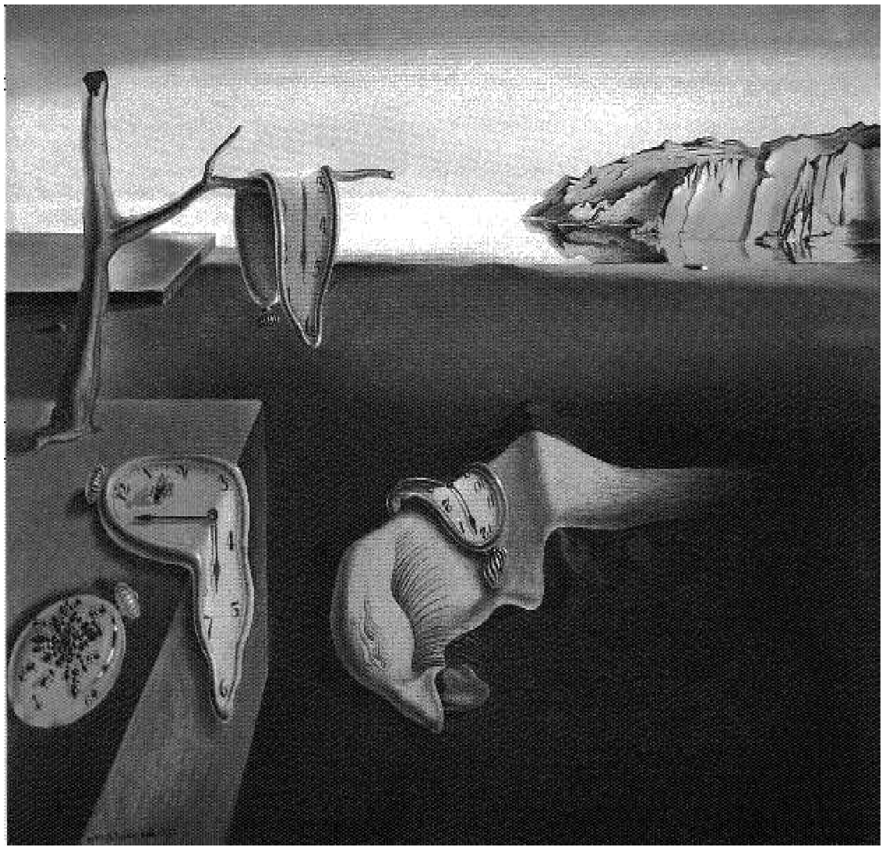
Institut für Automation
Abt. für Automatisierungssysteme

Technische Universität Wien

Projektbericht Nr. 183/1-123
October 2002

An Algorithm for Three-Process Consensus Under Restricted Link Failures

Günther Gridling


Salvador Dali, "Die Beständigkeit der Erinnerung"

An Algorithm for Three-Process Consensus Under Restricted Link Failures

Günther Gridling

Abstract

This technical report presents an algorithm that solves consensus among three processes under restricted omissive link failures. The problem was posed by Schmid and Weiss [1], who distinguished two types of systems, R-type resp. S-type, which are characterized by the fact that out of the three processes, in each round one may fail to receive (R-type) resp. send (S-type) values from resp. to its peers. In both systems, there is one process which never fails, but the processes do not know which one. Whereas one can give an impossibility proof for the R-type system, it turns out that the S-type system has a solution which even works in a slightly weaker link failure setting. This report presents an algorithm that solves the consensus problem in the S-type system and proves that the algorithm does indeed achieve consensus.

Definition 1 (System Model) *We assume three fully connected processes A, B, C. In any round, messages from $A \rightarrow B$ and $B \rightarrow A$ may be dropped at will. In addition, one of the messages $A \rightarrow C$ or $B \rightarrow C$ may be dropped, but not both in the same round. Messages from C must always arrive at the destination. Initially, none of the processes knows whether it is A, B, or C.*

In the following text, we will call C the *good process* or the *master*.

Definition 2 (Consensus) *An algorithm solves consensus if it fulfills the following properties:*

Agreement: *All processes decide on the same value.*

Validity: *If all processes start with the same value v, then the decision must also be v.*

Termination: *All processes eventually decide.*

This report presents a solution to the consensus problem under the system model of Definition 1. The general idea of the algorithm is as follows:

Since messages sent from the master never fail, once a process has experienced at least one fault at each of its two incoming links, it can safely assume that it must be the master. With that knowledge, it can send a message forcing the other processes to decide upon its current value: Since it is the master, both messages are guaranteed to arrive, and consensus is achieved.

If the master does not become aware of its special status, however, we cannot achieve consensus so easily. In this case, though, at least one of the master's incoming links must never fail, since the occurrence of at least one fault at each of the incoming links leads to the above situation. Therefore, since the master never suffers any send failure, at least one of its links must be bidirectional. This, however, improves the communication structure enough for consensus to be achieved. Let w.l.o.g. the bidirectional link be between processes B and C, then B and C can synchronize their input sets, and A receives the result over the non-faulty link $C \rightarrow A$.

In the following, we will describe the algorithm in detail.

Algorithm 1 (Algorithm $\mathcal{A}$) *The algorithm requires at most eight rounds and has two main phases: One in which the decision value is computed from all three input values, and one in which it is computed from only two of the input values.*

The first part of Figure 1 describes the initialization of variables. The value set $\mathcal{V}_p$ of process p is set to its own initial value v_p, and the flag $rec3$ which indicates whether the process has received a 3-value decision is set to false. The set $\mathcal{L}_p$ of processes which committed a link failure is initially empty.

The remainder of Figure 1 describes general actions executed by the algorithm. Computation of a decision value depends on the cardinality of the input set $\mathcal{V}_p$: If there are three values in the set, the majority is used as the decision value. If there are only two values, we use either the majority if it exists, or the default value 0.

If a process p experiences a link failure from q in any round except master rounds, then it puts q into its link failure set $\mathcal{L}_p$.

A process always checks its link failure set $\mathcal{L}_p$ first thing in a round and determines it is the master if the set contains two processes. In this case, the master computes its own decision value and sends this value as the master value *to its peers. The master then delivers the master value and halts.*

If a process p receives a master message, *i.e., a message from the master process containing a master value, then p immediately delivers the master value and halts.*

Figure 2 shows how processes compute a three-value decision dec3. In the first round, each process p sends its own initial value to its peers, and collects all received values in its input value set $\mathcal{V}_p$. In the second round, each process sends this value set to its peers and combines foreign sets with its own one.

```
  Initialization (Round 0):
$\mathcal{V}_p := \{v_p\}$; [input value set]
$\mathcal{L}_p := \emptyset$; [link failure set]
$rec3$ := **false**; [three-valued decision flag]
$dec := \emptyset$; [decision value]

  Computation of decision value:
if $|\mathcal{V}_p| = 3$
    return majority($\mathcal{V}_p$);
fi
if $|\mathcal{V}_p| = 2$
    return majority($\mathcal{V}_p$); [if it exists]
    else return 0; [no majority in $\mathcal{V}_p$]
fi

  Reaction to link failures at end of rounds except master rounds:
if not received message from $q$
    $\mathcal{L}_p = \mathcal{L}_p \cup \{q\}$;
fi

  Reaction to master status (at start of each round):
if $|\mathcal{L}_p| = 2$
    $dec$ := decision value; [master value]
    send $dec$; [master message]
    deliver $dec$;
    halt;
fi

  Reaction to a master message (done immediately):
if received master message
    $dec$ := master value;
    deliver $dec$;
    halt;
fi
```

Figure 1. *Initialization and general actions.*

At the beginning of the third round, each process checks whether it has three values in its set. If it has, then it computes the 3-value decision dec3 and sends it to its peers. If it does not have three values, then it just sends an empty message. Processes which receive a message containing a dec3 value store this value and set rec3 to true.

In the fourth round, each process which has obtained a dec3 message in the previous round forwards the message to its peers. Processes *which obtain such a message store the value and set rec*3 *to true.*

*In the fifth round, each process which has obtained a dec*3 *message in round 4 forwards this message to its peers, and processes receiving such messages store the value and set rec*3 *to true.*

*The sixth round is a master round, in which processes only wait for master messages but take no own actions except to send a master message if necessary. If no master message has arrived until the end of this round, then each process whose rec*3 *is set to true will deliver the 3-value decision stored in dec and halt. As we will show in Lemma 4, either all or none of the processes will deliver dec*3 *at this point.*

*If the processes cannot decide after round 6, then they must continue with the algorithm to decide on a common two-value decision. As Figure 3 outlines, each process p whose input value set $\mathcal{V}_p$ only consists of two values will compute a 2-value decision dec*2 *from it and send this decision to its peers. Nodes whose set has three values will send an empty message here. If a process receives such a two-value decision, it stores the value.*

*The last round 8 is again a master round, in which processes simply wait for a master message but do not send any messages of their own (except master messages). If no master message has arrived by the end of the round, the processes will deliver their decision value. If the algorithm proceeds this far, then all processes will deliver the same dec*2 *value here, as we will prove in Lemma 4.*

In the following, we will show that the above algorithm fulfills Agreement, Validity, and Termination. To prove this, we will prove each of the properties separately. But first, we will show some fundamental properties of the algorithm.

Lemma 1 *If C does not know it is the good process at the end of round r, then at least one of the links $A - C$ or $B - C$ must have been bidirectional up to and including round r.*

```
  Round 1 and 2 [send value set]
send $\mathcal{V}_p$;
if receive message containing $\mathcal{V}_q$
    set $\mathcal{V}_p = \mathcal{V}_p \cup \mathcal{V}_q$;
fi

  Round 3 [compute and send dec3]
if $|\mathcal{V}_p| = 3$ [three-valued decision]
    send decision value;
else
    send empty message;
fi
if received message containing value $dec3$
    $dec := dec3$;
    $rec3 :=$ true;
fi

  Round 4 [forward dec3 from Round 3]
if received $dec3$ in R3
    send $dec3$;
else
    send empty message;
fi
if received $dec3$
    $dec := dec3$;
    $rec3 :=$ true;
fi

  Round 5 [forward dec3 from R4]
if received $dec3$ in R4
    send $dec3$;
else
    send empty message;
fi
if received $dec3$
    $dec := dec3$;
    $rec3 :=$ true;
fi

  Round 6 [master round]

  Round 7 [in fact, end of Round 6]
if $rec3 =$ true
    deliver $dec$;
    halt;
fi
```

Figure 2. *Three-valued decision.*

```
  Round 7 [compute and send dec2]
if $|\mathcal{V}_p| = 2$
    send decision value;
else
    send empty message;
fi
if received message containing value $dec2$
    $dec := dec2$;
fi

  Round 8 [master round]

  Round 9 [in fact, end of Round 8]
deliver $dec$;
halt;
```

Figure 3. *Two-valued decision.*

Proof: If a process experiences a link failure from another process p, then it knows that p cannot be the good process. Therefore, as soon as C has seen a link failure from both its peers it knows that neither of them is the good process and can conclude that it must be the good process itself. So if C does not know it is the good process by round r, then it has not seen a link failure from at least one of its peers. Since its own links to the peers are always correct, there must have been at least one bidirectional link up to and including round r. □

In our proofs we will assume w.l.o.g. that the link $B - C$ is bidirectional.

Lemma 2 *After the second round of algorithm $\mathcal{A}$, either C knows that it is the master process, or A has the value set $\{v_A, v_B, v_C\}$ and B, C either have $\{v_A, v_B, v_C\}$ or $\{v_B, v_C\}$.*

Proof: Assume that C does not know it is good after round 2. Then according to Lemma 1 one of C's links must have been bidirectional. Assume w.l.o.g. that $B-C$ has been bidirectional. In the first round of $\mathcal{A}$, each process sends its own value to its peers. Therefore, at the end of the first round, A must have at least the set $\{v_A, v_C\}$, B must have at least $\{v_B, v_C\}$ and C must also have at least $\{v_B, v_C\}$. In the second round, the sets are exchanged and each process merges the set it receives with its own set of round 1. Therefore, at the end of the second round, A must have the full set $\{v_A, v_B, v_C\}$ and B, C will at least keep their set $\{v_B, v_C\}$. Of course, if the link from A to B or C works, then one or both might also get the full set. □

With these two lemmas, we are ready to prove the consensus properties of algorithm $\mathcal{A}$.

Lemma 3 (Validity) *Algorithm $\mathcal{A}$ fulfills Validity.*

Proof: From Figure 1 we see that every decision is computed from the input value set of the processes which only consists of initial values. The default value 0 is only used if there is no majority. But as long as all processes have the same initial value, the decision will be that value. □

Lemma 4 (Agreement) *Algorithm $\mathcal{A}$ fulfills Agreement.*

Proof: We will show that

1. if one process delivers the master value, then all do,
2. if one process delivers $dec3$ after R6, then all do,
3. if one process delivers $dec2$ after R8, then all deliver the same $dec2$.

1) If one process delivers the master value, then all do.

Only the good process C can send a master message because only the links to the good process may both fail. The master is the first process to deliver the master value at the beginning of the round, and its message must arrive at both A and B. If a process receives a master message, then it immediately delivers the master value and halts, regardless of its current state. Therefore, A and B will deliver the master value in the same round as the master. After a master round (R6, R8) a process cannot determine that it is master, therefore C can only become master at the start of R3-R6 or R8. Hence, its master message must arrive in these rounds. Since decisions are only made after master rounds and since a master message takes precedence over a decision, in the former case (arrival in R3-R6) the master message will override any decision $dec3$. In the latter case, either a decision $dec3$ has already been made and the algorithm will not come to R7, or the master message will arrive before the end of R8 and will override any $dec2$ decision.

2) If one process delivers $dec3$ after R6, then all do.

We have already seen in part (1) of the proof that if a process becomes master in R3-R6, then there will not be a $dec3$ decision. Therefore, if one process delivers $dec3$, then there is no master until the start of R6. From Lemma 1 we know that in this case there must have been at least one bidirectional link. In order for a process p to deliver $dec3$, it must have received at least one $dec3$ in R3-R5. Assume that the initial sender of $dec3$ in R3 has been q. Of course there may be more than one sender, but a process only sends a $dec3$ message in round R3 (in contrast to forwarding it in R4 and R5), and it only does so if its value set has contained three values. In order for a process to get a $dec3$ message, there must have been at least one sender of a $dec3$ message in R3.

We have three possible cases for the sender q:

- If $q = C$, then A and B get $dec3$ in R3 and C gets at least one $dec3$ in R4. So, by the end of R4, all processes have received $dec3$.

- If $q = B$, then at least C gets $dec3$ in R3 and sends it to A and B, and again all processes have received $dec3$ by the end of R4.

- If $q = A$, then theoretically neither B nor C may get $dec3$. But in this case there will be no forwarded $dec3$ message in R4 and no process will get $dec3$ by the end of R5. Therefore, either B or C must receive $dec3$ from A in R3. If C gets $dec3$, then in R4 A and B also get $dec3$. If B gets $dec3$, then in R4 at least C gets $dec3$ due to the bidirectional link $B - C$ and in R5 A gets $dec3$ as well. So all processes have received $dec3$ by the end of R5.

Hence, if one process delivers $dec3$ then all have received $dec3$ and will deliver it as well.

3) If one process delivers $dec2$ after R8, then all deliver the same $dec2$.

We have already seen in part (1) of the proof that if a process becomes master in R8, then there will not be a $dec2$ decision. Therefore, if one process delivers $dec2$, then there is no master until the start of R8 and there must be a bidirectional link $B - C$ by Lemma 1. From Lemma 2 we know that there is only one possible two-value set, $\{v_B, v_C\}$, which can at most be possessed by B and C. If we arrive in R7, then there was no $dec3$ value. From part (2) of the proof and from the above text we can see that in this case, only A has possessed a three-value set and was unable to send $dec3$ in R3. Therefore, both B and C have $dec2$ and will send it in R7. Hence, both A and B get the $dec2$ from C, and due to the bidirectional link C must get the (same) $dec2$ from B. Therefore, by R8 all processes have received $dec2$ and will decide on $dec2$. □

Lemma 5 (Termination) *Algorithm $\mathcal{A}$ fulfills Termination.*

Proof: From the proof of Agreement, we can see that if there is no $dec3$ decision, then there must be a $dec2$ decision at the end of R8. Both decisions can only be overruled by a master decision at the end of R8 or earlier. So the algorithm terminates at the latest after round R8. □

Remarks:

1. Note that R7-R8 can be executed in parallel with rounds R5-R6 if we assume that sending a $dec3$ message takes precedence over sending a $dec2$ message and that we always deliver a $dec3$ decision if $rec3$ is true. Hence, the algorithm in fact requires only 6 rounds.

2. The algorithm was formulated for binary consensus. To achieve multi-valued consensus, one should change the computation of the decision value in Figure 1 to deliver an appropriate default value whenever there is no majority (which can now also occur for $|\mathcal{V}_p| = 3$).

References

[1] U. Schmid and B. Weiss. Consensus with oral/written messages: Link faults revisited. Technical Report 183/1-110, Department of Automation, Vienna University of Technology, Feb. 2001.