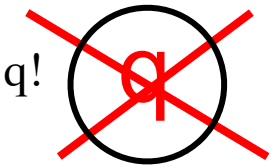
Failure detectors

Carole Delporte
IRIF/GANG
Université Paris Diderot

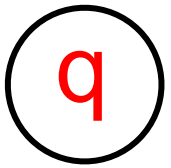
Asynchronous distributed system with process failures

There is no bound on the time it takes for a process to execute a computation step, or for a message to go from its sender to its receiver

don't wait for q!



wait!



SLOW

Some problems cannot be solved

Group Membership, Group Communication, Atomic Broadcast, Primary/ Backup systems, Atomic Commitment, Consensus, Leader election….

Consensus

- Processes have an input value and have to decide on a common value:
  - *agreement*: if two processes decide they decide the same value
  - *validity*: if a value is decided then this value has been proposed by some process
  - *termination*: every correct process decides
- [FLP 85]: impossibility to solve consensus in asynchronous system even if there is at most one crash

How to circumvent these impossibility results?

- Give to processes enough information on failures

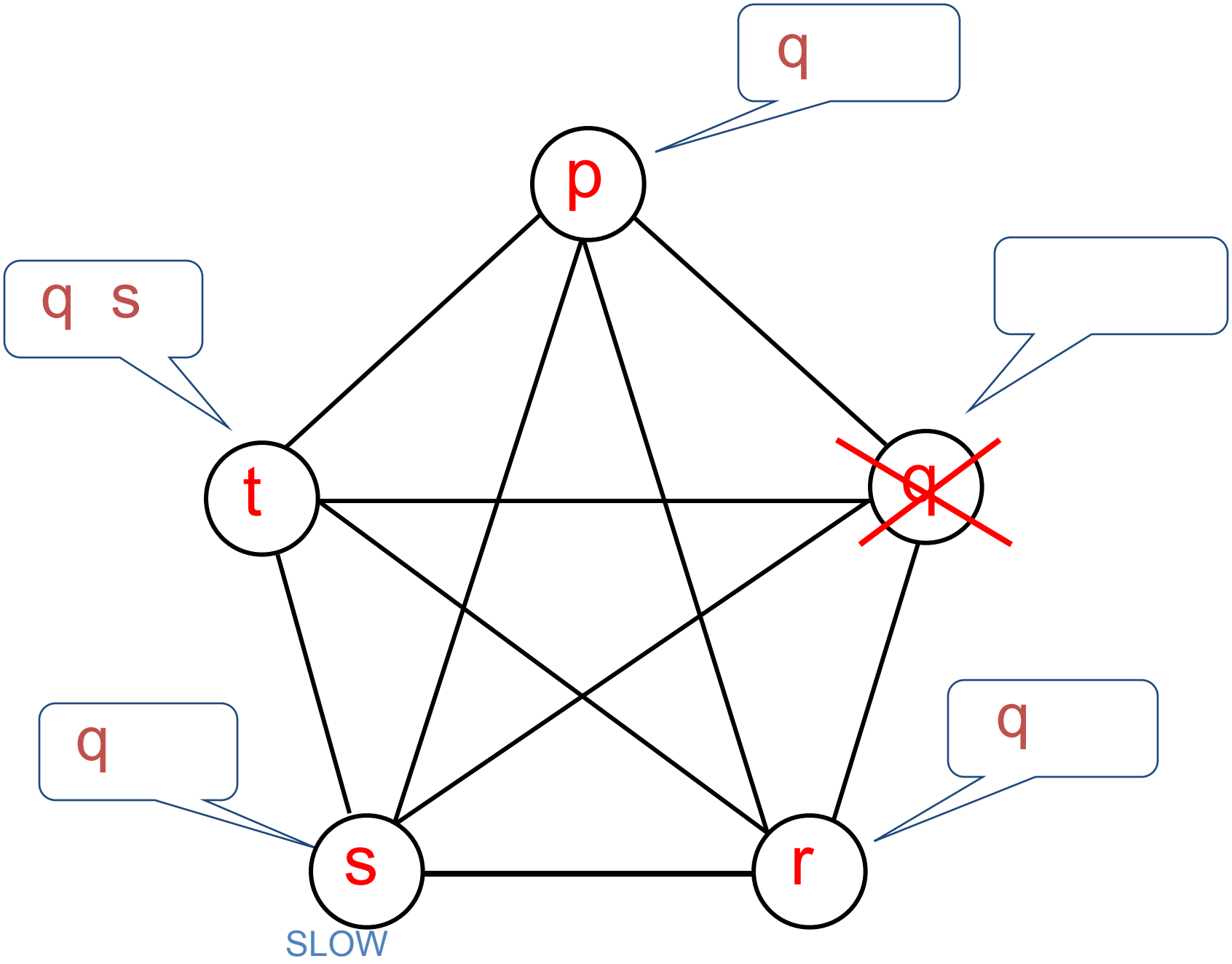
  (

  - change the model (partially synchronous models, add some objects)
  - change the specification : randomization, input vectors [MRR03]

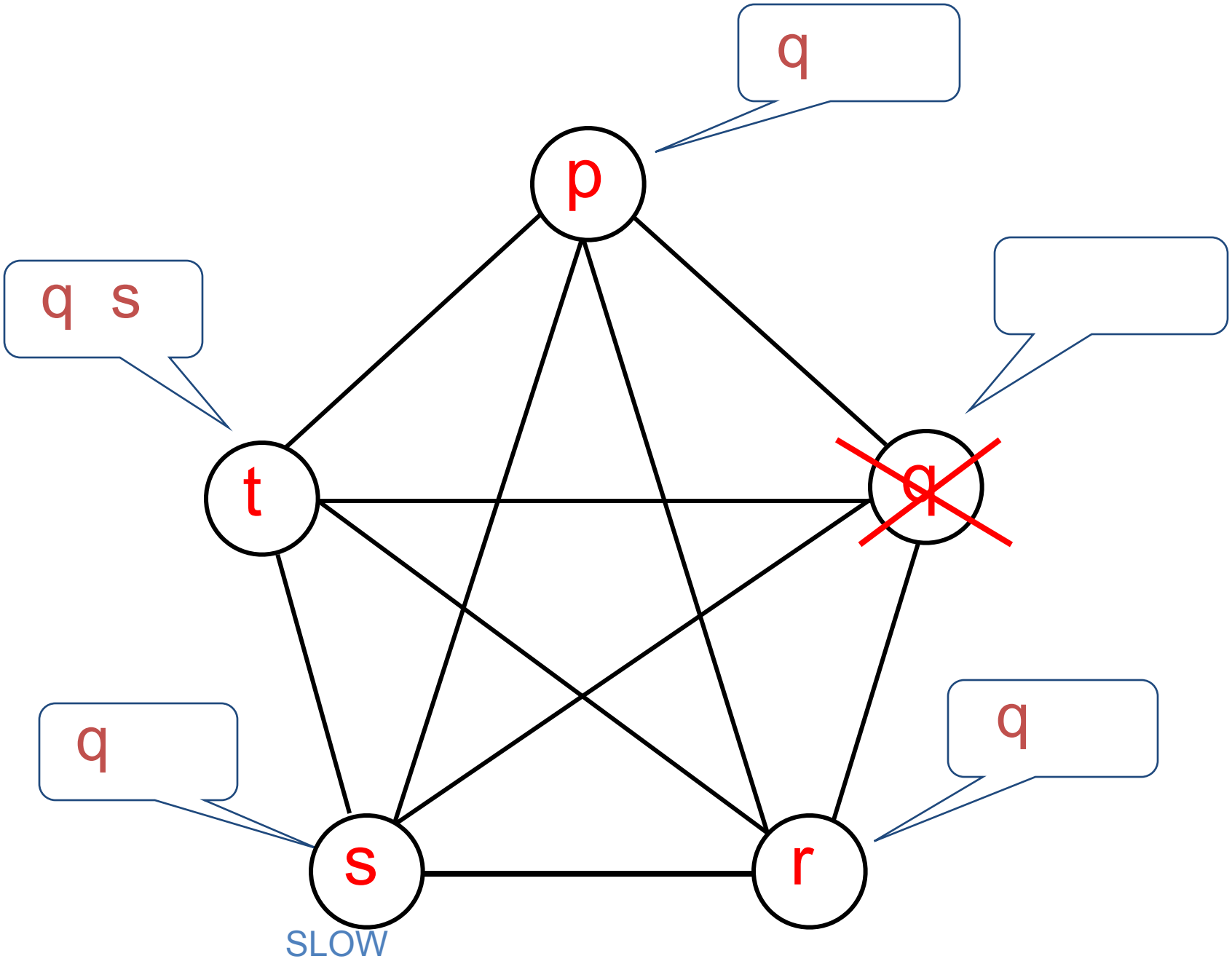
)

Failure Detector Chandra&Toueg

- PODC 91 - J. ACM 96
- Distributed oracles that give hints on the failure pattern (e.g set of suspected processes)
- A FD is basically defined by:
  - a completeness property: actual detection of failure
  - an accuracy property: restrict the mistake that a FD can make


q
p
q s
t
q
q
s
r
q
SLOW

- $\mathcal{P}$: perfect (each crashed process will be suspected, no correct process is suspected)
- $\mathcal{S}$: strong (each crashed process will be suspected, at least one correct process is never suspected)


q
p
q s
t
q
q
s
r
q
SLOW

- $\mathcal{P}$: perfect (each crashed process will be suspected, no correct process is suspected)
- $\mathcal{S}$: strong (each crashed process will be suspected, at least one correct process is never suspected)
  - here directly if a FD is perfect it is strong:

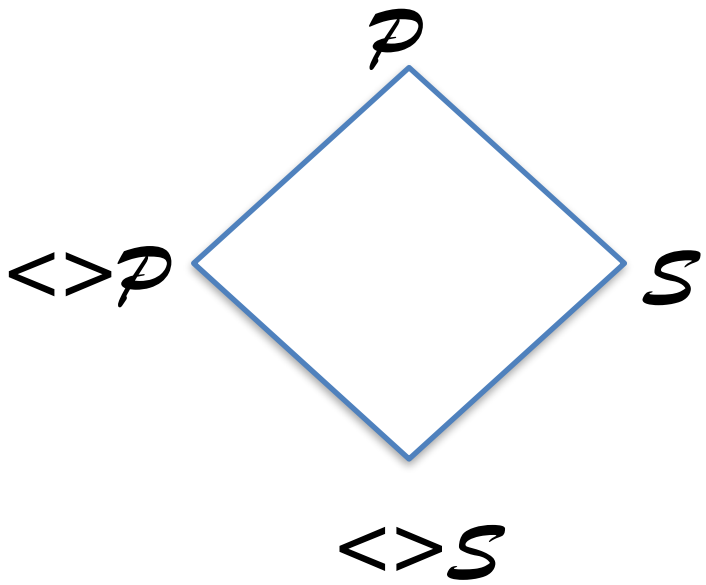
$\mathcal{P}$ is « stronger » than $\mathcal{S}$

Comparison:

- $\mathcal{D} \geq \mathcal{D}'$ if there exists a (distributed) algorithm where processes query $\mathcal{D}$ and output $\mathcal{D}'$

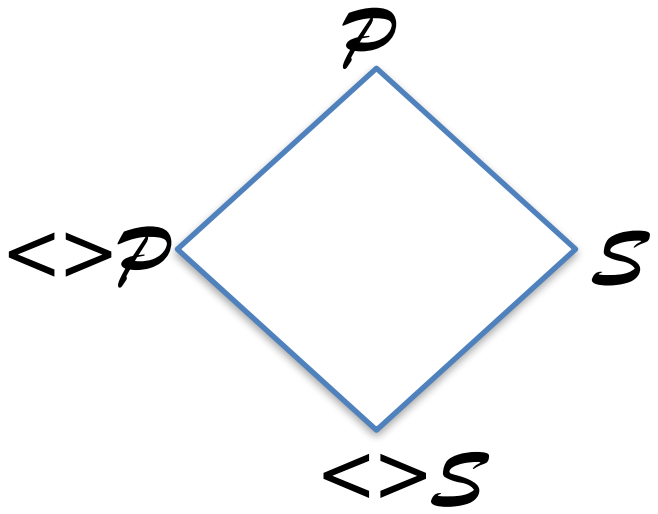
(trivially $\mathcal{P} \geq \mathcal{S}$ but not $\mathcal{S} \geq \mathcal{P}$)

- $\Diamond p$: eventually p
- $\Diamond s$: eventually s



Hints about failure

- How many « information » about failures ?
  - Sufficient : with the oracle the problem can be solved



  - Necessary: with less information the problem cannot be solved

weakest failure detector

- Problem $\mathscr{P}$
- Define a failure detector $\mathscr{D}$
  - $\mathscr{D}$ can solve $\mathscr{P}$
  - if $\mathscr{D}'$ solves $\mathscr{P}$ then $\mathscr{D}' \geq \mathscr{D}$

$\mathscr{D}$ is a weakest failure detector to solve $\mathscr{P}$

A weakest failure detector encapsulates the exact amount of information on failure necessary and sufficient to solve a problem

weakest failure detector?

- Given problem $\mathscr{P}$ and problem $\mathscr{P}'$, wk($\mathscr{P}$) (wk($\mathscr{P}'$))the weakest failure detector for $\mathscr{P}$ (for $\mathscr{P}'$)
- if wk($\mathscr{P}$)<wk($\mathscr{P}'$) (wk($\mathscr{P}$)≤wk($\mathscr{P}'$) and not wk($\mathscr{P}'$)≤ wk($\mathscr{P}$)) then $\mathscr{P}'$ is harder than $\mathscr{P}$ ($\mathscr{P}'$ needs more information about failure than $\mathscr{P}$)
- *chase the weakest failure detector for problems (…many papers…)*

weakest failure detector?

Every *problem* that is solvable with a failure detector has a weakest failure detector [JT08]

Problem: (1) given a run, it is possible to determine whether the problem requirements are met in the run, and (2) an algorithm is considered to solve the problem if every run of the algorithm meets the problem requirements.

Weakest: Chandra, Hadzilacos&Toueg

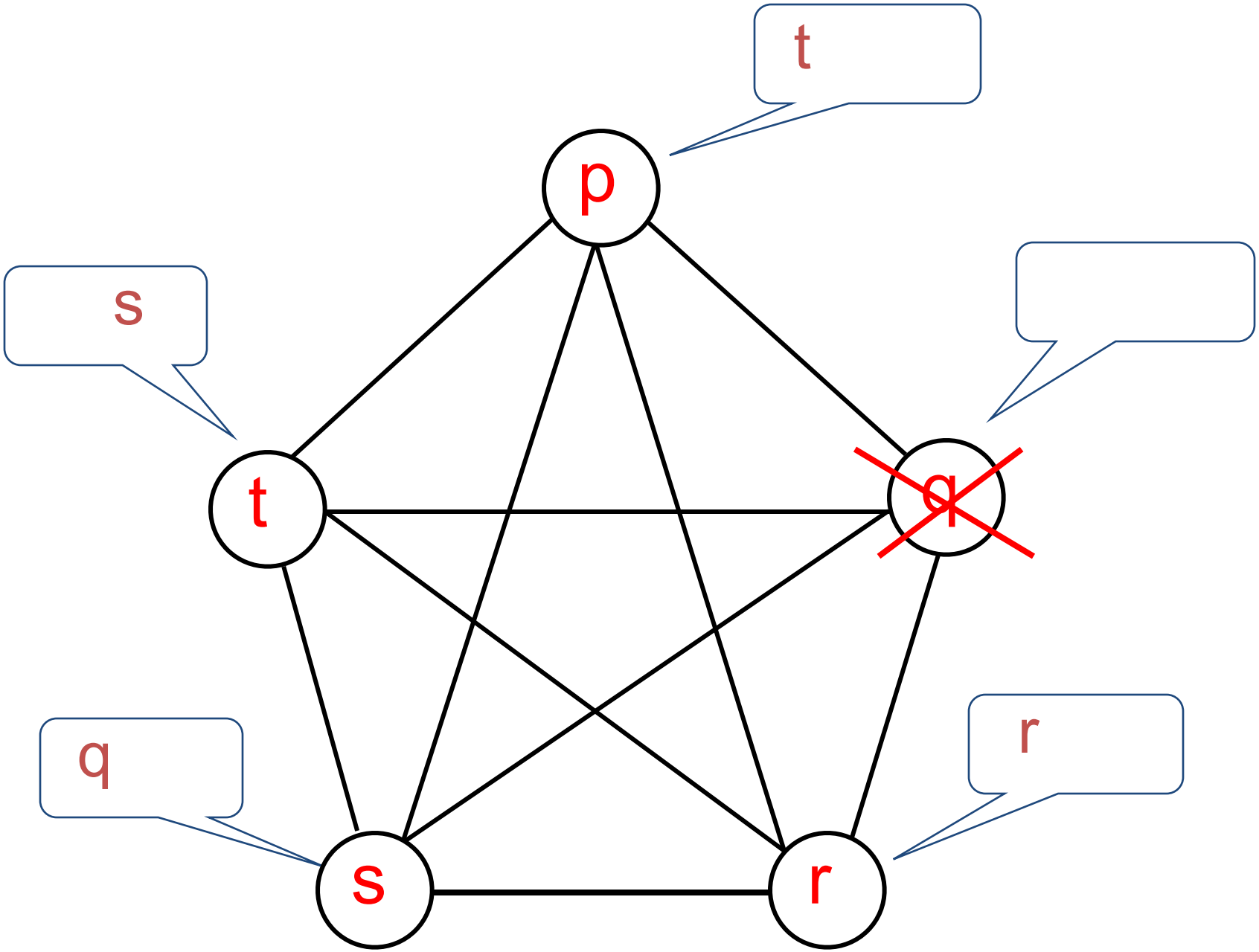
- PODC 92 - J. ACM 96

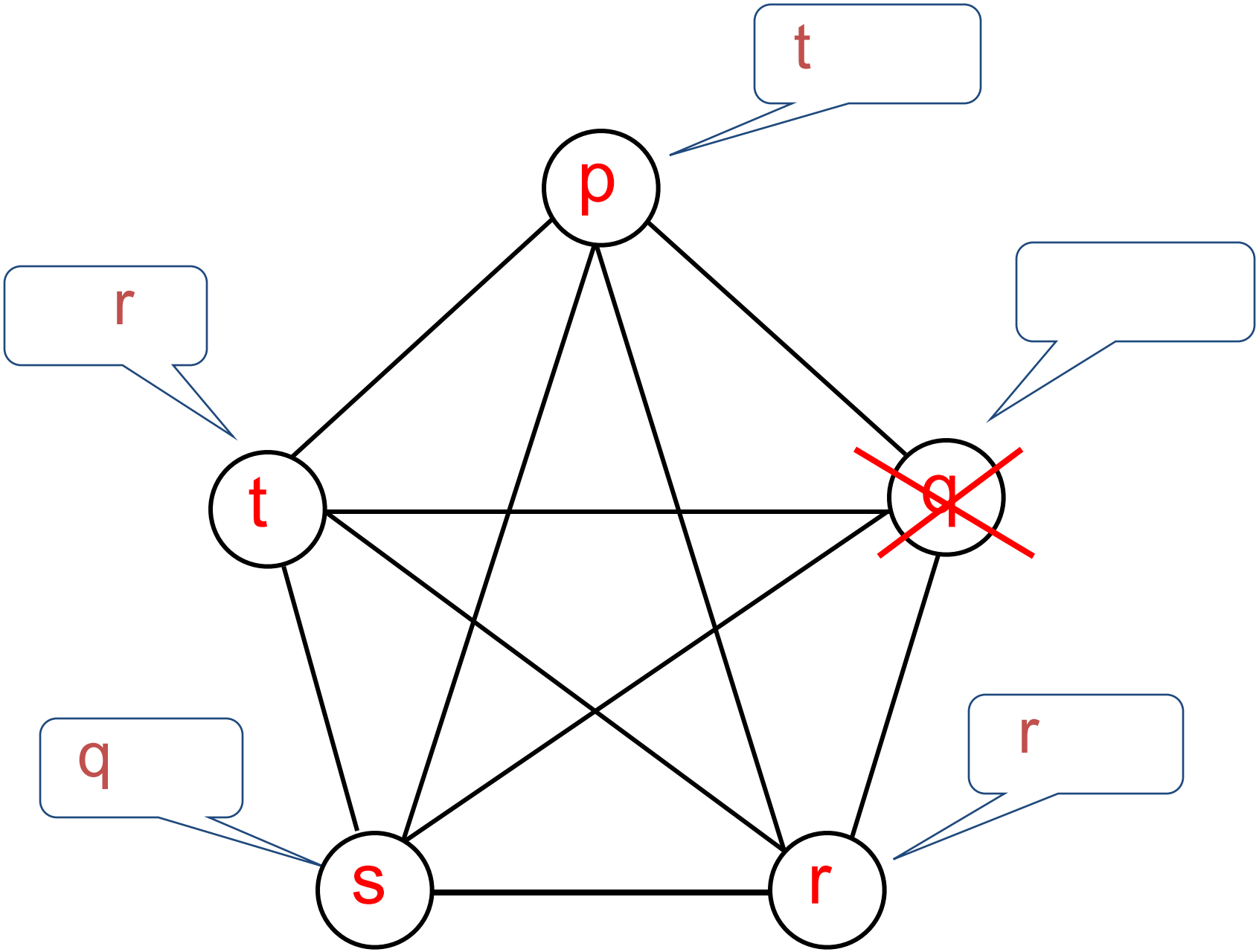
- Weakest failure detector to solve consensus in message passing with a majority of correct processes: leader

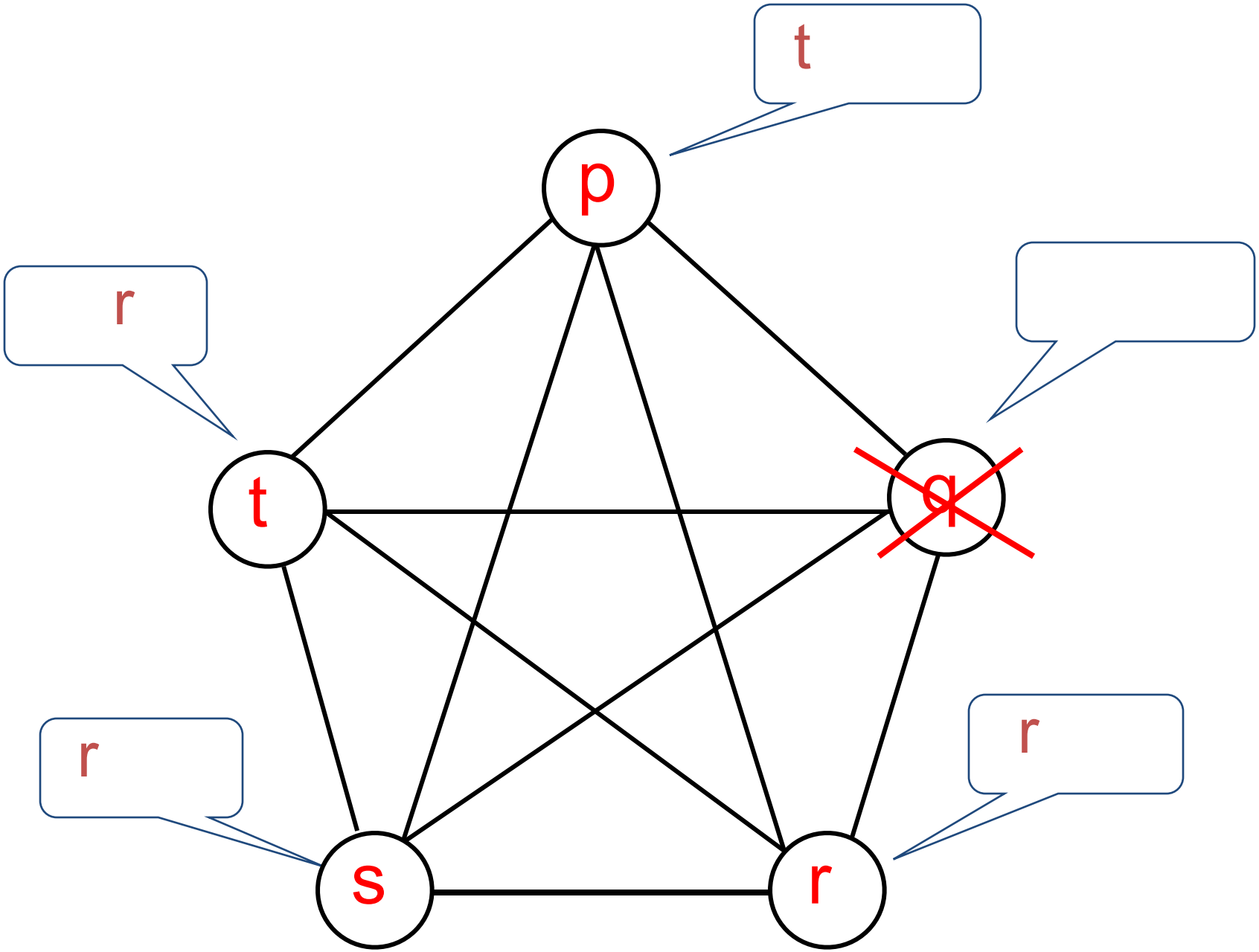
Leader Ω

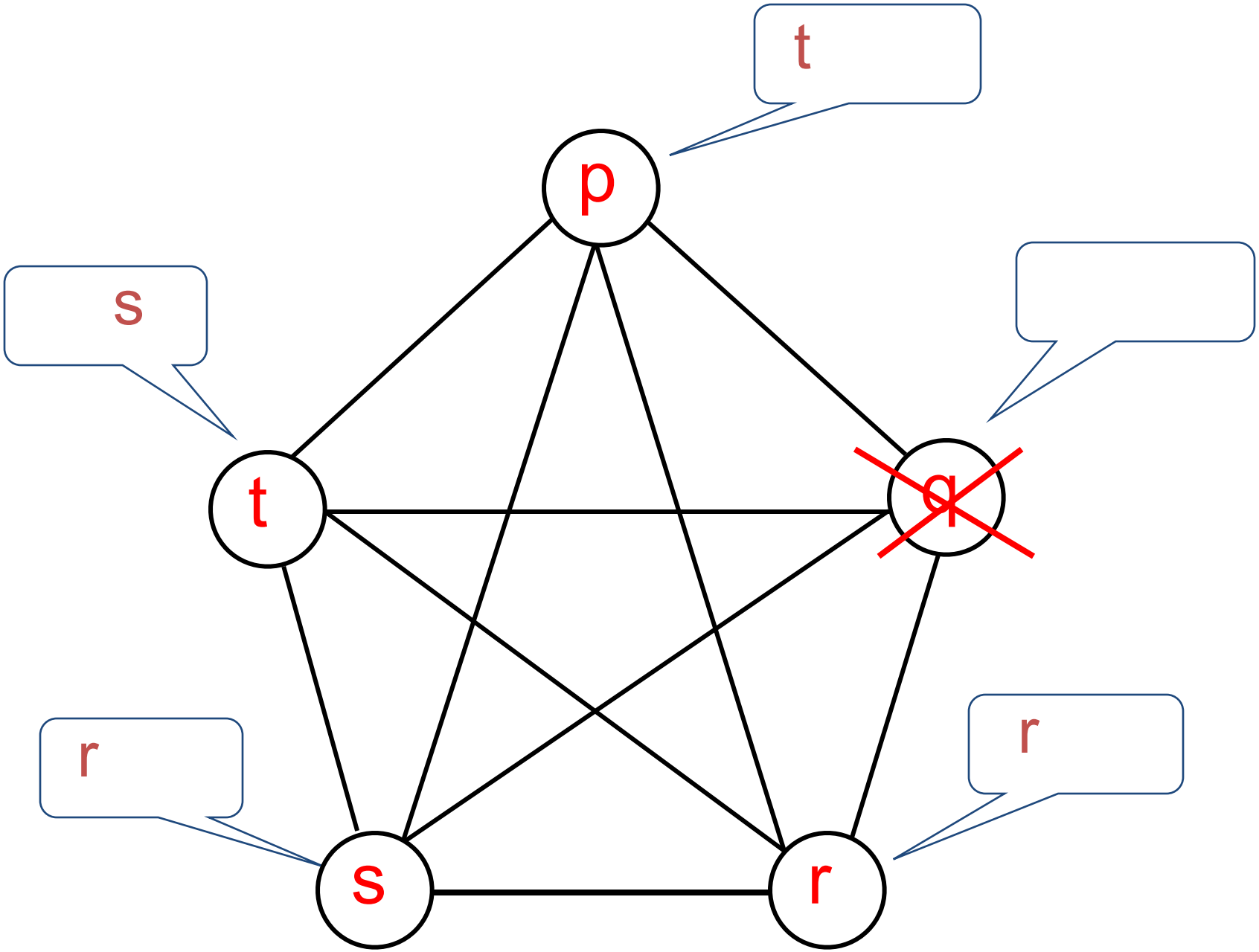
- There exists a correct process *q* and a time *t* after which such that every correct process trusts *q*

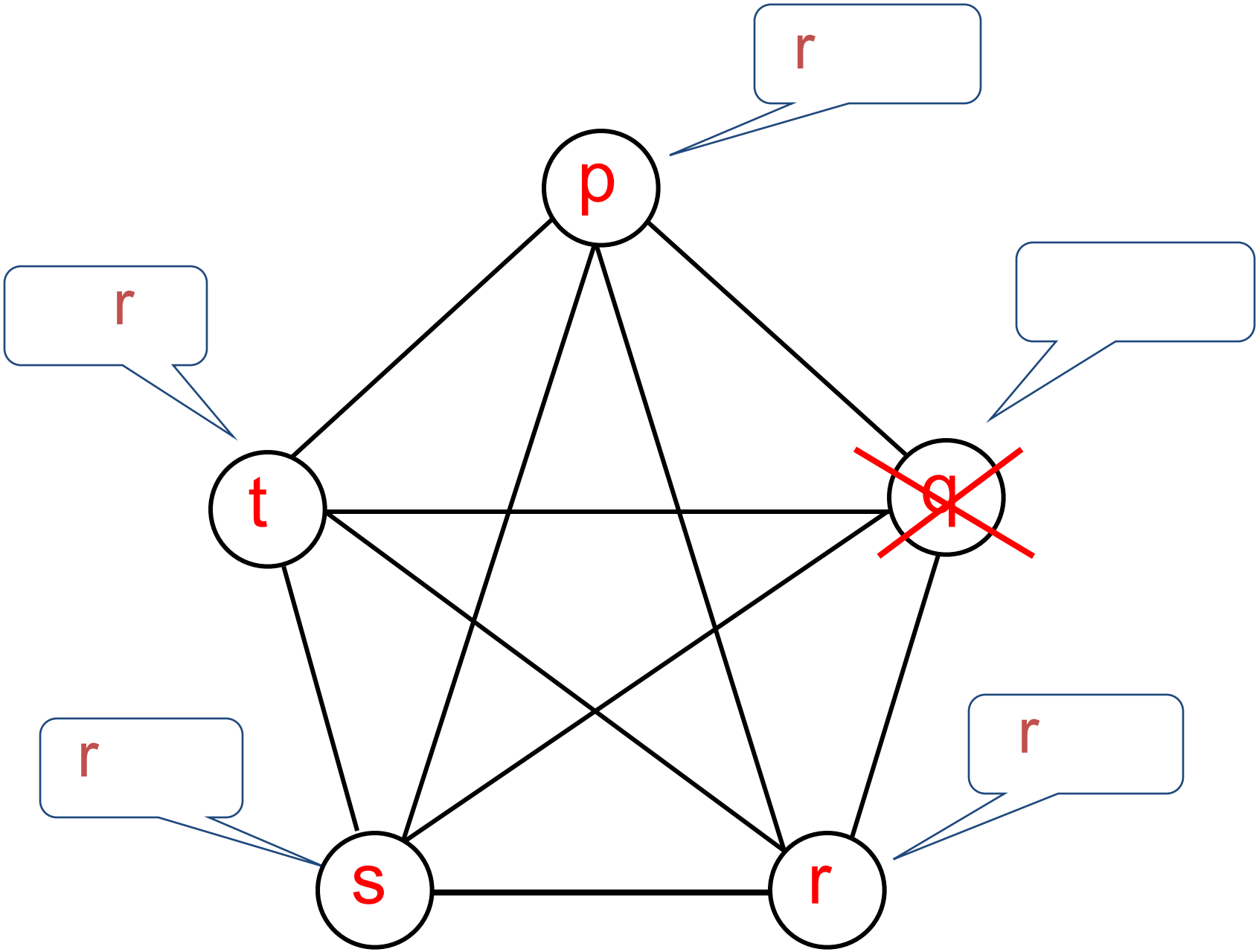
- *The output of the failure detector is one process*
- *Eventually the output of all processes is the same correct process*


t
p
s
t
q
q
r
s
r


t
p
r
t
q
q
r
s
r


t
p
r
q
t
r
r
s
r


t
p
s
q
t
r
r
s
r


r
p
r
r
t
q
r
r
s
r

Solve consensus with **a majority of correct processes**

- Rotating coordinator
- Hurfin&Raynal DC 1999

  A Simple and Fast Asynchronous Consensus Protocol Based on a Weak Failure Detector
- Mostefaoui & Raynal DISC 1999

  Solving Consensus Using Chandra-Toueg's Unreliable Failure Detectors: A General Quorum-Based Approach

Function Consensus(v_i)
cobegin

(1) **task** $T1$: $r_i \leftarrow 0$; $est_i \leftarrow v_i$; % $v_i \neq \perp$ %
(2) **while** *true* **do**
(3) $c \leftarrow (r_i \bmod n) + 1$; $est\_from\_c_i \leftarrow \perp$; $r_i \leftarrow r_i + 1$; % round $r = r_i$ %
(4) **case** $(i = c)$ **then** $est\_from\_c_i \leftarrow est_i$
(5) $(i \neq c)$ **then wait** ((EST(r_i, v) is received from p_c)$\vee$($c \in suspected_i$));
(6) **if** (EST(r_i, v) has been received) **then** $est\_from\_c_i \leftarrow v$
(7) **endcase**; % $est\_from\_c_i = est_c$ or $\perp$ %
(8) $\forall j$ **do** *send* EST$(r_i, est_from_c_i)$ *to* p_j **enddo**;
(9) **wait until** ($\forall p_j \in Q_i$: EST(r_i, est_from_c) has been received from p_j);
% Q_i has to be a *live* and *safe* quorum %
% For $\mathcal{S}$: Q_i is such that $Q_i \cup suspected_i = \Pi$ %
% For $\diamond\mathcal{S}$: Q_i is such that $|Q_i| = \lceil (n+1)/2 \rceil$ %
(10) **let** $rec_i = \{est\_from\_c \mid$ EST(r_i, est_from_c) is received at line 5 or 9$\}$;
% $est\_from\_c = \perp$ or v with $v = est_c$ %
% $rec_i = \{\perp\}$ or $\{v\}$ or $\{v, \perp\}$ %
(11) **case** $(rec_i = \{\perp\})$ **then skip**
(12) $(rec_i = \{v\})$ **then** $\forall j \neq i$ **do** *send* DECIDE(v) *to* p_j **enddo**; **return**(v)
(13) $(rec_i = \{v, \perp\})$ **then** $est_i \leftarrow v$
(14) **endcase**
(15) **enddo**

(16) **task** $T2$: **upon reception of** DECIDE(v):
$\forall j \neq i$ **do** *send* DECIDE(v) *to* p_j **enddo**; **return**(v)

coend

If $\mathscr{D}$ can be used to solve consensus then $\mathscr{D} \geq \Omega$

- Processes construct a directed acyclic graph (DAG) that represents a "sampling" of failure detector values in the run and some temporal relationships between the values sampled.
- This DAG can be used to simulate runs of Consensus with $\mathscr{D}$
- By considering several initial configurations of Consensus, we obtain a forest of simulated runs of Consensus
- It is possible to extract the identity of a process p that is correct in the run (the same correct process for all processes)

Weakest (shared memory)

Ω: consensus [HL 94]

Anti-Ω: set-agreement [Zielinski10]

Michel Raynal, in the rump session of PODC 2007, conjectured that a generalization of anti-Ω is the weakest failure-detector for n-process k-set agreement and proposed k-anti-Ω outputs, when queried, a set of n – k processes such that, eventually, at least one correct process is never output.

k-anti-Ω: k-set agreement [GK09] [FRT09][DFGT09]

Weakest (message passing)

Ω: consensus (majority of correct processes) [CHT96]

Σ : atomic register [DFG10]

Σ x Ω: consensus [DFG10]

L : set agreement [DFGT08]

Chasing the Weakest Failure Detector for k-Set Agreement in Message-Passing Systems….[BR09] [MRS12]

Open problem

Implementation

- implementing a FD in (as weak as possible) partially synchronous system

implementing Ω in message passing

- links are eventually timely [LAF99]
- only the output links of an unknown correct process are eventually timely [ADFT08]
- based on a query-response mechanism and assumes the responses from some processes to a query arrive among the (n – t) first ones [MMR03]

Conclusion