LAB 05: ABD QUORUMS
ALGORITMOS E SISTEMAS DISTRIBUÍDOS (ASD)

Alex Davidson

October 2023

a.davidson@fct.unl.pt

◇ Test 1 on Friday October 27th 2023

◇ Project Phase 2 starts next week

REMEMBER

◇ Use your source code in Phase 1 of the project to implement a `read/write` register, on top of an `ABD Quorum` design.

◇ More details at
https://github.com/MEI-ASD-2023/lab05-abd-quorum

◇ Will be useful for Phase 2 of the project

LAB FOR TODAY   3

ABD QUORUM REGISTER

◇ Particular case of state-machine replication

◇ Only supports two functions: read(), write(v)

◇ write(v) returns ACK upon termination

◇ read() returns previously written value, or initial value v_0

◇ Writes and reads can happen *concurrently*

READ/WRITE VARIABLE REPLICATION

◇ Quorum-based system

◇ **Assumption (Liveness)**: $2f+1$ correct processes, for f possible failures

◇ **Assumption**: Asynchronous

◇ **Assumption**: Reliable (pp2p) links

◇ Generalises to multiple values

> Identifiers for values, and parallel executions

ABD (ATTIYA, BAR-NOY, DOLEV) ALGORITHM 6

State for process i (st_i):

   $\diamond$ v_i: value, initially v_0

   $\diamond$ t_i: pair $(\mathrm{sn}_i, \mathrm{id}_i)$, initially $(0,0)$

   $\diamond$ $t_i > t_j$ iff $(\mathrm{sn}_i > \mathrm{sn}_j) \vee ((\mathrm{sn}_1 = \mathrm{sn}_2) \wedge (\mathrm{id}_1 > \mathrm{id}_2))$

STATE

Phase 1:

1. Send message read-tag() to all replicas

2. Wait for a quorum Q ($|Q| > 2f+1$) of replies

3. Let $sn_{max} = \max(\{sn_i\}_{t_i \in Q})$

Phase 2:

1. Send message write($t_v = (sn_{max}+1, id_j), v$) to all

2. Wait for a quorum Q ($|Q| > 2f+1$) of ACK

3. Return ACK (write considered terminated)

write(v) **ISSUED BY PROCESS** j 8

◇ Upon reception of $\boxed{\text{read-tag}()}$: return t_i

◇ Upon reception of $\boxed{\text{write}(t_v, v)}$:
  ▷ if $t_v > t_i$: $t_i = t_v$; $v_i = v$
  ▷ Return ACK

◇ Upon reception of $\boxed{\text{read}()}$: Return t_i, v_i

REPLICA PROCESS i

1. Send read() to all replicas

2. Wait for a quorum Q of replies $\{(t_i, v_i)\}_{i \in Q}$

3. Let $(t_{max}, v_{max}) = \max(\{t_i, v_i\}_{i \in Q})$

4. Return v_{max}

Question: Does this ensure **atomicity**?

READ ALGORITHM: NAÏVE ATTEMPT

Nope
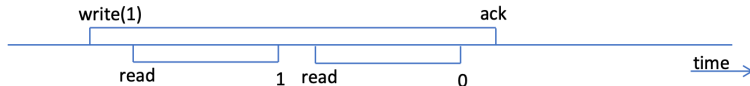
This execution can occur because t_{max} may be derived from a non-completed write.

READ ALGORITHM: NAÏVE ATTEMPT

Solution: add a `write-back()` phase

1. **Phase 1**: Proceed as in naïve case

2. **Phase 2**: Send $write(t_{max}, v_{max})$ to all replicas

3. Wait for a majority quorum of ACK

4. Return v_{max}

ABD READ ALGORITHM   12

Termination ($\leq f$ faults):

⬦ Assuming well-formed messages

⬦ Follows from properties of underlying links

◇ Serialisation points exist in **Phase 2** of
read() and write()

◇ Writes necessarily happen **before** reads

◇ Reads return most recent value by a
completed write(), otherwise read() `forces`
write() completion.

◇ Subsequent write() ops will not be impacted
by forced previous writes

◇ Serialisation points for write() are defined
between start and end of operation

ABD READ: ATOMICITY PROOF SKETCH

◇ "Sharing Memory Robustly in Message-Passing System" (JACM 1995)

◇ Edsger W. Dijkstra Prize in Distributed Computing: 2011

◇ Available here

MORE DETAILS