Solving Consensus using $\diamond\mathcal{S}$

Assumption: n processes; fewer than $n/2$ may crash

Rotating Coordinator Algorithm

- Processes are numbered $1, 2, \ldots, n$
- Each process executes a sequence of asynchronous *rounds*
- In round r, process $(r \bmod n) + 1$ is the *coordinator*
- Each process maintains:
  - its current round number r
  - an *estimate* of the eventual decision value
  - a timestamp ts indicating the round in which it got its current *estimate* (initially, $ts = 0$)

Basic Rotating Coordinator Algorithm using $\diamond\mathcal{S}$

$r \leftarrow 0$; $estimate \leftarrow$ proposed value ; $ts \leftarrow 0$

while (undecided)

$r \leftarrow r + 1$; $c \leftarrow (r \bmod n) + 1$

{in round r all messages sent/received are tagged with r}

Phase 1: *Every* p sends $(estimate, ts)$ to coordinator c

Phase 2: *coordinator* c {c gathers estimates}

- waits until it receives $\lceil (n+1)/2 \rceil$ $(estimate, ts)$ messages
- sets $new\_estimate \leftarrow estimate$ with the largest ts received
- sends $new\_estimate$ to all p

Phase 3: *Every* p

- waits until (a) it receives $new\_estimate$ from c
  or (b) c is suspected by $\diamond\mathcal{S}$
- if (a) then $estimate \leftarrow new\_estimate$
  $ts \leftarrow r$
  send ack to c
- if (b) then send $nack$ to c

Phase 4: *coordinator* c {c gathers acks}

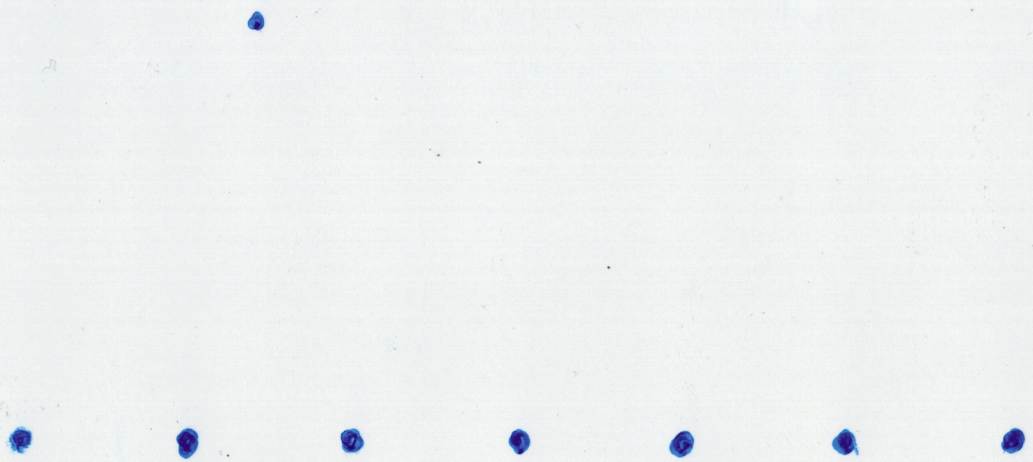
- waits until it receives $\lceil (n+1)/2 \rceil$ acks or $nack$s
- if they are all acks then **decide** $new\_estimate$
  and send this decision to all p

Majority-Based Decision-Locking Mechanism

$n = 7$
$t = 3$

ROUND 10

c decides v



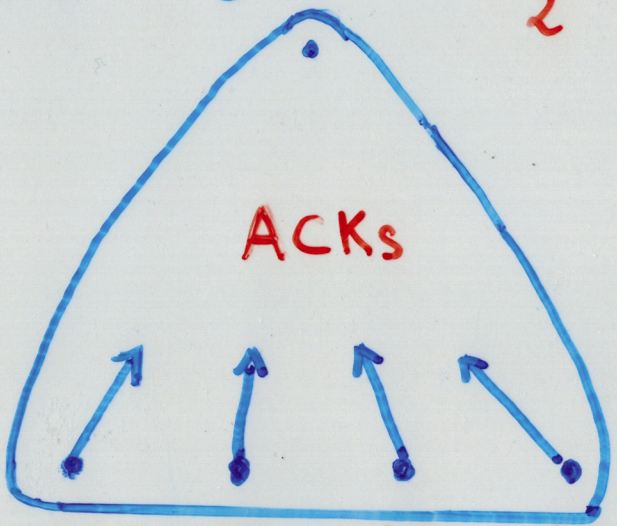
Majority-Based Decision-Locking Mechanism

$n = 7$
$t = 3$

ROUND 10

c decides v

$\Longleftrightarrow$ c gathered $> \frac{n}{2}$ ACKs



Estimates: v v v v

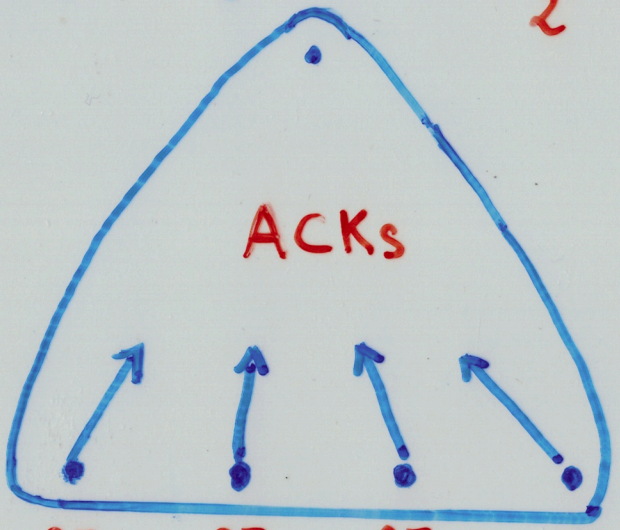
ts : 10 10 10 10

Majority-Based Decision-Locking Mechanism

$n = 7$
$t = 3$

ROUND 10

c decides v
$\Longleftrightarrow$ c gathered $> \frac{n}{2}$ ACKs



Estimates:	v	v	v	v	x	y	z
ts :	10	10	10	10	8	9	7

Majority-Based Decision-Locking Mechanism

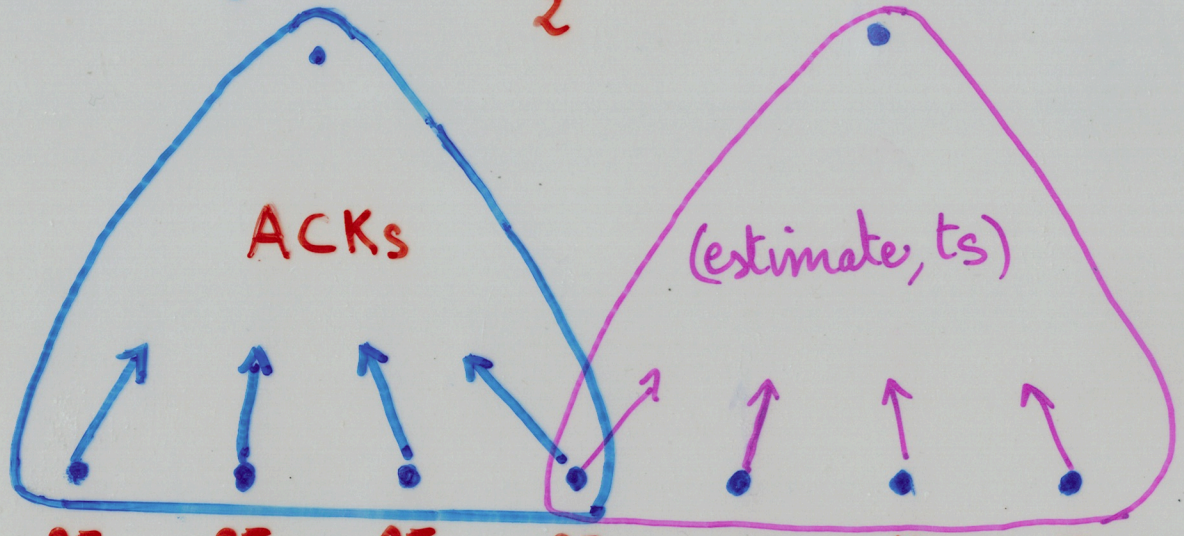
$n = 7$
$t = 3$

ROUND 10

c decides v
$\Longleftrightarrow$ c gathered $> \frac{n}{2}$ ACKs

ROUND 11

c' gathers $> \frac{n}{2}$ (estimates, ts)
c' selects v



Estimates:	v	v	v	v	x	y	z
ts :	10	10	10	10	8	9	7

Why Does this Work?

- *Agreement:* majority-based decision locking mechanism
- *Termination:*
  - By the completeness property of $\diamond\mathcal{S}$, no process blocks forever waiting for a message from dead coordinator
  - By the accuracy property of $\diamond\mathcal{S}$, eventually some correct process p is not suspected. When p becomes the coordinator, it will receive the requisite number of acks to force a decision.

Optimizations

Basic algorithm has many optimizations...

[DFKM96,GOS96,Schiper97]

Slow Catch-up Problem: slow process in a low round must execute all intermediate rounds to catch up with processes in higher rounds.

Skip-Rounds Solution:

Process p in round r goes to a higher round iff:

- if p receives a msg from round $t > r$
  - abort round r
  - go to first round $s \geq t$ such that p trusts $coordinator(s)$
- if p suspects $coordinator(r)$ then
  - abort round r
  - go to first round $s > r$ such that p trusts $coordinator(s)$

Dealing with Message Losses [DFKM96]

- Links are lossy but *fair:*
  a message repeatedly sent is eventually received
- Obvious solution: message retransmission

- Question: must retransmit *every* message?
- Answer: No!

  Thanks to Skip-Rounds, must retransmit only the *last* message sent to each process

Every process p executes the following:

procedure *propose*(v_p)
  $estimate_p \leftarrow v_p$ {$estimate_p$ *is p's estimate of the decision value*}
  $state_p \leftarrow undecided$
  $r_p \leftarrow 0$ {r_p *is p's current round number*}
  $ts_p \leftarrow 0$ {ts_p *is the last round in which p updated* $estimate_p$, *initially 0*}

  {*Rotate through coordinators until decision is reached*}

  while $state_p = undecided$
    $r_p \leftarrow r_p + 1$
    $c_p \leftarrow (r_p \bmod n) + 1$ {c_p *is the current coordinator*}

    Phase 1: {*All processes p send* $estimate_p$ *to the current coordinator*}
      send $(p, r_p, estimate_p, ts_p)$ to c_p

    Phase 2: {*The current coordinator gathers* $\lceil \frac{(n+1)}{2} \rceil$ *estimates and proposes a new estimate*}
      if $p = c_p$ **then**
        wait until [for $\lceil \frac{(n+1)}{2} \rceil$ processes q : received $(q, r_p, estimate_q, ts_q)$ from q]
        $msgs_p[r_p] \leftarrow \{(q, r_p, estimate_q, ts_q) \mid p$ received $(q, r_p, estimate_q, ts_q)$ from $q\}$
        $t \leftarrow$ largest ts_q such that $(q, r_p, estimate_q, ts_q) \in msgs_p[r_p]$
        $estimate_p \leftarrow$ select one $estimate_q$ such that $(q, r_p, estimate_q, t) \in msgs_p[r_p]$
        send $(p, r_p, estimate_p)$ to all

    Phase 3: {*All processes wait for the new estimate proposed by the current coordinator*}
      wait until [received $(c_p, r_p, estimate_{c_p})$ from c_p **or** $c_p \in \mathcal{D}_p$]{*Query the failure detector*}
      if [received $(c_p, r_p, estimate_{c_p})$ from c_p] **then** {p *received* $estimate_{c_p}$ *from* c_p}
        $estimate_p \leftarrow estimate_{c_p}$
        $ts_p \leftarrow r_p$
        send (p, r_p, ack) to c_p
      else send $(p, r_p, nack)$ to c_p {p *suspects that* c_p *crashed*}

    Phase 4: { *The current coordinator waits for* $\lceil \frac{(n+1)}{2} \rceil$ *replies. If they indicate that* $\lceil \frac{(n+1)}{2} \rceil$ *processes adopted its estimate, the coordinator R-broadcasts a decide message* }
      if $p = c_p$ **then**
        wait until [for $\lceil \frac{(n+1)}{2} \rceil$ processes q : received (q, r_p, ack) **or** $(q, r_p, nack)$]
        if [for $\lceil \frac{(n+1)}{2} \rceil$ processes q : received (q, r_p, ack)] **then**
          R-broadcast$(p, r_p, estimate_p, decide)$

{*If p R-delivers a decide message, p decides accordingly*}

when *R-deliver*$(q, r_q, estimate_q, decide)$
  if $state_p = undecided$ **then**
    $decide(estimate_q)$
    $state_p \leftarrow decided$

Figure 6: Solving Consensus using any $\mathcal{D} \in \diamond\mathcal{S}$.