node class

class variables

- nodes = []
- current_ballot = 0
- value_to_propagate = None
- leader = None

nodes → { ip, port, pkey, status }

status:
- prepare_sent (ballot)
- prepared_sent (ballot, value)
- proposed_sent (ballot, value)
- accept_sent (ballot, value)

start here

get nodes[] from Coordinator

init

open thread

listening_thread (calls msg_handler)

"prepare" (all from node[pkey])

- check ballot
- If higher → update it and leader
  - set node[pkey] status to "prepare sent"

send prepared (ballot, value)

"prepared"

- check if node[pkey] is "prepare sent"
- set value if prepared has a value in it

send propose (ballot, value) to node pkey

"accept"

→ mark node[pkey] as accepted (ballot, value)

send accepted

"propose"

- check if we have prepared to (ballot, pkey)
- either
  - ignore
  - send accept

} update node status

coordinator msgs:

1. new peer : replace node w/ new instance of class node
2. propagate_value (v): tells node to try to become leader and replicate v

start propagate_thread (v, ballot):

:= loops through nodes[] sending prepares and checking status for majority of accepts (of nodes)

will timeout, say after 5 min w/o paxos commit

listening_thr kills this if higher ballot msg received

3. Paxos_done (v): lets node learn the final commited value from paxos

send this msg

4. msg_commited (from node to coord.): lets coord. know that propagate worked, coord. will tell other nodes w/ (3).

Note: (1), (2), (3) are from coordinator to nodes