Failure Detectors (1)

- In this model the system is equipped with a distributed oracle (the failure detector) which provides indications (possibly not correct ones) on the processes which suffer a crash.
- Each process has access to a local module of the Failure Detector which monitors the other processes and inserts them in a proper list when suspects their crash. **(it is introduced to encapsulate some synchrony)**
- These are modules which can fail, therefore a process can be included in the list even if still progressing and not yet failed.
- *Rif. Bibl.: T. Chandra and S. Toueg, "Unreliable Failure detectors for Reliable Distributed Systems," Journal of the ACM, Vol. 43(2), pp. 225-267, Mar. 96.*

Failure Detectors (2)

- Suspected list are not static: if a local module realizes it has committed an error in suspecting a process it will update its list by adding or removing the involved process.
- Under this assumption there may exist 2 failure detectors with completely different lists, even in the same instant of time.
- We have also to underline that a mistake of a failure detector must not prevent correct processes to behave according to their specification.

FD and synchony

- How accurate is the information provided by a FD ?
- The more synchronous is the system the more accurate the information provided
- If the system is completely synchronous it is possible to design an FD able to provide perfect information
- If the system is only partially synchronous the accuracy can vary.
- It is prossible then to define several different specification of FD having different properties

Classes of Failure Detectors (1)

- Failure detectors can be partitioned in different classes according to the following properties:
  - Completeness: a failure detector eventually suspects every crashed process;
  - Accuracy: correctness of the suspect.
- More precisely it is possible to define two completeness and four accuracy properties thus having 8 different classes of failure detectors.

Completeness

- Completeness can be distinguished in:
  - **Strong completeness:** every process that crashed sooner or later gets permanently suspected by **EVERY** correct process.
  - **Weak completeness:** every process that crashed sooner or later gets permanently suspected **BY AT LEAST ONE** correct process.
- NOTE: satisfying the strong copmpleteness property is not sufficient to provide precise and correct information on the system behavior (a failure detector which suspects every process satisfies strong completeness …).
- It is necessary to limit the committed errors

Accuracy

- Four variants of the accuracy property are defined:

- **Strong accuracy:** no correct process is suspected before it crashes by any correct process
- **Weak accuracy**: some correct process is never suspected by any other correct process.
- **Eventual strong accuracy**: there is a time after which correct processes are not suspected by any other correct process.
- **Eventual weak accuracy**: there is a time after which some correct process is not suspected by any other correct process.

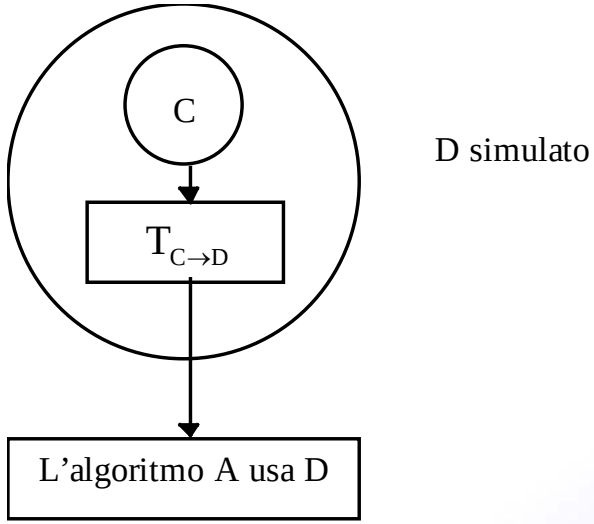
Classes of Failure detectors

- From the combination of properties we get the classes of Failure Detectors represented in the table :

Completeness	Accuracy			
	Strong	Weak	Eventual Strong	Eventual Weak
Strong	Perfect P	Strong S	Eventually Perfect ◊P	Eventually Strong ◊S
Weak	Q	Weak W	◊Q	Eventually Strong ◊W

Relations among the FD classes (1)

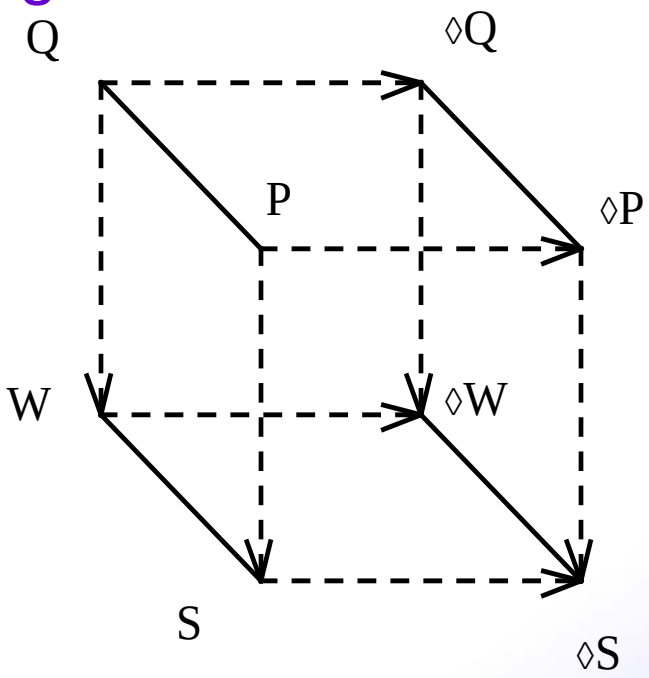
- The detector classes are not independent: by means of a reduction algorithm we can prove equivalence relations.



- The reduction algorithm is an 'emulator': if we can find an algorithm which transforms a FD C in another FD D this means that any problem that can be solved using D can also be solved using C. (C is able to behave as D)

Relations among the FD classes (2)

- If we have such reduction algorithm we say that D can be reduced to C (C ≥ D) or that D is weaker than C (in our case C offers at least the same information on processes offered by D)
- If C≥D and D≥C Then C and D are equivalent (C ≅ D).
- Among the classes of FD the following relations hold:



C - - -> D D è strettamente piu' debole di C

C ——— D C è equivalente a D

Consensus solved with a class S FD

- **Strong completeness:** every process that crashed sooner or later gets permanently suspected by **EVERY** correct process.
- **Weak accuracy**: some correct process is never suspected by any other correct process
- The algorithm develops in 3 phases
- In the first phase processes execute n-1 asynchronous rounds in which they broadcast their proposed values

- In each round every process P waits to receive a message from any other process which is not in its FDp (the list of processed suspected by the failure detector associated with process P) before moving to the next round.

Consensus solved with a class S FD

- In case P waits a message from Q it may happen that Q enters in FDp, In such a case P can move to the next round.
- At the end of Phase 2 the correct processes agree on a vector based on everyone proposal (the i-th position in this vector contains the value proposed by process I or the null value ⊥).
- In phase 3 process P decides the first non null value of is copy of the vector Vp.

Algorithm S

- 01 procedure propose(v_p)
- 02 V_p:= ($\perp$,$\perp$,...$\perp$) – p estimate of the proposed values
- 03 $V_p[p]$:=v_p
- 04 D_p := V_p
- 05 **phase1**: {asynchronous rounds r_p , $1 \leq r_p \leq n-1$}
- 06 for r_p :=1 to n-1
- 07 send(r_p, D_p,p) to all
- 08 wait until ($\forall$q: received(r_p, D_q , q) or q $\in$ FD_p)
- 09 $msgs_p(r_p)$:={(r_p , D_q , q) | received (r_p, D_q, q)}
- 10 D_p := ($\perp$,$\perp$,..$\perp$)
- 11 for k:=1 to n
- 12 **if** Vp(k) = $\perp$ **and** E (rp , Dq , q) $\in$ msgsp(rp) **with** Dq(k) not= $\perp$ **then**
- 13 Vp(k):= Dq(k)
- 14 Dp(k):= Dq(k)
- 15 **phase2:** send Vp to all
- 16 **wait until** ($\forall$ q: received Vq **or** q $\in$ FDp)
- 17 lastmsgsp := {Vq | received Vq}
- 18 **for** k:=1 **to** n
- 19 **if** $\forall$ Vq $\in$ lastmsgsp **with** Vq(k) = □ **then** Vp(k):= $\perp$
- 20 **phase3**: decide (first non $\perp$ component of Vp)

Solution with a FD of class ◊S (1)

- If the maximum number pf processes that may fail is less than half of the totality of processes it is possible to solve consensus with a failure detector of class ◊S (which satisfies the *Eventual weak accuracy* and the *Strong completeness*).

- The algorithm is based on the paradigm of coordinator rotation: every process performs in its turn the role of coordinator to determine a value that can be chosen among the various proposals.

- If the coordinator is correct and is not suspected by the non faulty processes then it will succeed in identifying such a value and will then perform a reliable broadcast of such value.

Solution with a FD of class ◊S (2)

- All the message exchanges are performed between the coordiantor (whose identidy is known: during round r the coordinator will be process **(r mod n) +1**) and the other processes.
- Also in this case we have the execution of many rounds (in each the coordinator changes) each of which is divided in 4 phases.
- In phase 1 each process sends to the coordiantor its own estimate of the value that should be dcided (its proposal) with a time stamp indicating the round number in which such an estimate has been taken.
- In Phase 2 the coordinator collects the majority of such estimates, selects the one with the biggest time-stamp and sends it to every process proposing it as the new estimate.

Solution with a FD of class ◊S (3)

- In phase 3 each process p has two possibilities:
  - 1) to receive the estimate from the coordinator and send to it an ack to inditate its adoption of the suggester value; or
  - 2) by checking it FD module suspect the crash of the coordinator and send a nack to it.
- In phase 4 the coordinator waits
- ⌈(n+1)/2⌉ responses (ack o nack).

- If all the responses are positive the coordinator knows that a majority of processes has changed their estimation adopting the proposed value, consequently it sends (through an R-Broadcast) the request to decide according to this result, which is then done by every process when executing the R-delivery of such proposal.

Class ◊W

- The solution shown for the class ◊S is important as class ◊S is equivalent to class ◊W.
- class ◊W is the WEAKEST class of FD which allows a solution of the consensus problem.
- To recap failure detectors with a *Perpetual Accuracy* (classes P, Q, S, W) solve consensus in asynchronous systems without limits on the number of failed processes, while failure detectors with an *Eventual Accuracy* (classes ◊P, ◊Q, ◊S, ◊W)require a majority of correct processes.

L'Early Consensus (1) qui

- The' Early Consensus algorithm has been proposed as an alternative to the one described for the failure detector of class ◊S.
- There is also here a rotation of the coordinator, but the message exchange is simplified.
- In round r the coordinator p_c proposes its estimate as the value to decide, however the algorithm does not use an exchange of ack (or nack) to reach a decision.
- When a process P receives a value from p_c it forwards it to every other process. By doing so, if P receives the same value from the majority of the processes the process can directly decide for it.
- If the coordinator is not suspected the protocol terminated directly at the first round.

L'Early Consensus (2)

- If the coordinator crashes or is suspected by a majority of the processes, all the processes move to the next round after having updated the estimates, ensuring so that if some process has decided for the estimated of the coordinator at round r every process that moved to round r+1 has this value as its own estimate..
-
- This way the property of Uniform Agreement is satisfied.

- Rif Bibl. *A. Schiper, "Early Consensus in an Asynchronous System with a weak Failure Detector", Distributed Computing, Springer-Verlag, Vol. 10, pp. 149-157, 1997*

Considerations

- The Failure detector model offers relevant hints for analysis and investigations. However its main limitation is given by the fact that FD have been proposed and analized considering abstract properties rather than specific implementations (hw or sw).
- This model presents also some limits in its applicability as it cannot manage the dynamic entrance or leave of the processes in the system, but most of all it exposes the system to indefinite blockings in instability situatuons.
- A contribution to make the model more ‘practical’ has been given in
- Chen, W., S. Toueg, and M. K. Aguilera. 2002. "On the Quality of Service of Failure Detectors." IEEE Transactions on Computers 51 (5): 561-580. doi:10.1109/TC.2002.1004595.