

## TERMINATING RELIABLE BROADCAST AND CONSENSUS WITH ARBITRARY FAILURES

When processes are subject to arbitrary failures the specification of Terminating Reliable Broadcast (TRB) and Consensus must be weakened somewhat to make sense. The problem is that the Integrity property of TRB makes reference to the message broadcast by a potentially faulty sender and the Validity property of Consensus makes reference to values proposed by potentially faulty processes. In the context of arbitrary failures it is not clear what it means for a faulty sender to broadcast a message or for a faulty process to propose a value. We can think of the message broadcast by the sender or the value proposed by a process as being given to it by a source external to the system. But in the case of arbitrary failures the process can behave “as if” the input given to it by the external source were something different than the one that was really given to it, thereby undermining the intent of these properties.

To avoid this, when dealing with arbitrary failures in TRB the Integrity requirement becomes simply:

- *Integrity*: Every *correct* process delivers at most one message.

Thus if the sender is faulty then any message whatever can be delivered by the correct processes, as long as they all agree on what message to deliver. Similarly, in Consensus we replace Validity by the following variant:

- *Validity*: If all *correct* processes propose the same value  $v$  then no correct process decides a value other than  $v$ .

It is easy to see that with this specification Consensus is unsolvable for arbitrary failures if  $n \leq 2t$  (why?). The revised specification of TRB, on the other hand, does not place any a priori constraints on the number of faulty processes that can be tolerated. But as we will see, in the case of arbitrary failures without authentication, TRB and Consensus are both unsolvable if  $n \leq 3t$ .

In Section 1 of these notes we present two algorithms for TRB in synchronous systems with arbitrary failures — one that needs message authentication and one that does not. The algorithm that needs message authentication works for any number of faulty processes. The algorithm that does not need message authentication requires that  $n > 3t$ . The transformation of TRB to Consensus that we have seen in the context of benign failures can be modified so that it works even with arbitrary failures (with or without message authentication) assuming  $n > 2t$  (how?). Therefore, the first TRB algorithm can be transformed to a Consensus algorithm for arbitrary failures with message authentication when  $n > 2t$ . As mentioned before, the requirement  $n > 2t$  is unavoidable for Consensus in this failure model. Also, the second TRB algorithm can be transformed to a Consensus algorithm for arbitrary failures without message authentication when  $n > 3t$ .

In Section 2 we prove that in the model of arbitrary failures without message authentication the requirement that  $n > 3t$  is unavoidable: No algorithm solves Consensus in that model, if  $n \leq 3t$ . Since, as we mentioned earlier, TRB can be transformed to Consensus if  $n > 2t$ , the same result is

true for that problem: No algorithm solves TRB for arbitrary failures without message authentication if  $n \leq 3t$ . Thus the TRB algorithm for arbitrary failures without message authentication presented in Section 1 has optimal fault tolerance.

## 1 Algorithms for TRB

### 1.1 An algorithm with authenticated messages

The algorithms for TRB that we have seen so far tolerate only benign failures (i.e., up to and including general omission failures) and so do not have to contend with the possibility of a faulty sender sending contradictory messages or faulty receivers conveying false information. This becomes an issue once we allow arbitrary failures. However, if we also assume the existence of a mechanism whereby correct processes can attach unforgeable signatures to messages they sent, the ability of faulty processes to lie is considerably limited. In particular, even though a faulty sender can lie arbitrarily and send conflicting messages, other processes cannot lie about information they have received from correct processes. This makes it relatively easy for correct processes to detect inconsistencies that a faulty sender might have tried to inject, and to deliver **SF** in that case.

First we explain in more detail the assumption of message authentication and its consequences. If  $s$  a string (say, the text of a message) and  $p$  is a process, we let  $s : p$  denote the result of process  $p$  applying its signature to  $s$ . Note that the string  $s$  could itself be the result of some process  $p'$  signing string  $s'$  — i.e.,  $s = s' : p'$  — and so on. We assume the following two properties regarding signed messages.

- (a) A process  $q$  that receives a message of the form  $s : p$  can extract  $s$  from it.
- (b) If  $p$  is a correct process then a process  $q$  (whether correct or faulty) cannot send a message that contains in it the string  $s : p$  unless  $p$  has sent  $s : p$  in the past.

The first assumption states that the recipient of a signed message can extract the “body” of that message. By induction, this assumption implies that if  $q$  receives a message of the form  $s : p_1 : p_2 : \dots : p_k$ , i.e. a message (supposedly) successively signed by  $p_1, p_2, \dots, p_k$ , it can extract  $s$  as well as the identities of the signers. The second assumption states that no process can forge the signature of a correct one. Note that faulty processes are permitted to collude and forge each other’s signature. Techniques from public key cryptography, such as the RSA scheme, can be used to implement signatures schemes with the above properties.

The algorithm we are about to describe (due to Pease, Shostak and Lamport) solves TRB for arbitrary failures assuming message authentication. Certain messages sent in this algorithm are considered valid. They are the only ones that “count” (all others are ignored by correct processes), and they are defined as follows:

- A message received in round  $i$  is *valid* iff it has the form  $m : p_1 : p_2 : \dots : p_i$  where  $m \in \mathcal{M}$ ,  $p_1$  is the sender and all  $p_k$ s are distinct. The first component  $m$  of this valid message will be called the valid message’s *value*.

Such a message should be interpreted as meaning: in round  $i$ ,  $p_i$  said that in round  $i - 1$ ,  $p_{i-1}$  said that ... that in round 1,  $p_1$  (the sender) broadcast  $m$ .

The idea of the algorithm is the following. In round 1 the sender (if correct) signs the message it wishes to broadcast and sends the signed message to all processes. Of course, if the sender is faulty it

---

Process  $p$

Initialisation:

```
if  $p = \text{sender}$  and wishes to broadcast  $m \in \mathcal{M}$  then  
     $\text{extracted} := \text{relay} := \{m\}$   
else  
     $\text{extracted} := \text{relay} := \emptyset$ 
```

In round  $i$ ,  $1 \leq i \leq t + 1$ :

```
for each  $s \in \text{relay}$  do send  $s : p$  to all  
receive round  $i$  messages from all processes  
 $\text{relay} := \emptyset$   
for each valid round  $i$  msg  $s = m : p_1 : \dots : p_i$  received do  
    if  $m \notin \text{extracted}$  then  
         $\text{extracted} := \text{extracted} \cup \{m\}$   
         $\text{relay} := \text{relay} \cup \{s\}$ 
```

At the end of round  $t + 1$ :

```
if  $\exists m \in \mathcal{M}$  s.t.  $\text{extracted} = \{m\}$  then deliver  $m$   
else deliver SF
```

---

Figure 1: A TRB algorithm for arbitrary failures with message authentication

---

can append its signature to different messages and send the signed messages to different destinations. A process that receives a valid round 1 message (from the sender) applies its signature to it and relays it to all processes in round 2. Subsequently, in each round a process “extracts” the value component from each valid message it receives in that round. If this is the first time it extracts that value, it remembers the valid message from which it extracted it. In the next round it applies its signature to that message and relays it to all processes, to “convince” any of them that have not already extracted its value to do so. At the end of round  $t + 1$ , each correct process decides what message to deliver according to the following rule: If it has extracted exactly one value it will deliver that value; if it has extracted no value or more than one value it will deliver **SF**. The correctness of the algorithm hinges on the fact that faulty processes can’t conspire in such a way as to cause one correct process to extract only one value while another extracts no value or several values. The algorithm is shown in Figure 1.

**Theorem 1** *The algorithm in Figure 1 solves TRB with arbitrary failures assuming message authentication.*

**PROOF.** *Termination:* Every correct process delivers a message at the end of round  $t + 1$ .

*Agreement:* We say that a correct process  $p$  **extracts**  $m$  **in round**  $i$  if  $p$  inserts  $m$  into the set  $\text{extracted}$  in round  $i$ . In particular, if the sender is correct and wishes to broadcast  $m$  we say that it extracts  $m$  in “round” 0 (see initialisation step in Figure 1).

**Claim 1.1** *If a correct process extracts  $m \in \mathcal{M}$  then all correct processes will extract  $m$ .*

PROOF OF CLAIM 1.1. Let  $i$  be the earliest round in which some correct process extracts  $m \in \mathcal{M}$ , and let  $p$  be such a process. If  $i = 0$  then  $p$  is the sender and it will send  $m : p$  to all processes in round 1, and all other correct processes will extract  $m$  in that round. Thus all correct processes will extract  $m$ , as wanted.

Thus, we may assume that  $i > 0$ . Process  $p$  extracted  $m$  because it received a valid message  $m : p_1 : \dots : p_i$  in round  $i$ . By the definition of valid message,  $p_1, \dots, p_i$  are all distinct. We claim that they are all faulty. Suppose, for contradiction, that  $p_j$  is correct, for some  $j$  such that  $1 \leq j \leq i$ . Since the signature of a correct process cannot be forged, it follows that  $p_j$  signed and relayed the message  $m : p_1, \dots, p_j$  in round  $j$ . Since  $p_j$  is correct it extracted  $m$  in round  $j - 1 < i$ , contradicting that  $i$  is the earliest round in which a correct process extracted  $m$ . Thus  $p_1, \dots, p_i$  are distinct and faulty; hence  $i \leq t$ .

Therefore,  $p$  will send a valid message  $m : p_1 : \dots : p_i : p$  to all processes in round  $i + 1 \leq t + 1$ . All correct processes will receive that message and will extract  $m$  in round  $i + 1$  if they have not done so already. Thus, all correct processes extract  $m$ , as wanted.  $\square$

From the claim it follows that all correct processes extract the same set of values. Thus they all deliver the same message, proving Agreement.

*Validity:* If the sender is correct and wants to broadcast  $m$ , by definition it extracts  $m$  (and no other value) in ‘round’ 0. It does not extract any other value in any round  $i > 0$ , because to do so it would have to receive a valid message  $m' : p_1 : \dots : p_i$ , where  $p_1 = \text{sender}$ , which contradicts the unforgeability property of authenticated messages. Thus, if the sender is correct, it extracts only the message it broadcast. This implies Validity.  $\square$

Dolev and Strong proposed an improvement of this algorithm: After a (correct) process has extracted and relayed two distinct values it need not relay any other values. The extraction and relay of two distinct values will cause the message delivered to be **SF** — no more are needed to ensure the consistency of the message delivered.

The message complexity of the algorithm incorporating this improvement is  $\Theta(n^2)$ , where we only count messages sent by correct processes.<sup>1</sup> The algorithm takes  $t + 1$  rounds in all runs. Thus, it is not an early-stopping algorithm. There exist early stopping algorithms that work for arbitrary failures with message authentication.

## 1.2 An algorithm without authenticated messages

Srikanth and Toueg were able to identify the essential properties of message authentication that make the algorithm in Figure 1 work and encapsulated them in two primitives:

- **offer**( $p, m, i$ ), executed by process  $p$  in round  $i$ ; and
- **accept**( $p, m, i$ ), executed by process  $q$  in some round  $j \geq i$ .

where  $m$  is a message.

The **offer** and **accept** primitives should be thought of as high-level send and receive operations, respectively. In what follows we’ll say that a process **offers** or **accepts** a triple  $(p, m, i)$  to mean

---

<sup>1</sup>In the arbitrary failure model (even in the presence of message authentication), it is not “fair” to count messages of faulty processes because a faulty process may send arbitrarily many messages.

---

Process  $p$

Initialisation:

```
if  $p = \text{sender}$  and wishes to broadcast  $m \in \mathcal{M}$  then  
     $extracted := relay := \{m\}$   
else  
     $extracted := relay := \emptyset$ 
```

In round  $i$ ,  $1 \leq i \leq t + 1$ :

```
for each  $s \in relay$  do offer( $p, m, i$ )  
accept round  $i$  messages (according to the offer/accept implementation)  
 $relay := \emptyset$   
for each  $m \in \mathcal{M}$  s.t.  $p$  has accepted at least  $i$  triples  
of the form  $(q_k, m, j_k)$  where the  $q_k$ s are distinct and  
the  $j_k$ s are in the range  $1..i$ , including (sender,  $m, 1$ ) do  
    if  $m \notin extracted$  then  
         $extracted := extracted \cup \{m\}$   
         $relay := relay \cup \{m\}$ 
```

At the end of round  $t + 1$ :

```
if  $\exists m \in \mathcal{M}$  s.t.  $extracted = \{m\}$  then deliver  $m$   
else deliver SF
```

---

Figure 2: A TRB algorithm for arbitrary failures without message authentication

---

that it executes the primitive **offer**( $p, m, i$ ) or **accept**( $p, m, i$ ). We define these primitives by stating the properties they are supposed to satisfy. Later we will show how to implement the primitives and prove that the given implementation is correct, i.e., it satisfies these properties. The properties of **offer** and **accept** are:

- *Validity*: If a correct process  $p$  offers ( $p, m, i$ ) in round  $i$ , then all correct processes accept ( $p, m, i$ ) in round  $i$ .
- *Relay*: If a correct process  $q$  accepts ( $p, m, i$ ) in round  $j \geq i$  then all correct processes accept ( $p, m, i$ ) by round  $j + 1$  (whether or not  $p$  is correct.)
- *Unforgeability*: If a correct process  $q$  accepts ( $p, m, i$ ) in round  $j \geq i$  and  $p$  is correct then  $p$  offered ( $p, m, i$ ) in round  $i$ . (In other words, faulty processes can't "fool" a correct process  $q$  into accepting ( $p, m, i$ ) for a correct  $p$ , unless  $p$  actually offered it.)

In Figure 2 we show an algorithm that uses these primitives to solve TRB. Underlying this protocol is an implementation of the offer and accept primitives which, among other things, specifies the conditions under which a process accepts a triple in each round on the basis of messages it has received.

**Theorem 2** *The algorithm in Figure 2 solves TRB (assuming a correct implementation of the offer/accept primitives).*

PROOF. *Termination:* Every correct process decides in round  $t + 1$ .

*Agreement:* As in the proof of Theorem 1, we say that a correct process  $p$  **extracts  $m$  in round  $i$**  if  $p$  inserts  $m$  into the set *extracted* in round  $i$ . In particular, if the sender is correct and wishes to broadcast  $m$  we say that it extracts  $m$  in “round” 0. We prove the following

**Claim 2.1** *If a correct process extracts  $m \in \mathcal{M}$  then every correct process extracts  $m$ .*

PROOF OF CLAIM 2.1. Let  $i$  be the earliest round in which any correct process extracts  $m$  and let  $p$  be such a process. If  $i = 0$  then  $p$  is the sender, and it will offer  $(\text{sender}, m, 1)$  in round 1. By Validity (of offer/accept), all correct processes will accept  $(\text{sender}, m, 1)$  in round 1 and hence will extract  $m$  in that round, as wanted. Thus, we may assume that  $i > 0$  and hence that the sender is faulty. Since  $p$  extracted  $m$  in round  $i$ , then by round  $i$  process  $p$  has accepted (at least)  $i$  triples  $(q_k, m, j_k)$ , where all  $q_k$ s are distinct and the  $j_k$ s are in the range  $1..i$ , and one of these triples is  $(\text{sender}, m, 1)$ . We claim that the processes  $q_1, \dots, q_i$  are all faulty. For, suppose that one of them, say  $q_k$ , was correct. Since  $p$  accepted  $(q_k, m, j_k)$ , by Unforgeability,  $q_k$  offered  $(q_k, m, j_k)$  in round  $j_k \leq i$ . But then  $q_k$  extracted  $m$  in round  $j_k - 1 < i$ , contradicting that the earliest round in which any correct process extracted  $m$  is  $i$ . Since  $q_1, \dots, q_i$  are distinct and faulty,  $i \leq t$ .

By the Relay property, all correct processes will accept these  $i$  triples by round  $i + 1$ . (Since  $i \leq t$ , round  $i + 1$  exists.) Furthermore,  $p$  will offer  $(p, m, i + 1)$  in round  $i + 1$  and, by Validity (of offer/accept), all correct processes will accept  $(p, m, i + 1)$  in round  $i + 1$ . Thus, all correct processes will accept (at least)  $i + 1$  triples  $(q_k, m, j_k)$  by round  $i + 1$ , where all  $q_k$ 's are distinct from each other and one of these triples is  $(\text{sender}, m, 1)$ . Thus all correct processes will extract  $m$  by round  $i + 1$ .  $\square$

By Claim 2.1, all correct processes extract the same set of values. Thus, they all deliver the same message, proving Agreement.

*Validity:* Suppose the sender  $s$  is correct and wishes to broadcast  $m$ . By our convention,  $s$  extracts  $m$  in round 0. Furthermore,  $s$  does not extract any  $m'$ , where  $m' \neq m$ , in any subsequent round. To do so, it would have to accept a triple of the form  $(s, m', 1)$ . But since  $s$  is correct, by Unforgeability, this would mean that  $s$  offered  $(s, m', 1)$  in round 1. This is impossible since  $s$  is correct and the message it wishes to broadcast is  $m$ , not  $m'$ . Since the correct sender extracts  $m$  and only  $m$ , it delivers  $m$ . This implies Validity.  $\square$

As with the algorithm in Figure 1 we can improve the message complexity of this algorithm by noting that there is no need for a process to offer more than two distinct values.

### Implementation of offer/accept

We now present an implementation of **offer** and **accept** that satisfies Validity, Unforgeability, and Relay, even if faulty processes behave arbitrarily, and without making any assumptions about the availability of an unforgeable signature scheme.

The basic idea is as follows: A process wishing to offer a message  $m$  does so through intermediary “witnesses”. That is, it first sends  $m$  to all processes; each correct process that receives  $m$  from the sender acts as a witness and “endorses” the message by forwarding it to everyone. If a process gets enough endorsements from witnesses for a message  $m$ , it accepts that message. For this idea to work, the faulty processes cannot be too many or else they could pretend to be witnesses to messages that

---

Implementation of **offer/accept**( $p, m, i$ )

phase  $2i - 1$ :

$p$  sends  $(init, p, m, i)$  to all

phase  $2i$ :

If  $q$  received  $(init, p, m, i)$  from  $p$  in phase  $2i - 1$  **then**

$q$  sends  $(echo, p, m, i)$  to all processes ( $q$  becomes a witness)

If  $q$  received  $(echo, p, m, i)$  from at least  $n - t$  processes in phase  $2i - 1$  **then**

$q$  accepts  $(p, m, i)$

phase  $j > 2i$ :

If  $q$  received  $(echo, p, m, i)$  from at least  $t + 1$  processes so far **then**

$q$  sends  $(echo, p, m, i)$  to all processes ( $q$  becomes a witness)

If  $q$  received  $(echo, p, m, i)$  from at least  $n - t$  processes so far **then**

$q$  accepts  $(p, m, i)$

Figure 3: Implementation of **offer** and **accept** primitives

---

were never offered and cause the desired properties to be violated. As we'll see we can make this idea work provided  $n > 3t$ . Note that one “high-level” round for offer/accept requires two “low-level” rounds of message sends and receives: One for the process that is offering the message to send it to the witnesses; and another for the witnesses to relay the endorsements of the message. To avoid confusion we will call the low-level rounds “phases”. Thus, round  $i$  will be implemented by phases  $2i - 1$  and  $2i$ .

Roughly speaking, the protocol works like this: To offer  $m$  in round  $i$ ,  $p$  sends the message  $(init, p, m, i)$  to all processes. An endorsement of  $m$  takes the form of the message  $(echo, p, m, i)$ ; when  $q$  sends such a message, it is a **witness for**  $(p, m, i)$ . Process  $q$  becomes a witness for  $(p, m, i)$  if it receives the  $(init, p, m, i)$  from  $p$  directly, or if it receives endorsements for  $(p, m, i)$  (i.e.,  $(echo, p, m, i)$ 's) from at least  $t + 1$  witnesses — and, therefore, from at least one correct witness. A process **accepts**  $(p, m, i)$  when it has received  $n - t$  endorsements. A more precise description is given in Figure 3.

Note that the implementation does not state when a process terminates; it specifies actions to be taken for the offering and acceptance of  $(p, m, i)$  for all phases  $\geq 2i - 1$ . This is not an issue for us because this implementation will be used within the algorithm in Figure 2 which runs for a fixed number of rounds  $(t + 1)$ . Each round requires two phases, so processes can halt after phase  $2(t + 1)$ .

**Theorem 3** *If  $n > 3t$ , the implementation in Figure 3 satisfies Validity, Unforgeability and Relay.*

PROOF. *Validity:* If  $p$  is correct and offers  $(p, m, i)$  in round  $i$  then  $p$  sends  $(init, p, m, i)$  to all processes in phase  $2i - 1$ . Every correct process  $q$  receives  $(init, p, m, i)$  in phase  $2i - 1$  and thus becomes a witness. Each such  $q$  then sends  $(echo, p, m, i)$  to all processes in phase  $2i$ . Since there are at least  $n - t$  correct processes, every correct process receives  $(echo, p, m, i)$ 's from at least  $n - t$  processes (the correct ones) in phase  $2i$  and thus every correct process accepts  $(p, m, i)$  in phase  $2i$ , which is also in round  $i$ .

*Relay:* Suppose a correct process  $q$  accepts  $(p, m, i)$  in round  $j$ , i.e., phase  $k = 2j - 1$  or  $2j$ . Then  $q$  received  $(echo, p, m, i)$ 's from at least  $n - t$  processes by phase  $k$ . Since at least  $n - 2t$  of these are correct, all correct processes received  $(echo, p, m, i)$ 's from at least  $n - 2t$  distinct processes by phase  $k$ . By assumption,  $n > 3t$  and therefore  $n - 2t \geq t + 1$ . Thus, all correct processes become witnesses by phase  $k$  and send  $(echo, p, m, i)$ 's by phase  $k + 1$ . Since there are  $n - t$  correct processes, all correct processes accept  $(p, m, i)$  by phase  $k + 1$ , which is in round  $j$  or  $j + 1$  (depending on whether  $k = 2j - 1$  or  $2j$ ).

*Unforgeability:* Suppose  $p$  and  $q$  are correct processes and  $q$  accepts  $(p, m, i)$  in round  $j$ . Then  $q$  received  $(echo, p, m, i)$ 's from at least  $n - t$  processes by phase  $k$ , where  $k = 2j - 1$  or  $2j$ . Look at the earliest phase  $k'$  such that some correct process has become a witness to  $(p, m, i)$ , and let  $q'$  be such a correct process. There are two cases:

CASE 1.  $k' = 2i - 1$ . Then  $q'$  received  $(init, p, m, i)$  from  $p$ , and since  $p$  is correct by assumption,  $p$  must have offered  $(p, m, i)$  in round  $i$ .

CASE 2.  $k' > 2i - 1$ . Then  $q'$  has become a witness by virtue of receiving  $(echo, p, m, i)$ 's from at least  $t + 1$  processes in earlier phases. Since there are at most  $t$  faulty processes, one of the senders of these messages must be correct, so this process was a witness to  $(p, m, i)$  earlier than  $q'$ , contrary to the choice of  $k'$ . So, this case is impossible.  $\square$

From Theorems 2 and 3, and the observation that each of the  $t + 1$  “high level” rounds of the algorithm in Figure 2 requires two phases (“low level” rounds) of the offer/accept implementation given above, we get the following

**Corollary 4** *If  $n > 3t$ , there is an algorithm that solves TRB in  $2(t + 1)$  rounds and tolerates arbitrary faults (without assuming message authentication).*

## 2 Lower bounds

In this section we establish two lower bounds concerning the number of faulty processes that a Consensus algorithm can tolerate when processes are subject to arbitrary failures (without message authentication). The first result, due to Pease, Shostak and Lamport, relates the number of faulty processes to the total number of processes. It states that no algorithm can solve Consensus if  $n \leq 3t$ ; in other words, no Consensus algorithm can tolerate  $\lceil n/3 \rceil$  or more faulty processes. The second result, due to Dolev, relates the number of faulty processes to the connectivity of the communication network. It states that no algorithm can solve Consensus if  $n \leq 2\kappa$ , where  $\kappa$  is the connectivity of the communication network; in other words, no Consensus algorithm can tolerate  $\lceil \kappa/2 \rceil$  or more faulty processes.

Since Consensus reduces to TRB in the model of computation under consideration, provided a majority of processes are correct — i.e.,  $n > 2t$  — these results hold for TRB as well. In fact, they were originally stated and proved for that problem. The proofs given here follow a methodology developed by Fischer, Lynch and Merritt and are more conveniently couched in terms of Consensus.

## 2.1 The Model

A **system** consists of a communication graph and an algorithm for each node in the graph. The **communication graph** is a digraph where the nodes represent processes and the edges represent communication links. An **algorithm** for a node of the communication graph is a (possibly infinite state) automaton. In each step it takes, the automaton performs the following actions:

- given its state, it sends messages through the outgoing links;
- it receives messages through the incoming links;
- given its state and the messages received, it changes its state.

Each automaton has initial states, each associated with one of the possible values that a process can propose; and final states, each associated with a decision value.

A **subsystem**  $U$  of system  $S$  consists of a subset of nodes of  $S$  with their associated algorithms and all edges of  $S$  with at least one node in  $U$ . (By a slight abuse of notation we will write  $u \in U$  and  $(u, v) \in U$  to indicate that node  $u$  and edge  $(u, v)$  are in subsystem  $U$ .)

Two subsystems  $U, U'$  of systems  $S, S'$  respectively are **isomorphic** iff there is a 1-1 correspondence  $\sim$  between nodes of  $U$  and  $U'$  and a 1-1 correspondence  $\approx$  between edges of  $U$  and  $U'$  such that for each  $u, v \in S$  and each  $u', v' \in S'$ :

- (a) if  $u \sim u', v \sim v'$  and  $(u, v) \in U$  then  $(u', v') \approx (u, v)$ ;
- (b) for each  $w \in S - U$ , if  $u \sim u'$  and  $(u, w) \in U$  (resp.  $(w, u) \in U$ ), then there is some  $w' \in S' - U'$  such that  $(u', w') \approx (u, w)$  (resp.  $(w', u') \approx (w, u)$ );
- (c) for each  $w \in S - U$  and  $w' \in S' - U'$ , if  $(w, u) \approx (w', u')$  or  $(u, w) \approx (u', w')$ , then  $u \sim u'$ ;
- (d) if  $v \sim v', v \in U, v' \in U'$  then  $v$  and  $v'$  have the same algorithm.

A **node behaviour** is a sequence of the form:

$$\sigma_0, (S_1, R_1, \sigma_1), (S_2, R_2, \sigma_2) \dots (S_k, R_k, \sigma_k)$$

where

- $\sigma_0$  is an initial state of the node.
- $S_i$  is the set of messages sent by the node in the  $i$ -th round.
- $R_i$  is the set of messages received by the node in the  $i$ -th round.
- $\sigma_i$  is the state of the node at the end of round  $i$ .

An **edge behaviour** is a sequence  $m_1, \dots, m_k$  where  $m_i$  is the message sent through the edge in round  $i$ . A **system behaviour** consists of a behaviour for each node and edge of the system, all with the same number of rounds and with the property that node behaviours say  $v$  sent  $m$  to  $u$  in round  $i$  iff the edge  $(v, u)$ 's behaviour says that message  $m$  was sent in round  $i$ . This is a formal way of saying that there are no communication failures. If  $B$  is a behaviour of  $S$  and  $U$  is a subsystem of  $S$ ,  $B_U$  denotes the behaviours in  $B$  of the nodes and edges of  $U$ .

A node  $v$  is *correct* in system behaviour  $B$  if  $v$ 's behaviour in  $B$  is

$$\sigma_0, (S_1, R_1, \sigma_1), \dots, (S_k, R_k, \sigma_k)$$

and satisfies, for each round  $1 \leq i \leq k$ , the following:

- (a)  $R_i = \{m : m \text{ is the } i\text{-th component in the behaviour of some } (w, v)\}$
- (b)  $S_i$  consists of exactly the messages the algorithm of  $v$  says should be sent when  $v$  is in state  $\sigma_{i-1}$ .
- (c)  $\sigma_i$  is exactly the state the algorithm of  $v$  says should be entered when the present state is  $\sigma_{i-1}$  and the messages received are  $R_i$ .

In other words, a correct node (a) receives the messages sent to it, (b) sends messages according to its algorithm, and (c) changes states according to its algorithm, in every round. A node is *faulty* in a system behaviour if it is not correct in that behaviour. Note that by this definition, a faulty node can act arbitrarily.

**Theorem 5** (*The Locality Theorem*) *Let*

- $S, S'$  be systems,
- $U, U'$  be isomorphic subsystems of  $S, S'$ .
- $B, B'$  be behaviours of  $S, S'$

such that

- All nodes in  $U, U'$  are correct in  $B, B'$  respectively.
- If  $v \in U, v' \in U'$  and  $v \sim v'$ , then the initial state of  $v$  in  $B =$  initial state of  $v'$  in  $B'$ .
- If  $w \in S - U, w' \in S' - U', v \in U, v' \in U'$  and  $(w, v) \approx (w', v')$  then the behaviour of the edge  $(w, v)$  in  $B$  is the same as the behaviour of  $(w', v')$  in  $B'$ .

Then,  $B_U = B'_{U'}$ .

Informally, this says that if two isomorphic subsystems consist of correct processes, where corresponding processes start in the same initial states and corresponding edges *into* the subsystems carry the same messages, then the two subsystems will exhibit identical behaviour, no matter what the overall systems into which they are embedded might look like. This is to say, the behaviour of a (correct) process depends exclusively on its initial state and the inputs it receives from the outside world.<sup>2</sup>

The Locality Theorem can be proved by a straightforward induction on the length of the system behaviours (number of rounds).

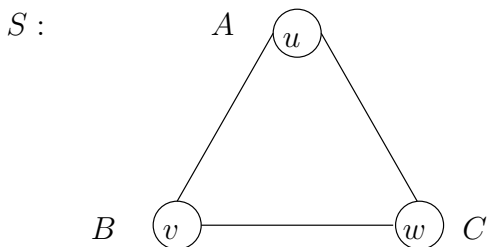
---

<sup>2</sup>We are assuming here that the algorithms of the nodes are defined by means of deterministic automata. This turns out not to be very important in this case: The results we'll prove hold even if nondeterministic algorithms are allowed. The proofs are similar, just slightly more complex. For the sake of simplicity then, we make this assumption.

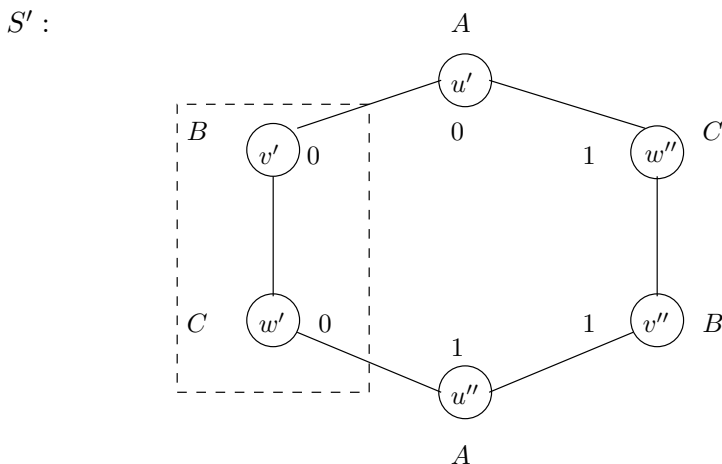
## 2.2 Lower bound on $t$ vs. $n$

**Theorem 6** *The Consensus problem is unsolvable in a three node system if at most one node can be faulty (i.e.,  $n = 3$  and  $t = 1$ ).*

PROOF. Suppose, by way of contradiction, that a system  $S$  that solves the problem in this case exists:



Construct a system  $S'$  as shown below (by “hooking together” two copies of  $S$ ):

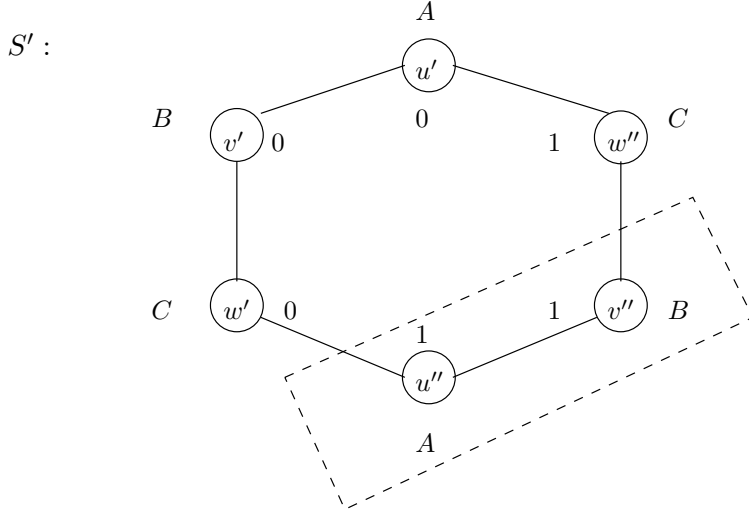


Assign initial values ( $\in \{0, 1\}$ ) to each of the nodes in  $S'$  as shown and let  $B'$  be the behaviour of  $S'$  when the initial values are shown and all nodes are correct. This behaviour is well-defined, because the nodes, when they are correct, follow deterministic algorithms. Note that we do *not* assert that  $S'$  solves the Consensus problem. In particular we do not (and need not), at this point, assume that in  $B'$  all nodes will reach final states that correspond to the same value (i.e. that they reach agreement). However, if we start the system in some state (by selecting initial states for the nodes),  $S'$  will exhibit *some* behaviour. We will now define three behaviours  $B^1$ ,  $B^2$  and  $B^3$  of system  $S$  with one process faulty by considering the behaviour of different subsystems in  $B'$ .

Let  $B^1$  be the behaviour of  $S$  when all nodes have initial value 0, and  $v, w$  are correct. Node  $u$  is faulty and behaves towards  $v$  like  $u'$  does towards  $v'$  in  $B'$  while it behaves towards  $w$  like  $u''$  does towards  $w'$  in  $B'$ . (This is illustrated in the figure above.) Since  $S$  is supposed to solve Consensus when at most one process is faulty, by Validity,  $v, w$  must decide 0 in  $B^1$ . Let  $U$  be the subsystem consisting of  $v$  and  $w$  in  $S$ , and  $U'$  be the subsystem consisting of  $v'$  and  $w'$  in  $S'$ . By the Locality Theorem,  $B_U^1 = B_{U'}^1$ , so  $v', w'$  decide in  $B'$  as they do in  $B^1$ . Therefore,

$$v', w' \text{ decide } 0 \text{ in } B' \tag{1}$$

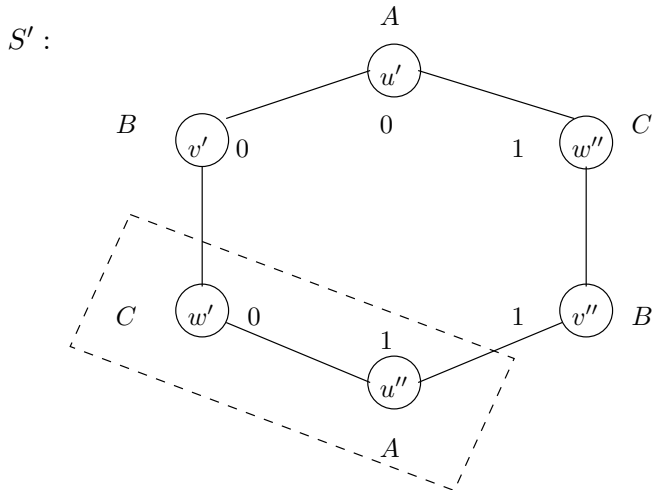
Look at another behaviour,  $B^2$ , of  $S$  defined as follows (see the next figure):  $B^2$  is the behaviour of  $S$  when all nodes have initial values 1 and  $u, v$  are correct. Node  $w$  is faulty and behaves towards  $u$  as  $w'$  behaves towards  $u''$  in  $B'$  and towards  $v$  as  $w''$  behaves towards  $v''$  in  $B'$ .



By Validity,  $u, v$  must decide 1 in  $B^2$ . By the Locality Theorem  $u'', v''$  must decide in  $B'$  as  $u, v$  do in  $B^2$ , so

$$u'', v'' \text{ decide 1 in } B' \tag{2}$$

Now, look at a third behaviour,  $B^3$ , of  $S$  defined as follows (see the next figure):  $B^3$  is the behaviour of  $S$  where  $u, w$  are correct with initial values 1 and 0 respectively while  $v$  is faulty and has any initial value. Node  $v$  behaves towards  $u$  as  $v''$  does towards  $u''$  in  $B'$ , while  $v$  behaves towards  $w$  as  $v'$  does towards  $w'$  in  $B'$ .



By Agreement,  $u$  and  $w$  must decide the same value in  $B^3$ . By the Locality Theorem,  $u'', w'$  must decide in  $B'$  as  $u, w$  do in  $B^3$ . This together with (1) and (2) imply  $0 = 1$  which is a contradiction.  $\square$

**Theorem 7** *If  $t \geq n/3$ , Consensus is unsolvable in any system with  $n$  nodes up to  $t$  of which may be faulty. (Recall that in our model faults are arbitrary.)*

PROOF SKETCH. Suppose, by way of contradiction, that there is a system  $S$  that solves Consensus if  $t \geq n/3$ . Let  $U, V, W$  be a partition of the nodes of  $S$ , where  $|U|, |V|, |W| \leq t$ . (Such a partition exists because  $n \leq 3t$ .) Construct a 3-node system  $S'$  with nodes  $u, v, w$  where  $u$  simulates  $U$ ,  $v$  simulates  $V$ , and  $w$  simulates  $W$ . By a node in  $S'$  (say  $u$ ) “simulating” a set of nodes (say  $U$ ) in  $S$ , we mean the following: The state of the algorithm in  $u$  is a vector with one component for each node in  $U$ . The messages that nodes send in  $S'$  are vectors of messages sent in  $S$  (a message sent from  $u$  to  $v$  has an entry for each pair of processes in  $U \times V$ ). An individual component of the state of  $u$  changes in response to the receipt of a message  $m$  by  $u$  as the corresponding node in  $S$  changes its state in response to the receipt of the messages in the components of  $m$  which contain messages addressed to it (the corresponding node in  $S$ ). Node  $u$  decides a value if one of the nodes in  $U$  (i.e., those simulated by  $u$ ) decides that value.

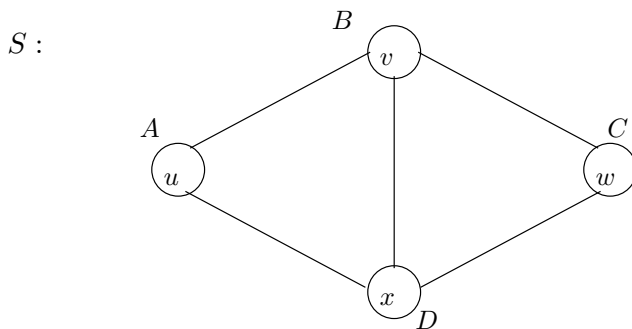
By construction, any behaviour of  $S'$  with at most one faulty node corresponds to a behaviour of  $S$  where at most the nodes in the set simulated by the faulty node are faulty. Since all simulated sets have cardinality at most  $t$ , any behaviour of  $S'$  with at most one faulty node corresponds to a behaviour of  $S$  with at most  $t$  faulty nodes. Since, by assumption,  $S$  solves Consensus with at most  $t$  faulty nodes, it is easy to verify that the simulating system  $S'$  will solve Consensus with at most 1 faulty node, contradicting Theorem 6.  $\square$

Corollary 4 and Theorem 7 imply the following

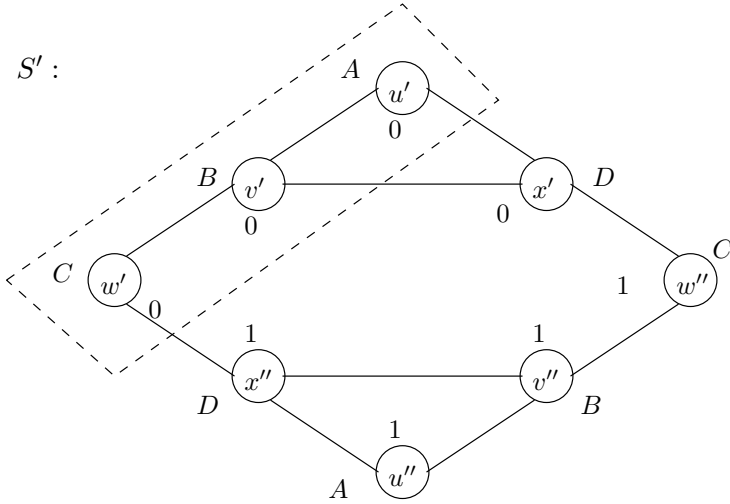
**Corollary 8** *In the arbitrary failure model, the Consensus Problem is solvable iff  $t < n/3$ .*

### 2.3 Lower bound on $t$ vs. connectivity

**Theorem 9** *No system with the communication graph below (which has connectivity 2) can solve Consensus if at most one node is faulty, even if only  $v$  or  $x$  may be faulty.*



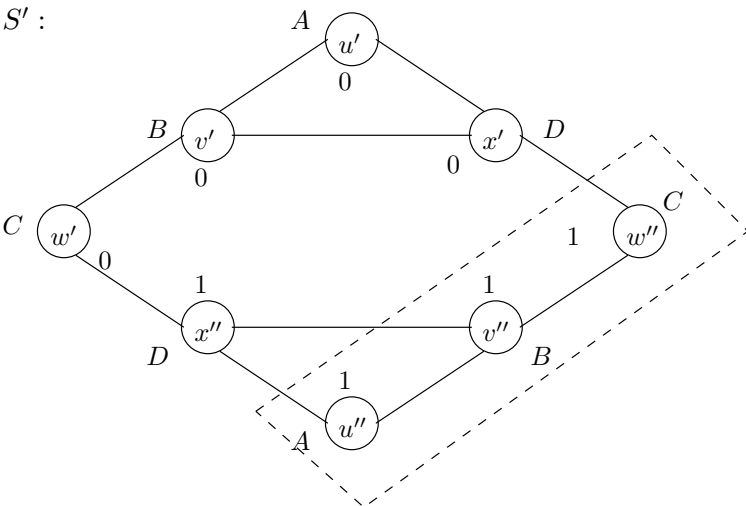
PROOF. Assume a system  $S$  exists which solves the Consensus problem and construct  $S'$  as shown:



Let  $B'$  be the behaviour of  $S'$  when all nodes are correct and have the initial values shown. Let  $B^1$  be the behaviour of  $S$  when all nodes have initial values 0, and  $u, v, w$  are correct while  $x$  is faulty and behaves as follows (see previous figure): It behaves towards  $u$  and  $v$  as  $x'$  does towards  $u'$  and  $v'$  (respectively) in  $B'$ ; and towards  $w$  as  $x''$  does towards  $w'$  in  $B'$ . By Validity and the Locality Theorem,

$$u', v', w' \text{ decide } 0 \text{ in } B'$$

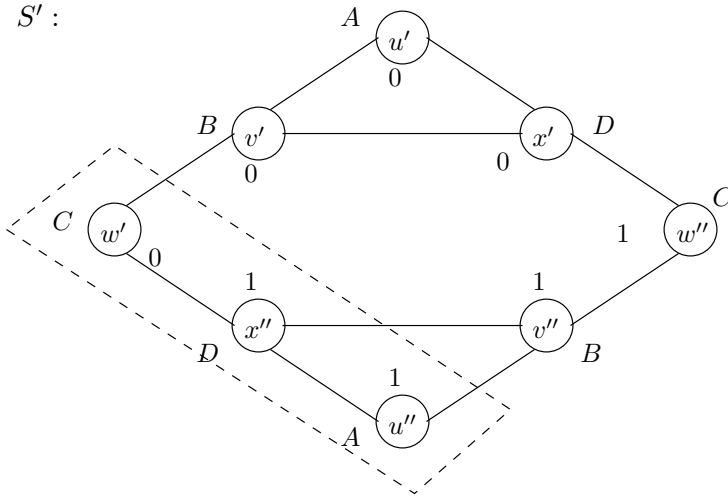
Now consider another behaviour,  $B^2$ , of  $S$  defined as follows:  $B^2$  is the behaviour of  $S$ , where all nodes have initial values 1,  $u, v, w$  are correct, and  $x$  is faulty and behaves as follows (see figure below):  $x$  behaves towards  $w$  as  $x'$  does towards  $w''$  in  $B'$ ; and towards  $u$  and  $v$  as  $x''$  does towards  $u''$  and  $v''$  (respectively) in  $B'$ .



Arguing as before,  $u'', v'', w''$  must all decide 1 in  $B'$ .

Finally, look at a third behaviour,  $B^3$ , of  $S$  defined as follows:  $B^3$  is the behaviour of  $S$ , where now  $u, x, w$  are correct with initial values 1, 1, 0 respectively and  $v$  is faulty, behaving as follows

(see next figure):  $v$  behaves towards  $u$  and  $x$  as  $v''$  does towards  $u''$  and  $x''$  (respectively) in  $B'$ ; and towards  $w$  as  $v'$  does towards  $w'$  in  $B'$ .



By Agreement,  $u, w, x$  must decide the same value in  $\hat{B}$ . By the Locality Theorem then,  $w', x', u''$  must decide the same value in  $B'$ . From this and the conclusions of the previous two behaviours of  $S$ , we get  $0 = 1$ , a contradiction.

Note that this proof required the use of behaviours of  $S$  in which only  $v$  or  $x$  are faulty — so we have honoured the stipulation in the statement of the theorem.  $\square$

**Theorem 10** *Consensus is unsolvable in a system with  $n$  nodes at most  $t$  of which can be faulty (arbitrary faults) if the system has communication graph with connectivity  $\leq 2t$ .*

PROOF SKETCH. By simulation as in the proof of Theorem 2.

Suppose such a system, say  $S$ , exists. Then there is an articulation set of nodes of  $S$  of cardinality  $\leq 2t$ .<sup>3</sup> Partition that articulation set into two subsets  $V$  and  $X$ , each consisting of at most  $t$  nodes. Let  $U$  and  $W$  be a partition of the remaining nodes of  $S$  (excluding  $V$  and  $X$ ), where all paths connecting nodes in  $U$  to nodes in  $W$  go through  $V \cup X$  (such a partition of the remaining nodes exists because  $V \cup X$  is, by choice, an articulation set).

Now construct a system  $S'$  for the communication graph of Theorem 3, where nodes  $u, v, w, x$  simulate, respectively, sets of nodes  $U, V, W, X$  of  $S$  (in the sense described in the proof of Theorem 7). A behaviour of  $S'$  where only one of  $v$  or  $x$  (and no other node) can be faulty corresponds to a behaviour of  $S$  where only the nodes in  $V$  or  $X$ , respectively, are faulty. Each of these sets has cardinality at most  $t$ . Thus, a behaviour of  $S'$  where only one of  $v$  or  $x$  (and no other node) can be faulty corresponds to a behaviour of  $S$  where at most  $t$  nodes are faulty. Since by assumption  $S$  solves Consensus when at most  $t$  nodes are faulty,  $S'$  must solve Consensus when only one of  $v$  or  $x$  (and no other node) can be faulty, contrary to Theorem 9.  $\square$

The lower bound of Theorem 4 is tight, because of the following result, due to Dolev.

<sup>3</sup>An articulation set of nodes is a set of nodes whose removal results in a disconnected or trivial graph.

**Theorem 11** *In a system with  $n$  processes, if  $n > 3t$  and the communication graph has connectivity  $> 2t$  then Consensus is solvable.*

PROOF SKETCH. Take any algorithm  $A$  that works in a completely connected graph (we know such algorithms exist — cf. Theorem 2). Using  $A$  we construct algorithm  $A'$  to solve Consensus in any graph with connectivity  $\geq 2t + 1$  as follows:

- When  $A$  says “ $p$  sends  $m$  to  $q$ ”,  $A'$  says “ $p$  sends  $m$  to  $q$  along  $2t + 1$  node-disjoint paths”. These paths must exist by Menger’s Theorem.<sup>4</sup>

In  $A'$   $q$  receives a message when it gets at least  $t + 1$  copies of that message. It turns out that  $A'$  is correct. □

For omission or crash (rather than arbitrary) faults we have the following tight bound regarding the connectivity of the communication graph: Consensus and TRB can be solved iff the communication graph has connectivity at least  $t + 1$ .

---

<sup>4</sup>Menger’s theorem: If a graph has connectivity  $k$ , then there exist at least  $k$  node-disjoint paths between any two nodes.