2022 FALL OS
Project 3 Help Document

JINYONG HA

Distributed Computing Systems Laboratory
Department of Computer Science and Engineering
Seoul National University, Korea
2021-11-11

NOTE

- 과제 **#3**

  기한: 11/26 midnight

  방법: 팀별 repository에 proj3 branch 생성 / 발표자료는 대표 한 명이 etl 제출

- 과제 **#4**

  기한: 12/17 midnight

  방법: 팀별 repository에 proj4 branch 생성 / 발표자료는 대표 한 명이 etl 제출

Project 3 Overview

- Implement rotation-based read/write lock
- Each lock has a “rotation range”
    - Every lock can be acquired when the current rotation is in the rotation range
    - Or, it is blocked until the current rotation is located in the rotation range

- Read lock could be acquired when no acquired write lock range is overlapping with its range
- Write lock could be acquired when no acquired read/write lock range is overlapping with its range
    - Exclusive access

Rotation Range

- 1 axis: rotation(use daemon)
  - Actually, Tizen orientation has three axes! (Azimuth, Pitch, Roll)
- (degree - range) <= range <= (degree + range)
- Rotation ranges are inclusive
  - Ex) [30, 60], [60, 90] are overlapped
- Rotation ranges are circular
  - Ex) [330, 30] and [30, 330]
  - [330 … 0 … 30]
  - [30 … 180 … 330]

Range Example (1)

- Rotation 1
  - degree = 30
  - range = 30
- Rotation 2
  - degree = 45
  - range = 30

Are they overlapped?

Range Example (1)

- Rotation 1
  - degree = 30
  - range = 30
- Rotation 2
  - degree = 45
  - range = 30

- Rotation 1
  - [0, 60]
- Rotation 2
  - [15, 75]

Are they overlapped? YES!

Range Example (2)

- Rotation 1
  - degree = 30
  - range = 60
- Rotation 2
  - degree = 315
  - range = 30

Are they overlapped?

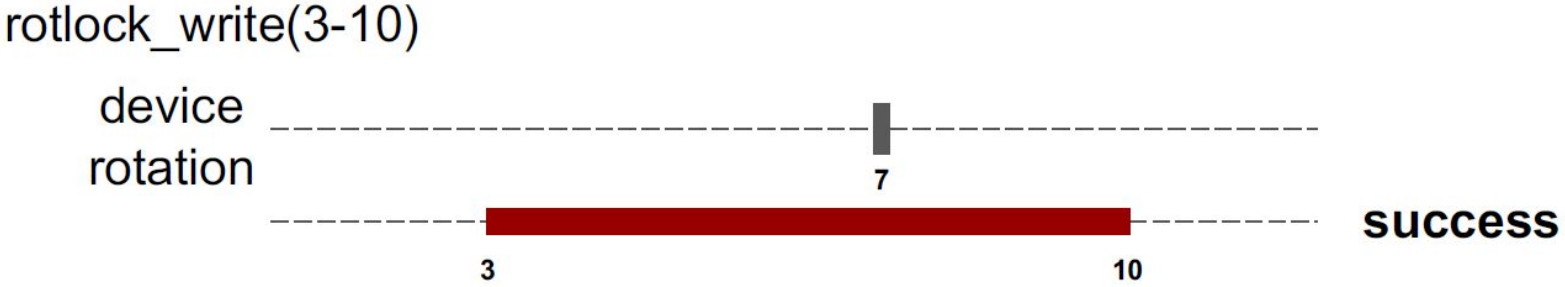
Range Example (2)

- Rotation 1
  - degree = 30
  - range = 60
- Rotation 1
  - [0, 90] + [330, 360)

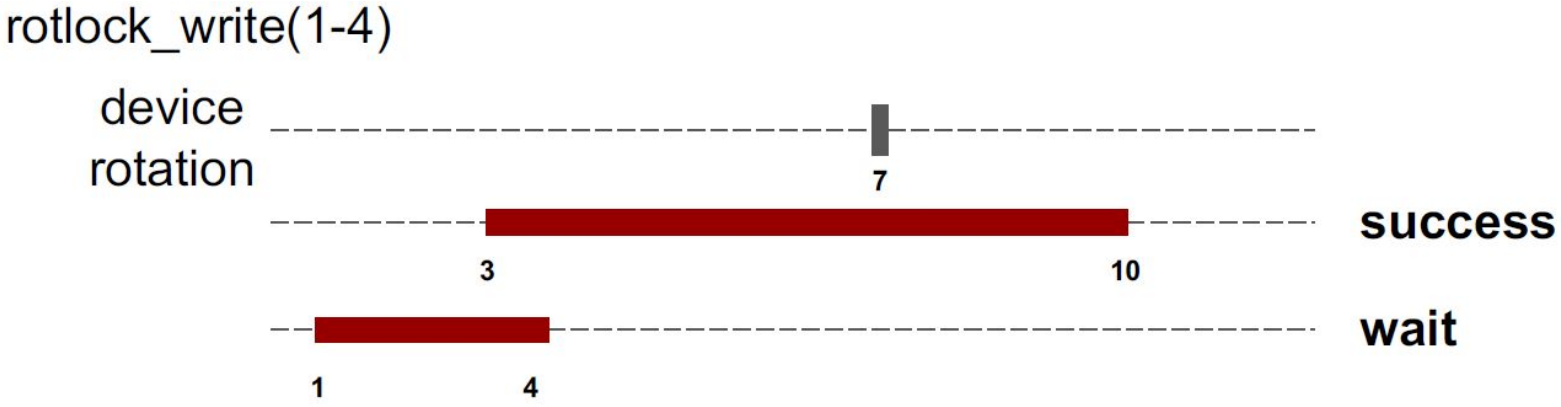
- Rotation 2
  - degree = 315
  - range = 30
- Rotation 2
  - [285, 345]

Are they overlapped? YES!

Rotation Lock Example



Rotation Lock Example

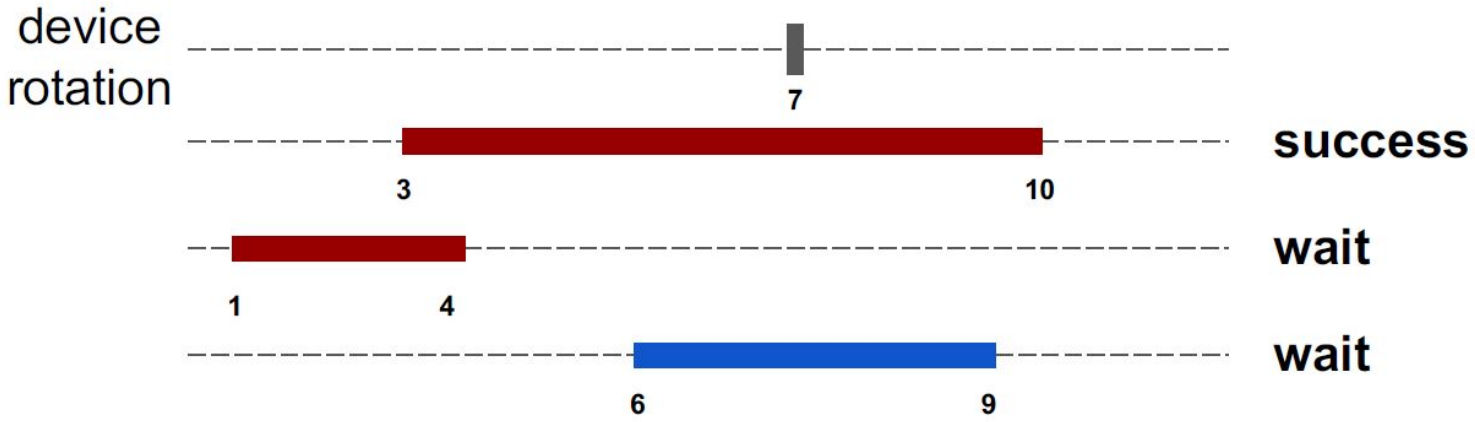


writer

reader

Rotation Lock Example

rotlock_read(6-9)

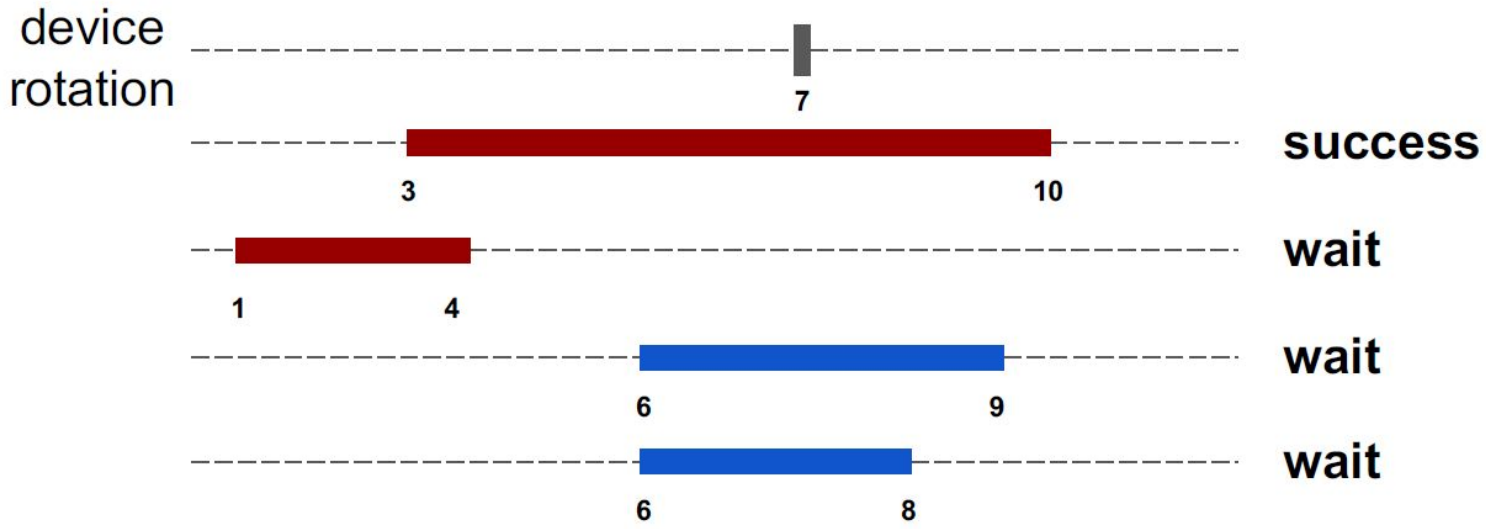


writer

reader

Rotation Lock Example

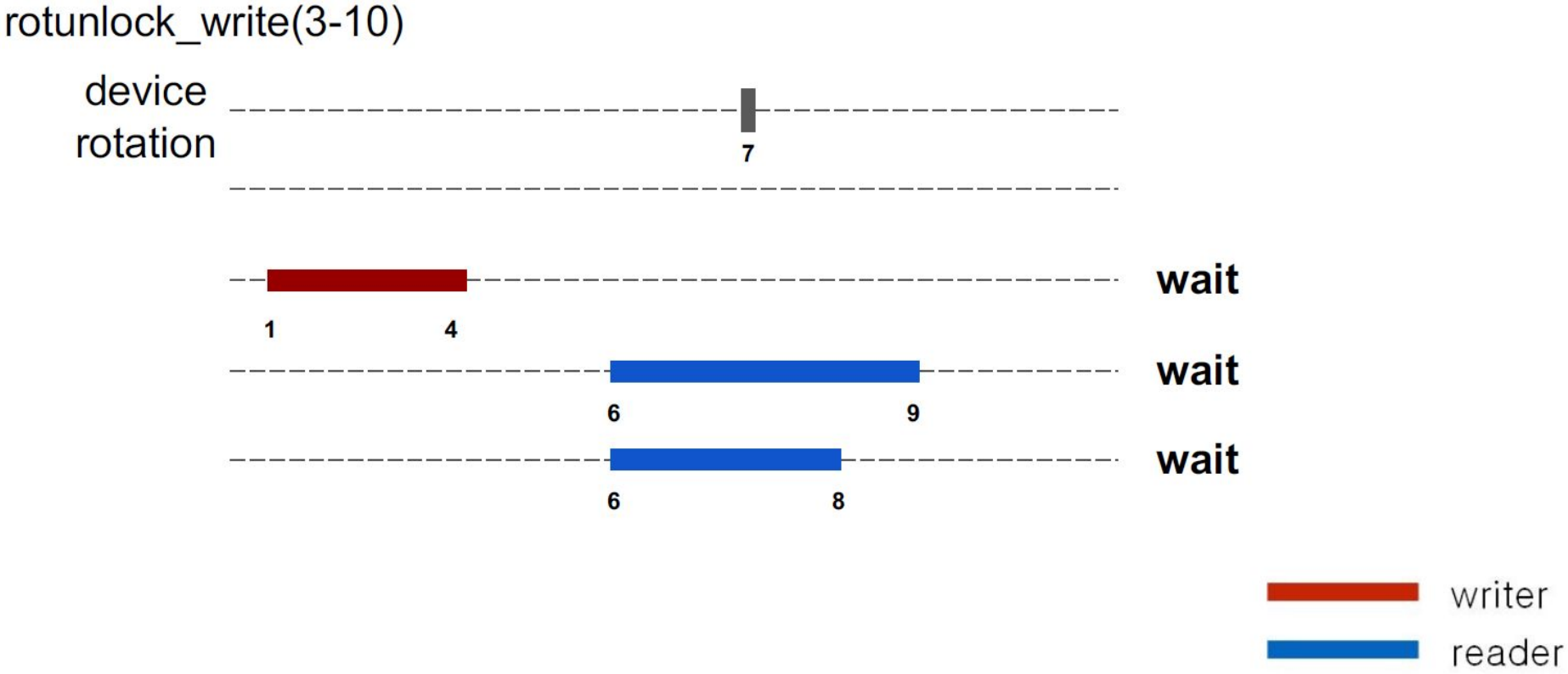
rotlock_read(6-8)



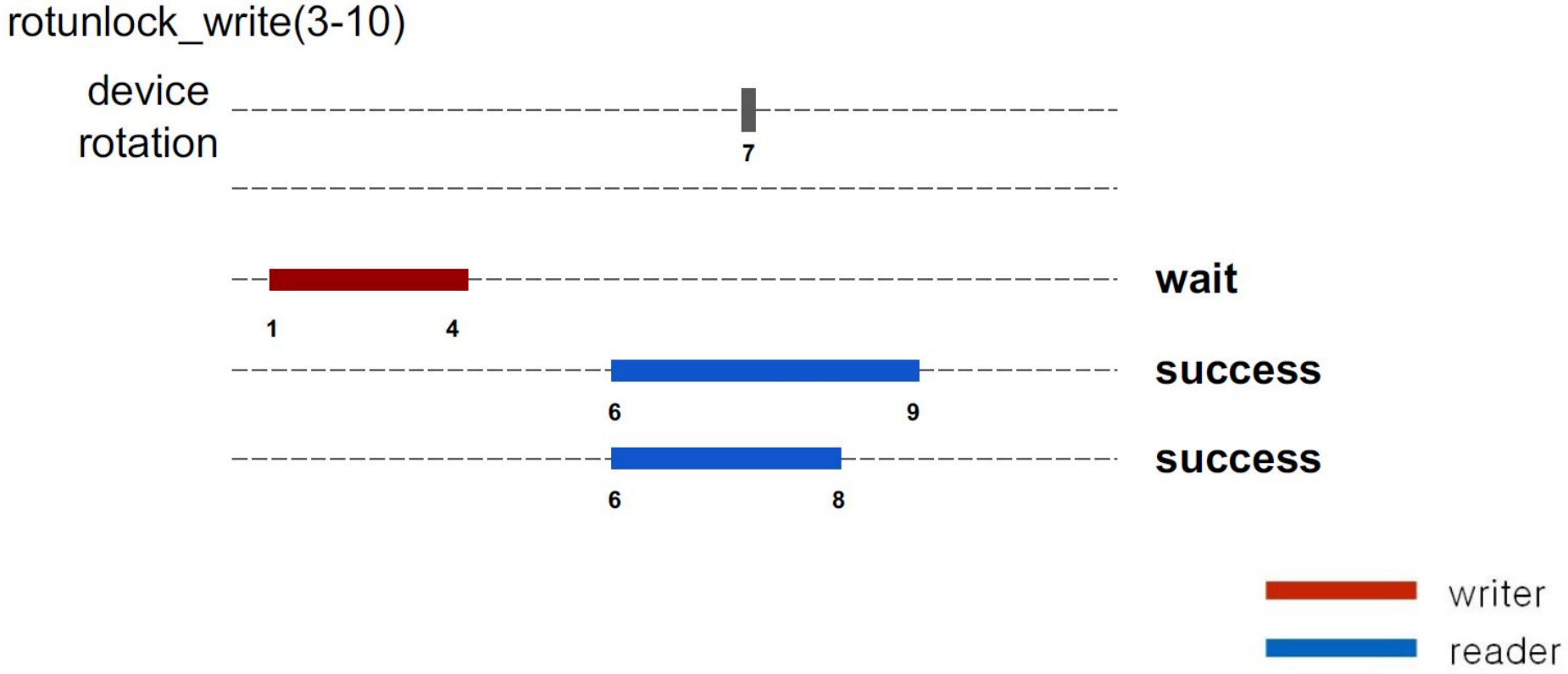
writer

reader

Rotation Lock Example



Rotation Lock Example



Rotation Lock Example

Device rotation changed to 3

writer

reader

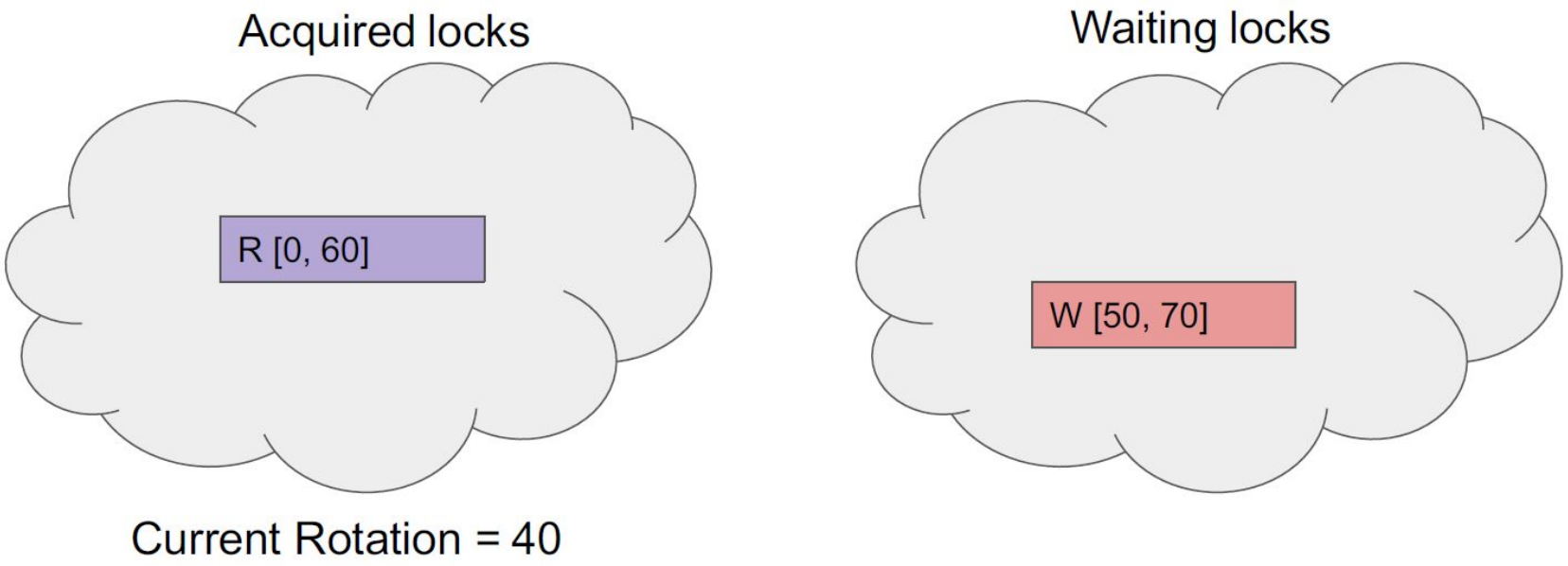
Preventing writer starvation

- You should implement a policy for preventing starvation of writers
  - Why?

    If reader comes in infinitely, a writer could wait forever!
- If a reader holds a lock and a writer wants to take the lock, no more readers can take the lock
- If you design your own additional policy, explain that in your README.md file, report, and slides!

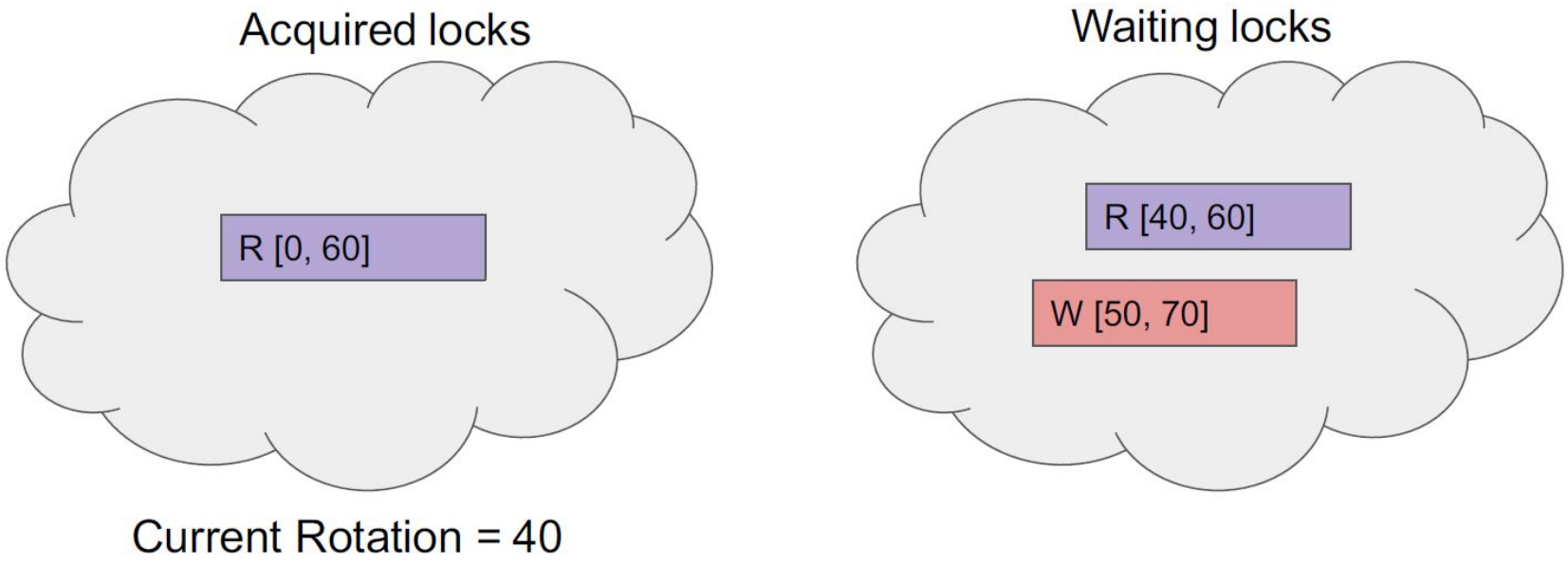
How to prevent Starvation?

- A write lock is waiting for the rotation changes & the reader to release its lock



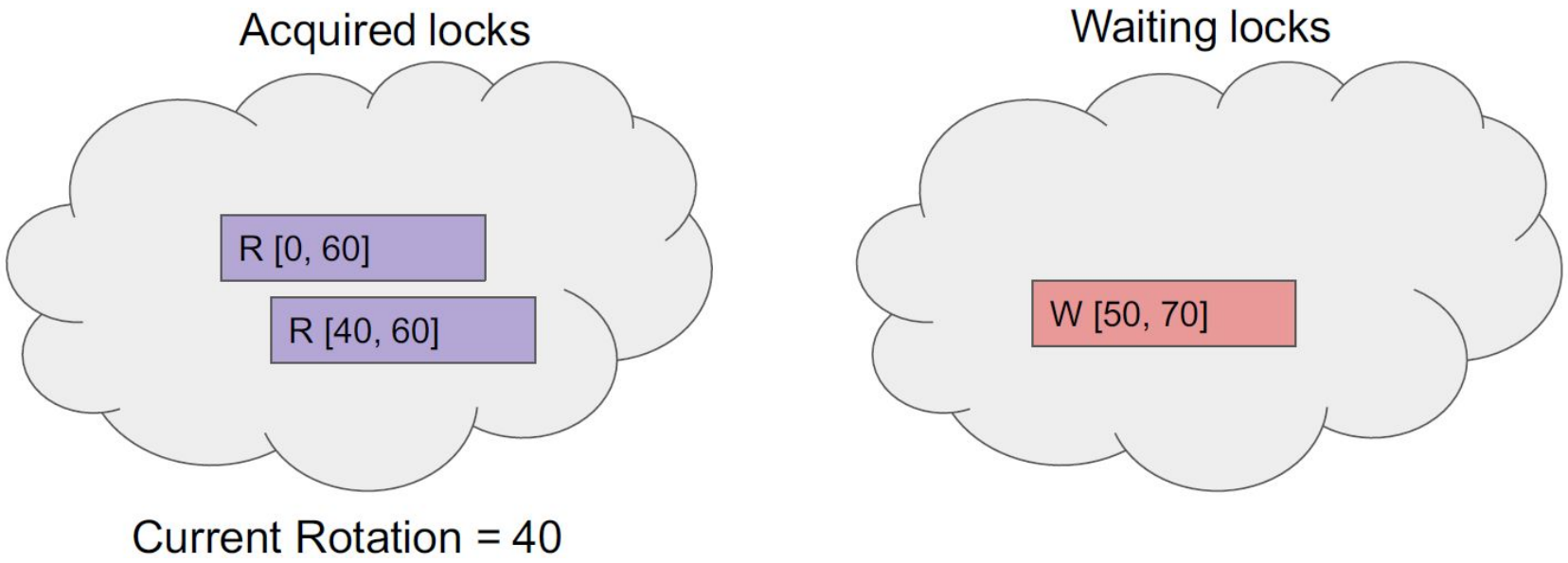
How to prevent Starvation?

- A read lock [40, 60] came



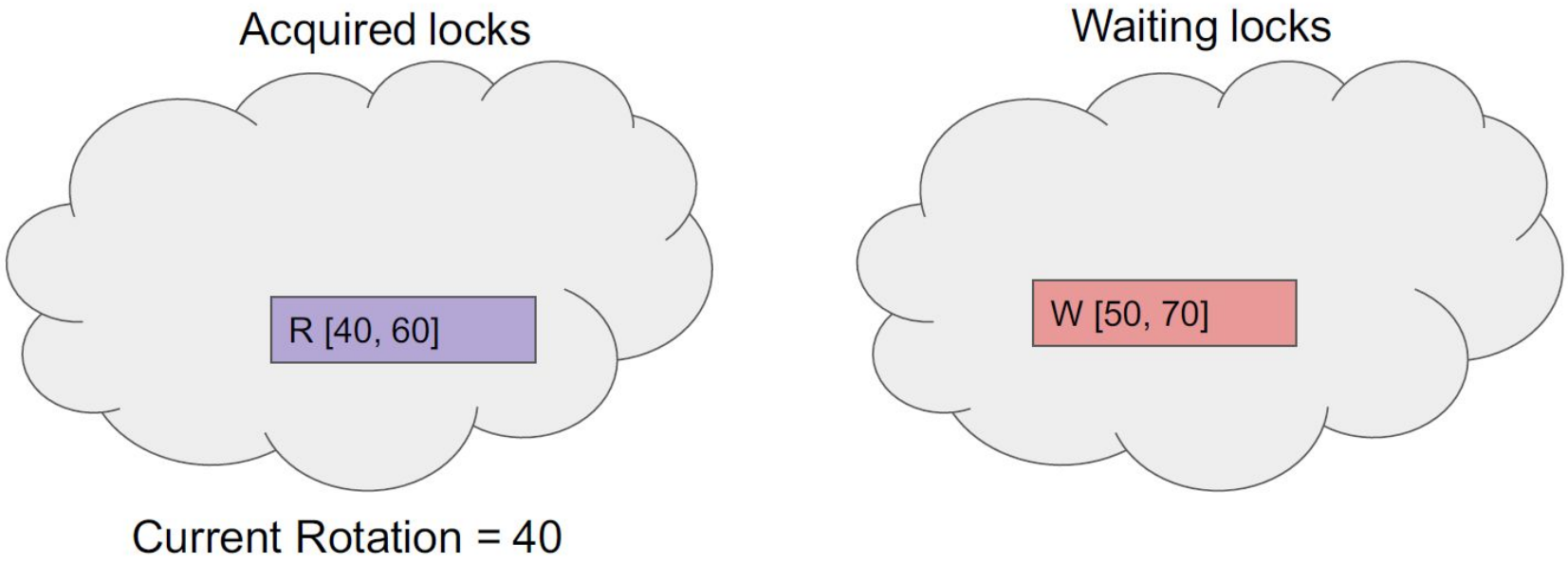
How to prevent Starvation?

- 40 ∈ [40, 60] → Acquires its lock immediately



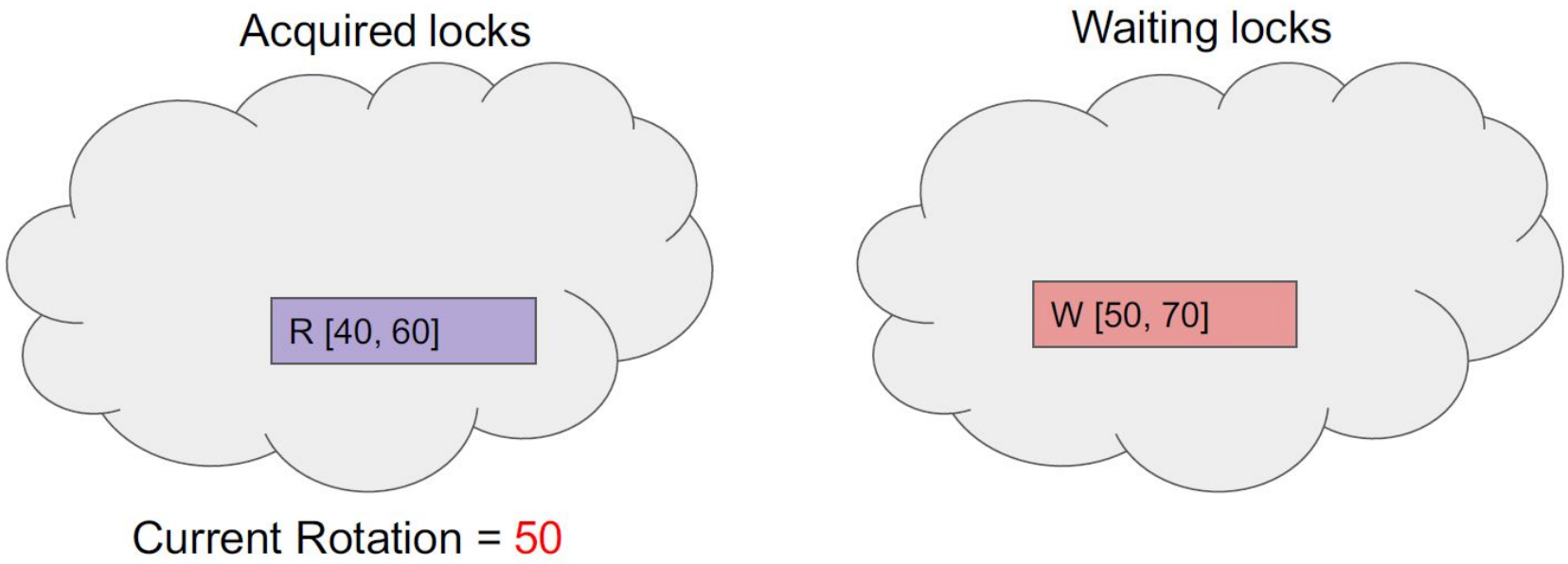
How to prevent Starvation?

- R [0, 60] releases its lock



How to prevent Starvation?

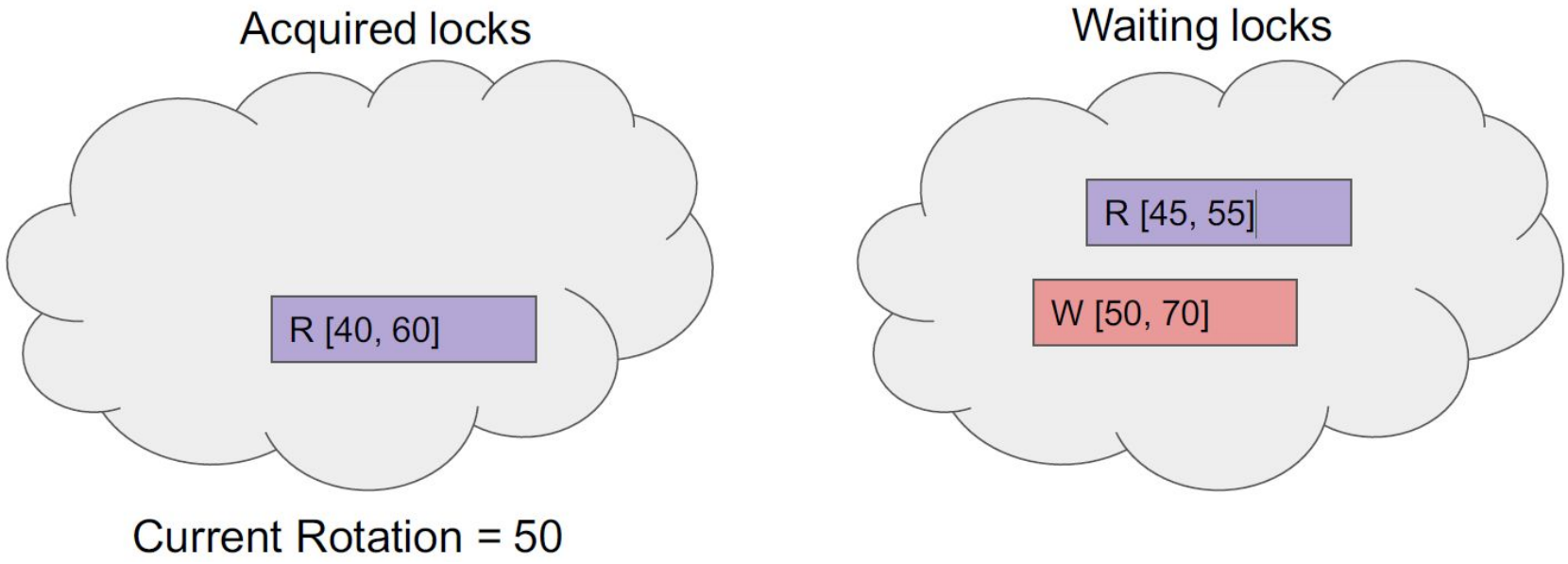
- Rotation changes 40 →50



How to prevent Starvation?

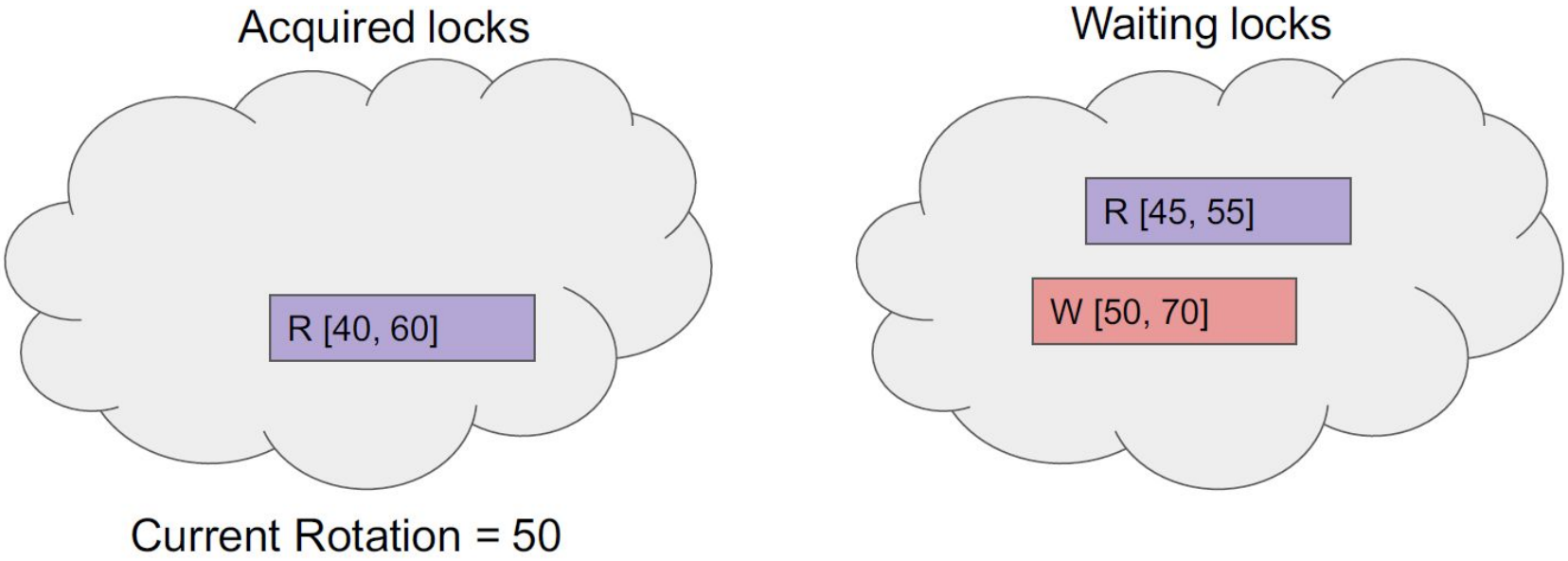
How to prevent writer starvation?

Acquired locks Waiting locks



How to prevent Starvation?

- W [50, 70] is waiting and 50 ∈ [50, 70]
- W [50, 70] is waiting for R [40, 60] to release its lock
- [45, 55] overlaps with [50, 70]



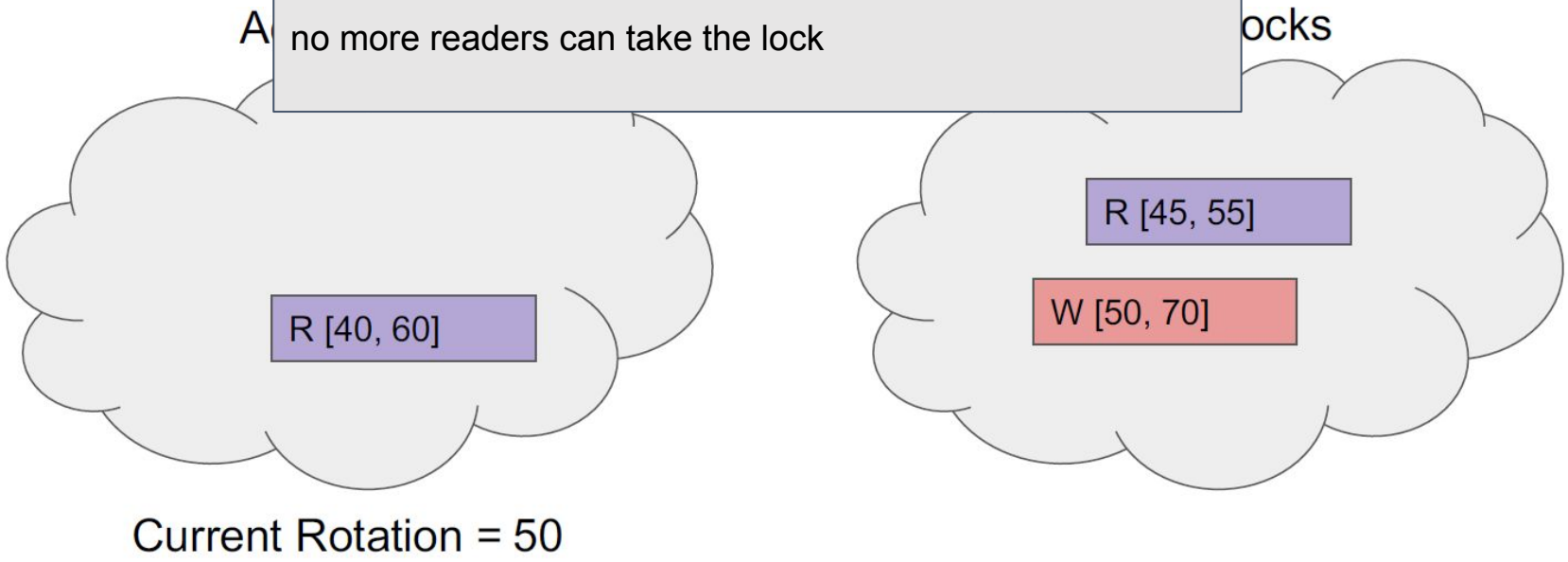
How to prevent Starvation?

- W [50, 70] is waiting and 50 ∈ [50, 70]
- W [50, 70] is waitin
- [45, 55] overlaps w

If a reader holds a lock

and a writer wants to take the lock,

no more readers can take the lock



How to prevent Starvation?

- W [50, 70] is waiting and 50 ∈ [50, 70]
- W [50, 70] is waitin
- [45, 55] overlaps w

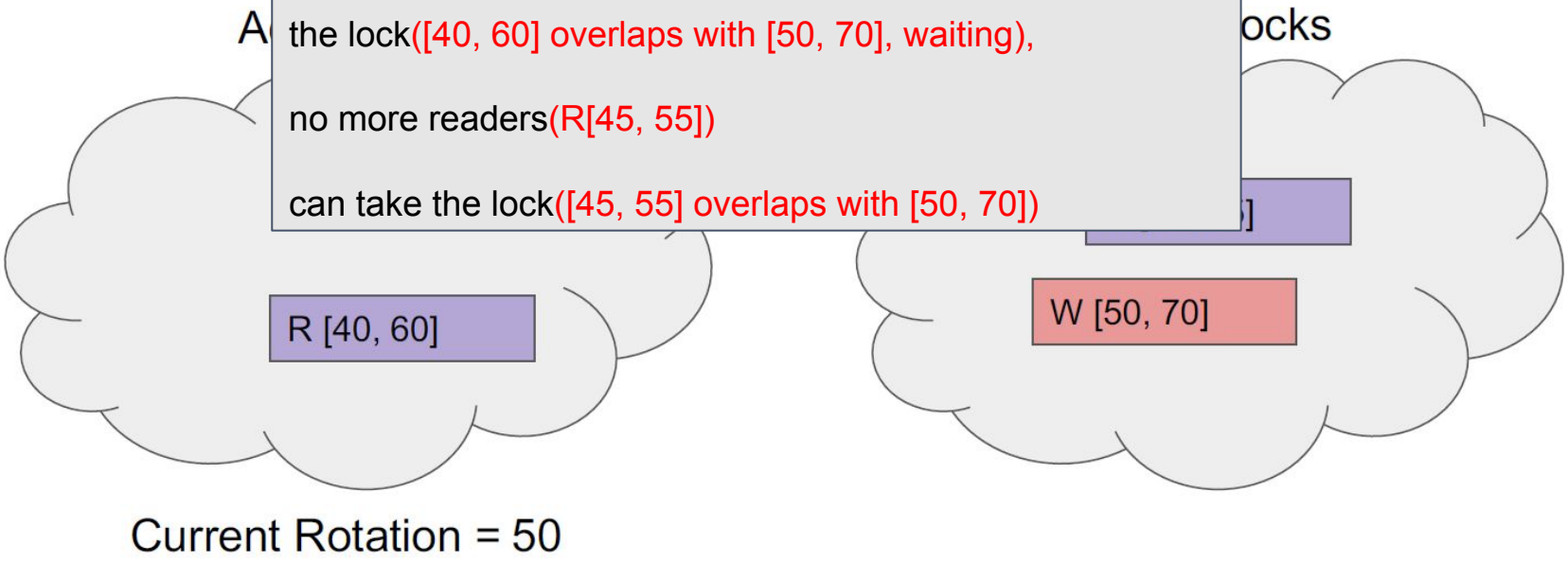
If a reader(R[40, 60]) holds a lock

and a writer(W[50, 70]) wants to take(50 ∈ [50, 70])

the lock([40, 60] overlaps with [50, 70], waiting),

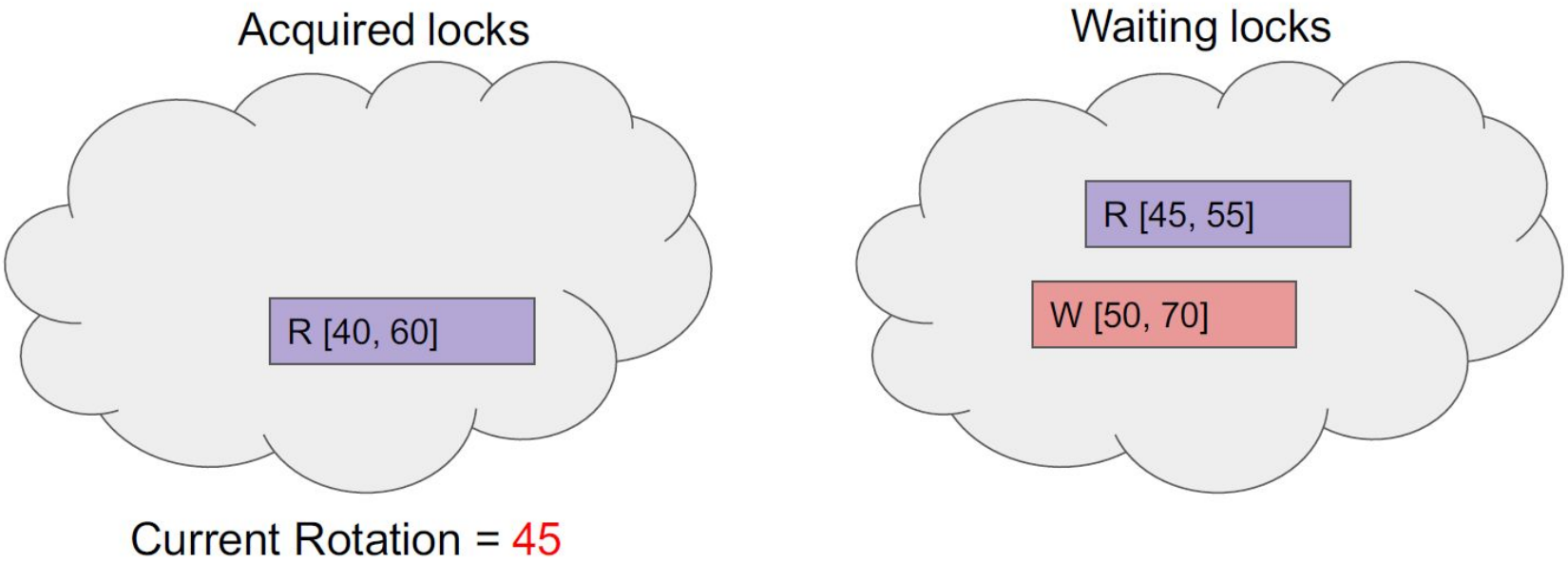
no more readers(R[45, 55])

can take the lock([45, 55] overlaps with [50, 70])



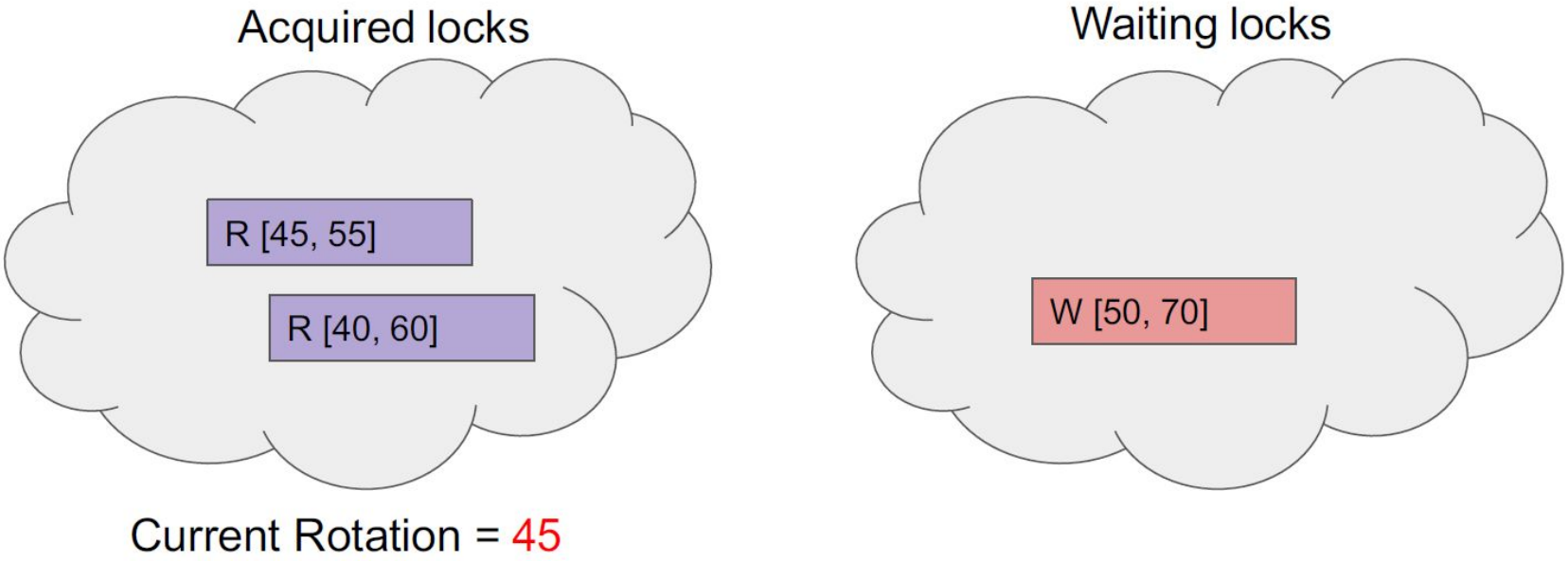
How to prevent Starvation?

- Rotation changes 50 →45



How to prevent Starvation?

- W [50, 70] cannot grab its lock anymore (∵ 45 ∉ [50, 70])
- Starvation prevention policy is no more applied → R [45, 55] acquires its lock

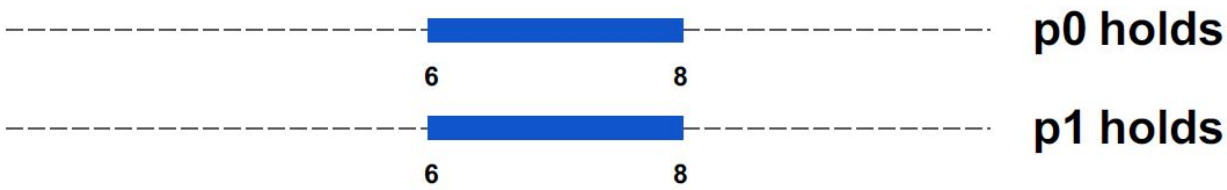


Terminating routine

- When a thread that has some holding or waiting locks is terminating, the remaining locks should be released (holding) or removed (waiting).
- Hints
  - exit_rotlock() in kernel/rotation.c
    - Release holding locks
    - Remove waiting locks
  - Inject exit_rotlock() to do_exit() in kernel/exit.c

Isolation

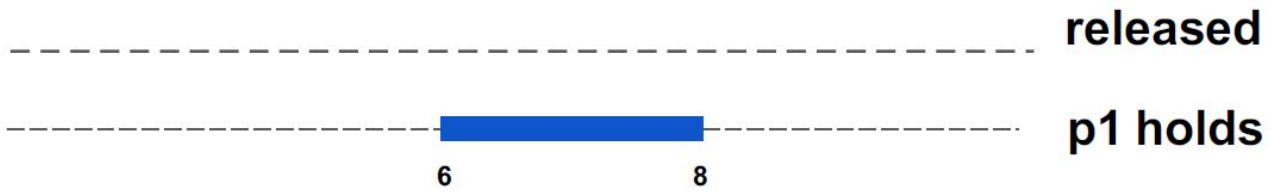
- Multiple processes shares same rotation lock system.
  - You have to identify which process (thread) the lock belongs to.
- A process can’t release locks that other processes hold.



Isolation

- A process can’t release locks that other processes hold.

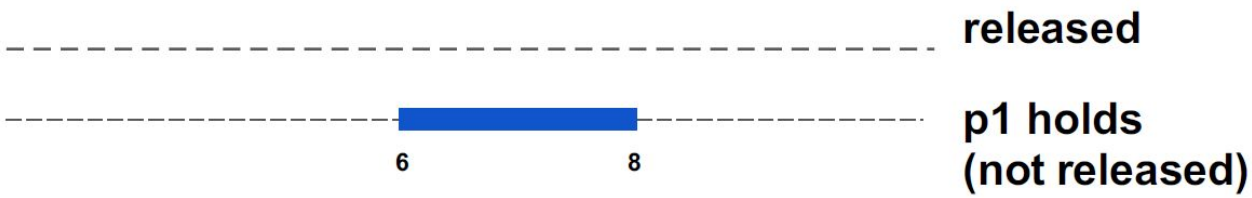
p0 calls unlock(6-8)



Isolation

- A process can’t release locks that other processes hold.

p0 calls unlock(6-8) **again**



TA hints

- You will have create things like acquired lock list and waiting lock list
  - Different approaches are also possible :)
  - Ex) Bitmap, linked list, ...
- Accessing those lists could result in race conditions
  - if you are working on multicore machine
  - You should carefully design your code to prevent possible race conditions
    - Ex) One thread is removing a lock from waiting list, but another thread can access to the waiting list at the same time (inconsistency issues)

TA hints

- Possible approaches
  - Global
    - Ex) Manage acquired lock list and waiting lock list using a same lock
  - Fine-grained
    - Ex) Manage acquired lock list and waiting lock list using separated locks
    - Better concurrency, more complicated :)
- Synchronization mechanism
  - Spin Lock
    - Eligible for short sleep (e.g. short list iteration)
  - RCU (Read-Copy-Update)
  - …

TA hints

- How to block processes || How to wake up blocked processes
  - You may use - not explained in this ppt :)
    - Wait Queue (Starts with DECLARE_WAIT_QUEUE_HEAD)
    - Condition Variable (Define your own CV)
    - Mutex
    - ...

TA hints

- Lists could be changed during iteration
  - list_for_each_entry_safe could be useful to you
- Please remember that …
  - The rotation range is circular!
    - You should implement a logic for determining two circular ranges are overlapping or not
  - Beware of deadlocks!

Selector and trial

- Selector & Trial require a same lock (0 <= range <= 180)
  - If current rotation is 240, both selector & trial wait.
- When the device rotation is out of that range, both Selector & Trial stop working
- When the device rotation gets inside that range, Selector & Trial start to work

Selector **Trial**

write_lock_____

_____read_lock & wait

10

write_unlock_____acquire_lock

10 = 2 * 5

acquire lock_____read_unlock

About submission

Same rules as previous projects

- Make sure your branch name: proj3

- Save your C program as: test/selector.c, test/trial.c
- Check for format : slides title / demo name / test file names / branch name

and directory name

- Please aggregate your demo videos (=submit only one video!)

****Important****

- Git diff file that contains all of your "**source code**" modification from master branch: TeamX.diff

  diff file should be buildable when applying it on master branch (git apply TeamX.diff)

  Document files (like *.md) are not required to be included

  diff files not buildable cannot be scored

  Your kernel implementation should work when just boot your kernel Image (no additional command like insmod)

- Deadline
  - Due:11/26 midnight + 3 days of late submission allowed (10% will be deducted for every day)