Errata for

Introduction to Reliable and Secure Distributed Programming

Christian Cachin, Rachid Guerraoui, Luís Rodrigues

Second Edition

Springer, 2011, XIX, 320 pages
ISBN-10: 3-642-15259-7
ISBN-13: 978-3-642-15259-7

The most recent version of this file can be found on the web under

`http://www.distributedprogramming.net/`

The errors are classified into three categories:

- **Typo.** A typographical error that does not affect correctness or understanding.
- **Minor.** A minor technical or textual error that is easy to spot and to fix.
- **Major.** A major technical error that requires significant change.

Date of last change: November 20, 2015

Page 32, line 3 (typo)

Change "represents" to "stands".

Page 61, first text paragraph, line 6 (typo)

Change "trust" to "trusts".

Page 113, last paragraph, line −3 (from bottom) (typo)

In "Processes q and r receive" change r to s.

Page 119, 4th para., lines 4 and −2; page 120, 2nd para., lines 3 and 7 (typo)

Change every "*bcr*-delivers" to "*brb*-delivers".

Page 218, last paragraph, line −3 (from bottom) (typo)

Change "every initializes" to "every process initializes".

Page 219, Algorithm 5.5 and paragraph below (major)

There was an omission that could cause the algorithm to violate the *eventual leadership* property. In Algorithm 5.5, change the second event handler, which starts with **upon event** ⟨ *Ω*, *Trust* | *p* ⟩ **do** to this

```
upon event ⟨ Ω, Trust | p ⟩ do
    if $p \neq$ trusted then
        trigger ⟨ pl, Send | trusted, [NACK] ⟩;
    trusted := p;
    if $p = self$ then
        $ts := ts + N$;
        trigger ⟨ beb, Broadcast | [NEWEPOCH, $ts$] ⟩;
```

In the paragraph below Algorithm 5.5, line 5, change the text starting from "Consider any . . . " until the end of the paragraph on the next page to

> Consider any correct process p that receives this message. We distinguish two cases: (1) Suppose p last trusted q and $qts >$ *lastts*, where *lastts* denotes the variable of p. Then p starts epoch (q, qts) as required. One of two things may happen now. If (1a) p continues to trust q forever, then p may not start any further epoch with a different leader according to the algorithm; hence, the property holds. However, it may be (1b) that p trusts some process $r \neq q$ later, which is a prerequisite for it to start any epoch with a leader different from q. But then, p sends a NACK message to q at least once, according to the revised algorithm. This message causes q to increment its variable ts and to broadcast another NEWEPOCH message. When p receives it, then p either trusts q and starts the epoch with leader q, and the property holds; or p sends another NACK message to q and the NACK/NEWEPOCH exchange with q repeats. Since q trusts itself forever and p eventually trusts q forever, this process may terminate only by p starting an epoch with leader q as required. This ensures that the last epoch started by p has leader q.
>
> On the other hand (2), process p may not trust q or $qts \leq$ *lastts* when p delivers the NEWEPOCH message with timestamp qts. Then it sends a NACK message to q and the *eventual leadership* property follows analogous to case (1b) before.
>
> It remains to show that every process eventually starts *some* last epoch. The properties of *Ω* ensure that eventually all correct processes trust q forever; after this time, only q may increment its ts variable and no other process broadcasts NEWEPOCH messages. Consider the last NACK message that is delivered to q. Then, q broadcasts a NEWEPOCH message with a timestamp qts^* to all processes. Because q is correct, all correct processes deliver this message and the epoch with timestamp qts^* is the last epoch that every correct process starts.

Page 236, last paragraph, line -2 (from bottom) (typo)

Change “cloud” to “could”.

Page 252, last line of text (typo)

Delete “also” and insert before “UNDEFINED” the text “$\perp$ and different from”.

Page 252, line 18 of Algorithm 5.17 (typo)

In the last clause, in the line “**if exists** $ts \geq 0, v \neq \perp$ from S such that …”, replace “S” with “*states*”.

Page 255, line 6 (typo)

Throughout the paragraph starting with “As we will see, the inputs …”, replace “S” with “*states*”.

(It would be preferrable to harmonize this notation everywhere in description of the “Byzantine Read/Write Consensus” algorithm. This would mean to replace every occurrence of variable S in pages 252–259 with *states*.)

Page 256, line 14-15 (minor)

In the paragraph of lines 6–15 that ends with “… and only the writeset of s changes to $ws'_s = \{(6, w)\}$,” replace this text with “… and the writeset of r changes to $ws'_r = \{(6, x)\}$ and the writeset of s changes to $ws'_s = \{(6, w)\}$.”

Moreover, in Figure 5.6 (page 256), replace ws.r with ws.r’.

Page 256, last line (minor)

In the equation $S = \dots$, replace ws_r with ws'_r.

Page 266, Exercise 5.8 (typo)

In the first sentence, change the first occurence of “Algorithm 5.6” to “Algorithm 5.7.” In the last sentence, also change “Algorithm 5.6” to “Algorithm 5.7.”

Pages 270–271, Algorithm 5.22–5.23 (minor)

In Algorithm 5.22 in line 12

```
    estimate := $\perp$; votes := $[\perp]^N$;
```

replace “$[\perp]^N$” with “$[\text{UNDEFINED}]^N$”.

In Algorithm 5.23 in lines 3–4

```
upon $\#(votes) > N/2 \wedge sentvote = \textsc{True}$ do
    $V := \{v \mid \text{there exists } p \in \Pi \text{ such that } votes[p] = v\}$;
```

replace "$\#(votes)$" with "$\#(\{p \in \Pi \mid votes[p] \neq \textsc{undefined}\})$" and add "$\wedge\, v \neq \textsc{undefined}$" after "$votes[p] = v$".

Page 285, line 23 of Algorithm 6.1 (minor)

Two lines near the end of Algorithm 6.1 need adjustment. Specifically, the commands

```
$delivered := delivered \cup decided$;
$unordered := unordered \setminus decided$;
```

must be indented by one more level (so that they are executed inside the **forall** loop) and be replaced with

```
    $delivered := delivered \cup \{m\}$;
    $unordered := unordered \setminus \{(s, m)\}$;
```