

Object implementations out of faulty base objects

Prof R. Guerraoui
Distributed Programming Laboratory



© R. Guerraoui

1



Failure modes

- ☞ Responsive: once \perp , forever \perp
- ☞ Non-responsive: no reply

NB. In the asynchronous model, it is impossible to distinguish a non-responsive failed object from a slow object

2

Register implementations

- ☞ Algorithm 1: implements a SWMR *register* out of $t+1$ SWMR base responsive failure-prone *registers*
- ☞ Algorithm 2: implements a SWSR *register* out of $2t+1$ SWSR base non-responsive failure-prone *registers*
- ☞ Algorithm 3: implements a *C&S* object out of $t+1$ base responsive failure-prone *C&S*

3

Responsive model

- ☞ Write(v)
 - ☞ For $j = 1$ to $(t+1)$ do
 - ☞ Reg[j].write(v);
 - ☞ return(ok)
- ☞ Read()
 - ☞ For $j = t+1$ to 1 do
 - ☞ $v :=$ Reg[j].read();
 - ☞ if $v \neq \perp$ then return(v)

4

Non-responsive model

- ☞ Init: seq := 1
- ☞ Write(v)
 - ☞ w_seq := w_seq + 1;
 - ☞ For $j = 1$ to $(2t+1)$ do **II**:
 - ☞ Reg[j].write(w_seq, v);
 - ☞ « wait until a majority of oks are returned »
 - ☞ return(ok)

5

Non-responsive model

- ☞ Init: (sn, val) := (-1, \perp);
- ☞ Read()
 - ☞ For $j = 1$ to $(2t+1)$ do **II**:
 - ☞ (s, v) := Reg[j].read();
 - ☞ (sn, val) := (s, v) with the highest s from majority, including (sn, val)
 - ☞ return (val)

6

Responsive model (single-shot compare&swap)

- ☞ C&S(v)
- ☞ $r := v$;
- ☞ for $j = 1$ to $t+1$ do
- ☞ $r' := \text{CS}[j].\text{C\&S}(r)$;
- ☞ if $r' \neq \perp$ then $r := r'$;
- ☞ return(r)

7

Exercises

- ☞ (1) Is it possible to build a **C&S** with base **C&S** objects among which one can be non-responsive?
- ☞ (2) Build a SWMR **register** that tolerates t non-responsive base SWMR **registers**

8