14 CONSENSUS AND AGREEMENT ALGORITHMS

FAILURE MODELS

fail-stop → a process may crash in the middle of a step (fail to send a msg to all desired processes)
(crash)

byzantine → a process may behave arbitrarily (propagate fake news)

Empire on middle ages

(A)SYNCHRONOUS COMMUNICATION

Sync → process P_i's failure to send msg can be recognized at end of round. (default data values for next round)

Async → failure to send msg can't be distinguished from reciept of msg taking very long time.

NETWORK CONNECTIVITY

processes fully connected (logically)

SENDER IDENTIFICATION

Sender of a msg is always known!

CHANNEL RELIABILITY

channels are reliable, only processes may fail.

UNAUTHENTICATED MESSAGES

faulty process may tamper a msg.
(also called oral msg)

AGREEMENT VARIABLE

we use boolean variables for simplicity (other data types can be used with some results).

AGREEMENT PROBLEM

all non-faulty processes agree about initial value of source process.

agreement: same value — termination: all decide
validity: same as source process — non-faulty

CONSENSUS PROBLEM

each process has an initial value, but all must agree on one value.

agreement: same value — termination: all decide
validity: same as initial value if all initial values are same

in a failure-free system, consensus can be reached by each process by collecting the initial value of every process, and arriving at a decision using a function, like, min, max, majority. the decision would be the same for every process since there are no failures.

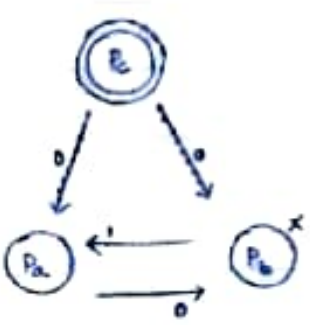
FAIL-STOP CONSENSUS

for n processes with up to f failures, the algorithm has f+1 rounds. at start of each round, each process sends its variable to all other processes. at end of round, each process takes minimum of the values recieved from each process and sets its variable.

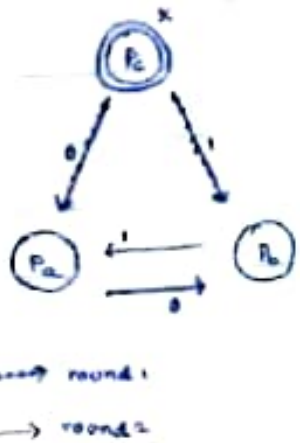
on f+1 rounds, at least 1 round has no failures. at the end of that round consensus is reached. further rounds do not affect the consensus. msg complexity = $O((f+1)n^2)$

BYZANTINE CONSENSUS

with n=3 processes byzantine consensus can't be achieved (f=1). this is because when P_b is faulty or P_c is faulty. P_a recieves the same set of msgs.
can't tell who is faulty, and thus which msg to ignore)



P_c - source process

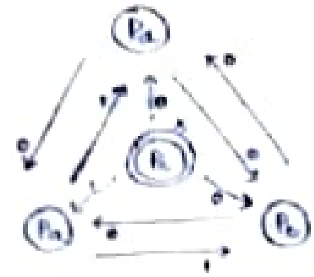


---> round 1

—> round 2

When $n=4$ processes and $f=1$, consensus can be achieved by each process taking the majority of the values it recieves.

(At end of round 2 ($f+1$) each process can identify who is faulty) [illegible] used

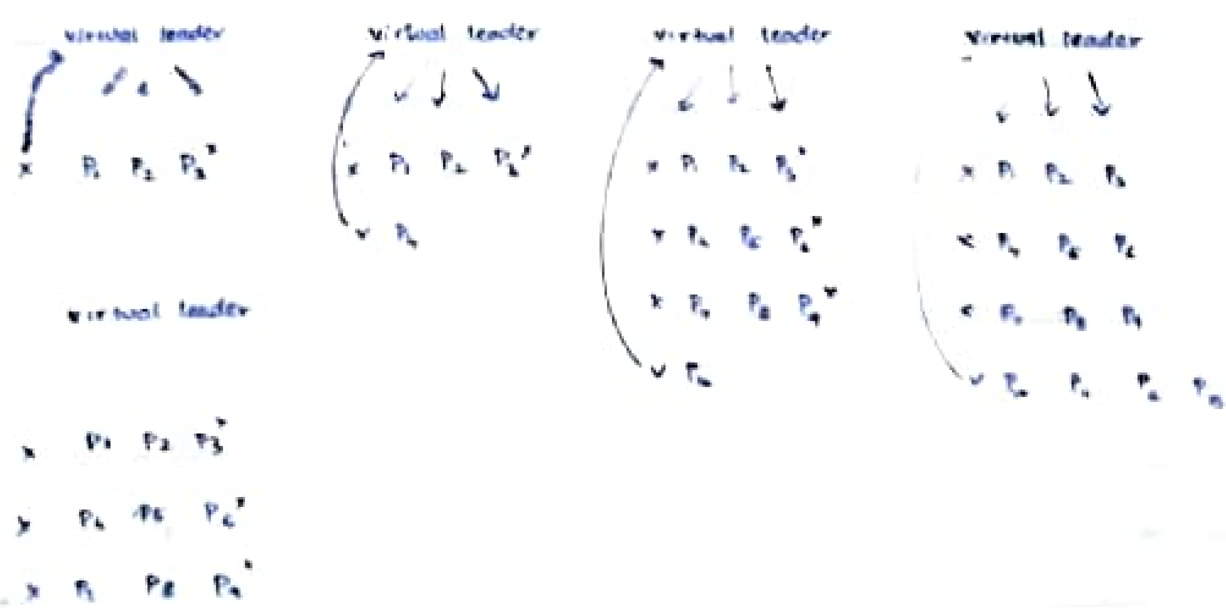


Byzantine consensus can only be achieved if $f \leq \left\lfloor \frac{n-1}{3} \right\rfloor$

cannot be achieved if $3f \geq n$

PROOF :- Consider $n=3f$. In the worst case, n processes can be distributed into f groups each with 1 faulty process. In any other case, atleast one group can achieve consensus, but as a whole they may not. (why?)

Also adding a single extra process means as a whole consensus can be achieved. Any additional process (non-faulty) can always achieve consensus among themselves.



fisher, lynch, paterson

failure model	synchronous system	asynchronous system
none	✓ agreement	✓ agreement
	✓ common knowledge	✓ concurrent common knowled.
~~fail-stop~~		
fail-stop	✓ agreement	x agreement
	$f < n$ processes	
	$\Omega(f+1)$ rounds	
byzantine	✓ agreement	x agreement
	$3f < n$ processes	
	$\Omega(f+1)$ rounds	

introducing a leader will solve all problems!

but what if the leader becomes faulty?

PAXOS ——— fictional legislative consensus system used on paxos island on greece

although no deterministic fault tolerant consensus protocol can guarentee progress in an async network (proved by fischer, lynch, and paterson), paxos guarentees safety (consistency), and the conditions that could prevent it from making progress are difficult to provoke.