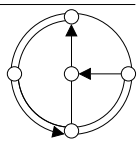
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich



SS 2004 Prof. R. Wattenhofer / Dr. C. Cachin / F. Kuhn / R. O'Dell

Principles of Distributed Computing
Exercise 7: Sample Solution

1 Failure Detectors

a) All nodes regularly (always after time τ) send an *alive* message to all other nodes. Then, no node has to wait longer than $\tau + \Delta$ to receive a message of a correct server.

Algorithm 1 Code for P_i:

```
1: D_i := {1, ..., n}
2:
3: // Thread 1:
4: while true do
5:    send alive to all servers;
6:    sleep(τ)
7: end while
8:
9: // Thread 2:
10: upon receiving alive from P_j, remove j from D_i
11: when more than τ + Δ time has passed since the last (alive) message from P_j was received,
    add j to D_i
```

b) See Algorithm 2 on the next page.

2 Timed Reliable Broadcast

First note that as soon as a correct server r-delivers a message m, all other correct server have r-delivered a message m after time $d\Delta$ because every two correct servers are connected by a path of at most length d consisting only of correct servers.

Suppose that the sender is faulty. It may be that he still manages to send a message to some of its neighbors before he fails. Like that, the message can first be sent from one faulty server to another until reaching a correct server after at most f steps. Thus, if a there is a correct server which r-delivers a message m, there must be a correct server which r-delivers a message after at most time $f\Delta$.

Adding the two times ($f\Delta$ and $d\Delta$), we get the $(f + d)\Delta$-Timeliness.

Algorithm 2 Code for P_i:

1: $\mathcal{D}_i := \emptyset$
2: $\Delta :=$ default time-out interval
3:
4: *// Thread 1:*
5: **while true do**
6: **send** *alive* to all servers;
7: **sleep**(τ)
8: **end while**
9:
10: *// Thread 2:*
11: **while true do**
12: **for all** $j \in \{1, \ldots, n\}$ **do**
13: **if** $j \notin \mathcal{D}_i$ and P_i did not receive *alive* during the last $\tau + \Delta$ ticks of P_i's clock **then**
14: $\mathcal{D}_i := \mathcal{D}_i \cup \{j\}$ *// time-out: P_i suspects P_j has crashed*
15: **end if**
16: **end for**
17: **end while**
18:
19: *// Thread 3:*
20: **upon receiving** *alive* from P_j:
21: **if** $j \in \mathcal{D}_i$ **then**
22: $\mathcal{D}_i := \mathcal{D}_i \setminus \{j\}$;
23: $\Delta := \Delta + 1$
24: **end if**