16-124-836
Marcel ZAUDER

3.1 Relations among Failure Detectors

We can implement a perfect failure detector *PFD* by using a non-perfect failure detector *NFD* with strong accuracy and weak comleteness as follows:

Building an PFD out of an NFD
<u>Init:</u> $\mathit{alive} := \Pi$ $\mathit{detected} := \emptyset$ <u>upon</u> $\langle \mathit{NFD.Crash} \mid p \rangle$: $\mathit{detected} := \mathit{detected} \cup \{p\}$ $\mathit{alive} := \mathit{alive} \setminus \{p\}$ trigger $\langle p\ \mathit{crashed} \rangle$ to all $p' \in \Pi$ <u>upon</u> $\langle p\ \mathit{crashed} \rangle$ from p': $\mathit{detected} := \mathit{detected} \cup \{p\}$ $\mathit{alive} := \mathit{alive} \setminus \{p\}$

Additionally we can use the fact that if a process p running *NFD* detects a process p' to have crashed, because of the strong accuracy property, p' has actually crashed.

3.3 Quorum Systems

- **SINGLETON**
  There mustn't be any failed process. Because:

$$\nexists p \in \Pi : (\nexists Q \in \mathbb{Q} : p \in Q)$$

- **MAJORITY:**
  MAXIMUM/MINIMUM: $\lfloor \frac{n-1}{2} \rfloor$, because:

$$\forall Q \in \mathbb{Q} : \mid Q \mid = \lceil \frac{n+1}{2} \rceil$$

$$\Rightarrow \mid \Pi \mid - \mid Q \mid = n - \lceil \frac{n+1}{2} \rceil = \lfloor \frac{n-1}{2} \rfloor$$

- **GRID:**

  MINIMUM:
  We take the Q with the fewest elements, which would be equal to the last row of the grid. Therefore we must have k correct processes and at most $k^2 - k$ faulty processes.

  MAXIMUM:
  We take the Q with the most elements which would be equal to the last row with $k-1$ additional processes. Therefore we must have in total $2k-1$ correct processes, so at most $k^2 - 2k + 1$ faulty processes.

3.2 Perfect Failure Detector

First we assume that we have access to a timer, which has the follwing properties:

1. Each timer carries a reference to a process p, which is emitted as a parameter in its timeout event
2. Each such timer can be acessed via its parameter p.
3. Each timer can be reset - that is resetting its countdown to the value it was started with.

With this information, we can construct a perfect failure detector using unidirectional HEARTBEATS:

Implementing a perfect failure detector with heartbeats

<u>Init:</u>
  alive $:= \Pi$
  detected $:= \emptyset$
 	
for $p \in (\Pi \setminus \{itself\})$:
  Per process timer to track lifetime of remote process
  $startTimer(\Delta + \Phi + 1,\ p)$
  Start timer to peridically send own heartbeats and send initial heartbeat
  $startTimer(\Delta + \Phi,$ itself)
  trigger $\langle p, \text{HEARTBEAT} \mid \textit{empty} \rangle$ to all $p' \in \textit{alive} \setminus \{\textit{itself}\}$
 	
<u>upon</u> $\langle \textit{timeout} \mid p \rangle$:
  if $p == \textit{itself}$
    resetTimer(itself)
    trigger $\langle p, \text{HEARTBEAT} \mid \textit{empty} \rangle$ to all $p' \in \textit{alive} \setminus \{\textit{itself}\}$
  else:
    detected := detected $\cup \{p\}$ alive := alive $\setminus \{p\}$ trigger $\langle P.Crash \mid p \rangle$
 	
<u>upon</u> $\langle p, \text{HEARTBEAT} \mid \textit{empty}$
  Restart timer tracking lifetime of p
  resetTimer(p)

Because every process (which is alive) sends a HEARTBEAT in the beginning and every $\Delta + \Phi$ and every HEARTBEAT takes at most $\Delta + \Phi$ to arrive and be processed a process from which no HEARTBEAT was received for $\Delta + \Phi + 1$ is guaranteed to be not alive and have crashed.