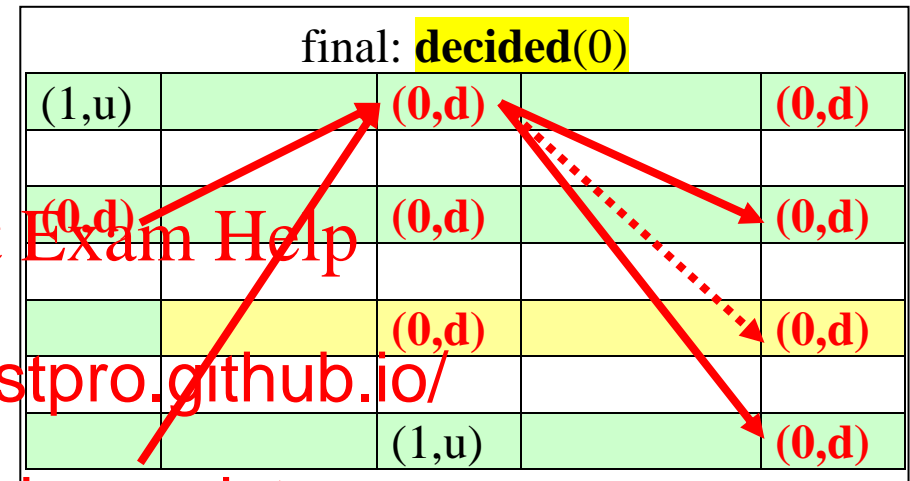
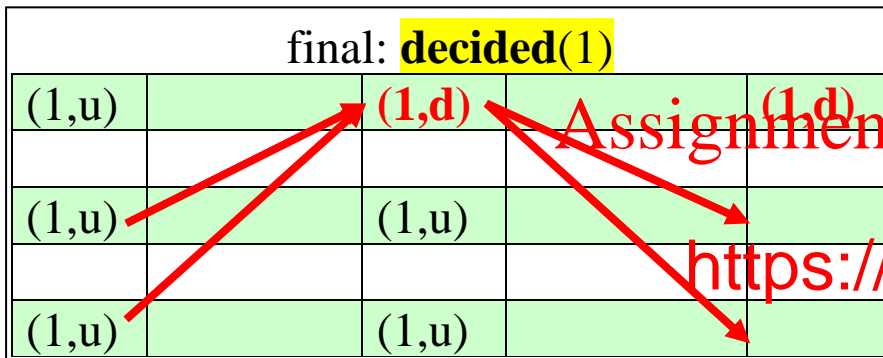
THE 2 PHASE COMMIT – WEAK TERMINATION (“BLOCKING”)

We do not discuss here:

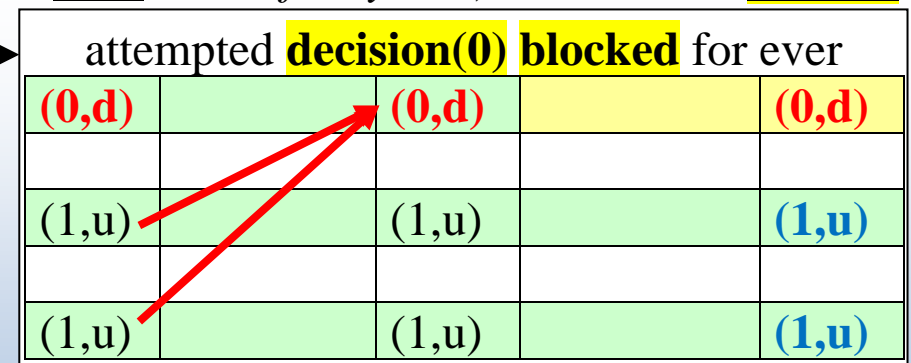
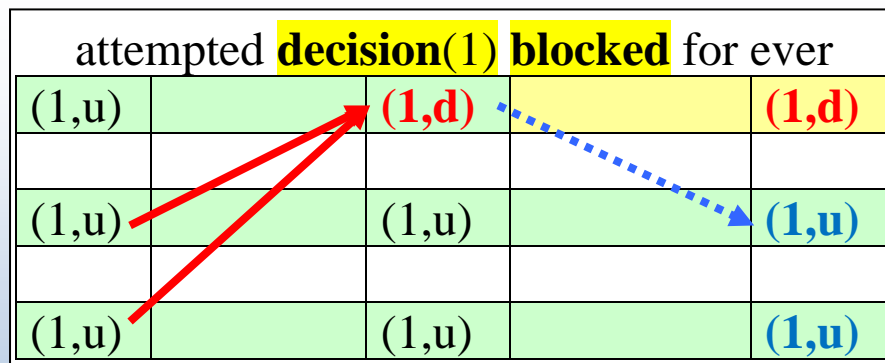
- how the processes start this
- how do they recover after failures.

- Participants: coordinator (leader), cohorts.

- State $uncertain(1) \rightarrow decided(1)$, or $+failed$



If #1 fails, then #2&3 cannot differentiate between left if they talk), will remain **blocked**



- 2PC blocks because the leader is too greedy to decide on 1 and fails before sending its messages

THE 3 PHASE COMMIT – STRONG TERMINATION (“NON-BLOCKING”)

Assignment Project Exam Help

3PC algorithm is less greedy than 2PC and new coordinators arise in place of failed ones

States (one more, ready: $\text{decided}(0) \leftarrow \text{ready}(1) \rightarrow \text{decided}(1)$, or failed)

final: **decide(1)**

(1,u)		(1,r)		(1,d)		(1,d)
(1,u)		(1,u)				
(1,u)		(1,u)				

final: **decide(0)**

(1,u)		(0,d)		(0,d)		
(0,d)		(0,d)		(0,d)		
(0,d)		(0,d)		(0,d)		
(1,u)		(1,u)		(0,d)		

○ State transition rules are “natural” and similar for all rounds, except:

○ In the first 3 rounds:

○ Any missing message \Rightarrow **decided(0)!**

○ If all are **uncertain**, they will *attempt* \Rightarrow **decided(1)!**

○ In round 4 and following:

○ Any missing message is **ignored!**

○ If all the rest are **uncertain**, they will *certainly* \Rightarrow **decided(0)!**

AVOIDING THE BLOCK

Assignment Project Exam Help

2PC would attempt **decision(1)** but **3PC doesn't block** \Rightarrow final **decided(1)**

(1,u)		(1,r)		(1,r)								
(1,u)		(1,u)		(1,r)		(1,r)		(1,r)		(1,d)		(1,d)
(1,u)		(1,u)		(1,u)		(1,u)		(1,u)		(1,r)		(1,d)

Assignment Project Exam Help

2PC would attempt **decision(0)** but **3PC doesn't block** \Rightarrow final **decided(0)**

(1,u)		(1,r)										
(1,u)		(1,u)		(1,u)		(1,u)				(0,d)		(0,d)
(1,u)		(1,u)		(1,u)		(1,u)		(1,u)		(0,d)		(0,d)

2PC would attempt **decision(0)** but it **would block** – 3PC doesn't block \Rightarrow final **decided(0)**

(0,d)		(0,d)		(0,d)								
(1,u)		(1,u)		(1,u)		(1,u)		(0,d)		(0,d)		(0,d)
(1,u)		(1,u)		(1,u)		(1,u)		(1,u)		(0,d)		(0,d)