Probabilistic Analysis of Disagreement in Synchronous Consensus Protocols
(Preliminary draft)

Negin Fathollahnejad*, Risat Pathan*, Emilia Villani**, Raul Barbosa***, Johan Karlsson*
* Department of Computer Science and Engineering,
Chalmers University of Technology, Gothenburg, Sweden.
** Instituto Tecnologico de Aeronautica,
ITA, Praca Marechal Eduardo Gomes, Sao Jose dos Campos SP
*** Department of Informatics Engineering,
University of Coimbra, Coimbra, Portugal

Abstract

This report presents a probabilistic analysis of a family of simple synchronous round-based consensus algorithms aimed at solving the 1-of-n selection problem. In this problem, a set of \( n \) nodes are to select one common value among a set of \( n \) proposed values. There are two possible outcomes of each node’s selection process: it can decide either to select a value, or to abort. Agreement implies that all nodes select the same value, or all nodes decide to abort. We analyse this problem under the assumption of massive communication failures considering symmetric and asymmetric message losses. Previous research has shown that it is impossible to guarantee agreement among the nodes in a synchronous system subjected to an unbounded number of message losses. Our aim is to find algorithms for which the probability of disagreement is as low as possible. To this end, we study how the probability of disagreement varies for three different decision criteria, the optimistic, pessimistic and the moderately pessimistic. Our results show that the probability of disagreement varies significantly with the number of nodes, the number of rounds, and the probability of message loss. In general, the optimistic decision criterion performs better (has a lower probability of disagreement) than the pessimistic one when the probability of message loss is less than 30% to 70%. On the other hand, the optimistic decision criterion has in general a higher maximum probability of disagreement compared to the pessimistic decision criterion. Moreover we show that the outcome of the moderately pessimistic decision criterion generally lies in between the two other decision criteria.

Keywords
distributed algorithms; consensus; reliability analysis; symmetric and asymmetric communication failures.

I. INTRODUCTION

The problem of reaching consensus on a value among a set of cooperating distributed computing units has been studied extensively over the last thirty years. Despite this, we still lack definite answers to how consensus is best solved in distributed systems that rely on wireless communication.

A main challenge in solving the consensus problem in wireless distributed systems is that the communication channel can be subjected to disturbances of varying duration and magnitude. Consequently, we cannot make any assumptions about the number of messages that can be lost during the execution of a distributed consensus algorithm in such systems. We know from previous research [1], [2] that it is impossible to construct an algorithm that guarantees consensus in the face of unrestricted communication failures.

Design of consensus algorithms that minimize the probability of disagreement in the presence of unrestricted communication failures is an important emerging problem in the area of cooperative systems. Examples of applications that demand fast and reliable real-time consensus include autonomous and semi-autonomous cooperative systems for improving traffic safety and fuel-efficiency of road vehicles, such as vehicle platooning, virtual traffic light[1] and coordinated lane change [3]. Similar demands also emerge in applications for safe and fuel-efficient autonomous manoeuvring of air vehicles.

We are interested in investigating the possibility of using simple deterministic synchronous algorithms for fast consensus in safety-critical cooperative systems. To this end, we investigate in this paper a family of synchronous round-based consensus algorithms to solve the 1-of-n selection problem in the presence of symmetric and asymmetric communication failures.

Since we know that we cannot construct an algorithm that solves this problem perfectly, our aim is to find algorithms for which the probability of disagreement is as low as possible.

In 1-of-n selection, each node (in a system of \( n \) nodes) proposes a value and then all nodes must either select the same value, which has to be one of the proposed values, or decide to abort. Disagreement occurs if some nodes decide to select

---

1In a virtual traffic light, road vehicles approaching an intersection interact via wireless communication to form a virtual, or imaginary, traffic light.
a value, while the remaining nodes decide to abort. (The algorithms we study in this paper is safe in the sense that they guarantee that the nodes will never decide on different values.) As we will show later, the probability of disagreement for a given algorithm depends on a number of parameters: i) the number of nodes in the system, ii) the number of rounds of message exchange, and iii) the probability of message loss. In addition, it also depends on the decision criterion that determines whether a node should decide to abort or to select a value.

The main contributions of this report are as follows. We investigate three practical decision criteria called the optimistic, pessimistic and moderately pessimistic decision criterion. For these, we present closed-form expressions for calculating the probability of disagreement in the presence of symmetric communication failures. For asymmetric communication failures, we use a probabilistic model checking tool to calculate the probability of disagreement. We use this tool and the closed-form expressions to illustrate how the probability of disagreement varies for different system configurations.

The remainder of the paper is organized as follows. Section II describes the system model and our failure assumptions. In addition we present the description of the 1-of-n selection algorithm and the three decision criteria. To provide an intuitive understanding of the impact of the decision criteria, we present in Section III an analysis of a simple system consisting of two processes executing a two-round consensus algorithm. We present closed-form expressions for calculating the probability of disagreement for symmetrical communication failures in Section IV. In Section V we briefly describe PRISM [4] the probabilistic model checking tool which we used to model the 1-of-n selection algorithm. Section VI presents a number of diagrams that illustrates how the probability of disagreement varies for different protocol configurations. We discuss related work in Section VII, and finally in Section VIII we conclude and outline some directions for future research.

II. PROTOCOL DESCRIPTION

A. System Model and Failure Assumptions

Our 1-of-n selection algorithm is based on the classical round-based computational model used by many researchers, such as in [5], [6] and [7]. We consider a synchronous system consisting of a set of \( n \) processes. The processes are indexed respectively with their identifiers as: \( \Pi = \{ p_1, \ldots, p_n \} \). We assume that processes are fully connected to one another via wireless broadcast links and that they execute a deterministic protocol in \( R \) rounds of message exchange. Each round consists of three phases: send, receive and compute.

Failures occur as message losses and can occur on any communication link at any time during the execution of the algorithm. In other words, there are no restrictions on the number, timing or pattern of the lost messages.

We consider two different scenarios for a lost message: (i) when all the intended receivers of the message fail to receive the message (symmetric message loss), and (ii) when only a subset of the intended receivers fail to receive the message (asymmetric message loss).

For simplicity, we assume that the nodes are fault-free. Note, however, that a send omission failure of a node is equivalent to a symmetric message loss, and that a receive omission failure is equivalent to asymmetric message loss where only one node fails to receive a message.

B. The Consensus Algorithm

Alg. 1 shows the pseudocode of the 1-of-n selection algorithm executed by each process \( p_i \in \Pi \). We assume that all processes initially constructs a message, denoted as \( msg_i \), that contains a proposed value \( (\text{proposed}_i) \) and a bit-vector \( (v_i) \) of length \( n \) that represents the view of process \( p_i \). Initially, \( v_i[j] = 0 \) for all \( j \) (i.e., at this point \( p_i \) has not received any message from other processes in the system). We define \( v_i \) as complete if all elements of this vector are set to 1. Similarly, we define \( v_i \) as incomplete if at least one element of \( v_i \) is 0.

**Algorithm 1** Generic algorithm for 1-of-n selection (\( p_i \))

\[
\begin{align*}
msg_i &\leftarrow \{v_i, \text{proposed}_i\}; \\
\text{for } r = 1 \text{ to } R \text{ do do} \begin{align*}
\text{begin}_\text{round} \begin{align*}
&\text{send}\_\text{to}\_\text{all}(msg_i); \\
&\text{receive}\_\text{from}\_\text{all}(); \\
&\text{compute}(msg_i); \\
\end{align*}
\text{end}_\text{round}
\end{align*}
\end{align*}
\]

execute\_decision\_algorithm();

We use this tool and the closed-form expressions to illustrate how the probability of disagreement varies for different system configurations.
After initialization, each process iterates the send, receive and compute phases in each of the \( R \) rounds. These phases work as follows:

**Send:** Process \( p_i \in \Pi \) broadcasts \( msg_i \) to all other processes.

**Receive:** Process \( p_i \) receives messages from the other \( n - 1 \) processes. If \( p_i \) does not receive a message from process \( p_j \) within a bounded time, it assumes that message to be lost.

**Compute:** Each process \( p_i \) performs the computations specified for each decision algorithm and updates its local state including \( msg_i \), if necessary.

After a process finishes the send, receive and compute phases for all \( R \) rounds, it executes the decision algorithm. At the end of the execution of the algorithm each process either decides to select a value or decides to abort.

We consider three different decision criteria for the consensus algorithm given in Alg. 1; namely the optimistic decision criterion, pessimistic decision criterion and the moderately pessimistic decision criterion. Alg. 2 shows the description of the optimistic decision criterion. Executing the optimistic decision criterion, if the view of a process \( p_i \) is complete at the end of the \( R^{th} \) round it selects its \( proposed_i \) as the highest value. Indeed a process with complete view optimistically assumes that all the other processes have also complete views and select a value. A process with incomplete view at the end of the \( R^{th} \) round decides to abort.

Algorithm 2 Optimal decision criterion (\( p_i \))

\[
\text{if } v_i \text{ is complete then}
\begin{align*}
& p_i \text{ selects proposed}_i; \\
\text{else}
& \text{ abort;}
\end{align*}
\]

Alg. 3 shows the compute phase for a process \( p_i \) executing the optimistic criterion. \( p_i \) updates \( proposed_i \) to \( proposed_j \) if \( proposed_j > proposed_i \). Process \( p_i \) also updates its \( v_i \) vector at the end of each round as follows: For all the elements of \( v_j \) that are set to 1, process \( p_i \) sets the corresponding elements in \( v_i \) to 1 also (If it is not already 1).

Algorithm 3 Compute phase: optimistic decision criterion (\( p_i \))

\[
\forall p_j \in \Pi - \{p_i\} \\
\text{if } p_i \text{ received } msg_j \text{ then}
\begin{align*}
& \text{if } proposed_i < proposed_j \text{ then} \\
& \text{ proposed}_i = proposed_j;
\end{align*}
\]

\[
\forall p_k \in \Pi - \{p_i, p_j\} \\
\text{if } (v_j[k] = 1 \text{ and } v_i[k] = 0) \text{ then}
\begin{align*}
& v_i[k] = 1; \\
& v_i[j] = 1
\end{align*}
\]

Alg. 4 shows the description of the pessimistic decision criterion. A process \( p_i \) with incomplete \( v_i \) at the end of the execution of the algorithm decides to abort while a process \( p_i \) with complete \( v_i \) pessimistically assumes that other processes do not have complete views unless they confirmed this at some point during \( R \) rounds of execution. If process \( p_i \) does not receive such confirmations from all processes it decides to abort. We define a vector \( C_i \) of size \( n \) for each process \( p_i \) in order to keep a record of the processes who have sent a confirmation to \( p_i \) (to confirm that their view is complete). Initially, \( C_i[j] = 0 \) for all \( j \). When \( (C_i)_j \) is set to 1, it means that \( p_i \) has received a message from \( p_j \) showing that \( v_j \) is complete. \( C_i \) is complete if all of its elements are set to 1. Alg. 5 shows the compute phase for the pessimistic decision criterion. At the end of all rounds except for the last round, process \( p_i \) updates its proposed value and its view vector in the same way as in the compute phase of the optimistic criterion. In addition at the end of all rounds \( p_i \) updates its \( C_i \), its confirmation vector, by definition.

Algorithm 5 shows the description of the compute phase defined for the moderately pessimistic decision criterion. When a process as \( p_i \) receives a message from a process as \( p_j \) it updates \( proposed_i \) to \( proposed_j \) if \( proposed_j > proposed_i \).
Algorithm 4 Pessimistic decision criterion ($p_i$)

\[
\begin{align*}
\text{if } v_i \text{ is complete then} & \\
& \quad \text{if } C_i \text{ is complete then} \\
& \quad \quad p_i \text{ selects } \text{proposed}_i; \\
& \quad \text{else} \\
& \quad \quad \text{abort}; \\
& \quad \text{end if} \\
& \text{else} \\
& \quad \text{abort}; \\
& \text{end if}
\end{align*}
\]

Algorithm 5 Compute phase: pessimistic decision criterion ($p_i$)

\[
\begin{align*}
\forall p_j \in \Pi - \{p_i\} \\
& \text{if } p_i \text{ received } msg_j \text{ then} \\
& \quad \text{if } r \neq R \text{ then} \\
& \quad \quad \text{if } \text{proposed}_i < \text{proposed}_j \text{ then} \\
& \quad \quad \quad \text{proposed}_i = \text{proposed}_j; \\
& \quad \quad \text{end if} \quad \triangleright \text{update } \text{proposed}_i \\
& \quad \quad \forall p_k \in \Pi - \{p_i, p_j\} \\
& \quad \quad \text{if } (v_j[k] = 1 \text{ and } v_i[k] = 0) \text{ then} \\
& \quad \quad \quad v_i[k] = 1; \\
& \quad \quad \text{end if} \quad \triangleright \text{update } v_i \\
& \quad \quad \text{if } v_j \text{ is complete then} \\
& \quad \quad \quad C_i[j] = 1; \\
& \quad \quad \text{end if} \quad \triangleright \text{update } v_i \\
& \quad \text{end if} \\
& \text{end if}
\end{align*}
\]

Algorithm 6 Compute phase: Moderately pessimistic decision criterion ($p_i$)

\[
\begin{align*}
1: & \forall p_j \in \Pi - \{p_i\} \\
2: & \text{if } p_i \text{ received } msg_j \text{ then} \\
3: & \quad \text{if } r \neq R \text{ then} \\
4: & \quad \quad \text{if } \text{proposed}_i < \text{proposed}_j \text{ then} \\
5: & \quad \quad \quad \text{proposed}_i = \text{proposed}_j; \\
6: & \quad \quad \text{end if} \\
7: & \quad \quad \forall p_k \in \Pi \setminus \{p_i, p_j\} \\
8: & \quad \quad \text{if } (v_j[k] = 1 \text{ and } v_i[k] = 0) \text{ then} \\
9: & \quad \quad \quad v_i[k] = 1; \\
10: & \quad \quad \text{end if} \\
11: & \quad \text{end if} \\
12: & \text{end if}
\end{align*}
\]

Algorithm 7 Moderately pessimistic decision criterion ($p_i$)

\[
\begin{align*}
1: & \text{if } v_i \text{ is complete then} \\
2: & \quad \text{if receives some incomplete view in round } R \text{ then} \\
3: & \quad \quad \text{abort}; \\
4: & \quad \text{else} \\
5: & \quad \quad p_i \text{ selects the highest value}; \\
6: & \quad \text{end if} \\
7: & \text{else} \\
8: & \quad \text{abort}; \\
9: & \text{end if}
\end{align*}
\]
Process $p_i$ also updates its $v_i$ vector at the end of each round as follows: for all the elements of $v_j$ that are set to 1, $p_i$ sets the corresponding elements in $v_i$ to 1 (if it is not already 1). The update of its proposed value and its view vector occurs at the end of all round except for the last round (i.e., round $R$).

Alg. 7 shows the description of the moderately pessimistic decision criterion. A process $p_i$ executing the moderately pessimistic decision criterion decides to abort if its view, $v_i$ is incomplete. Otherwise it checks the second if statement given at line 2 (See Alg. 7). If $p_i$ at round $R$, receives a message from a process $p_j$ indicating that $v_j$ is incomplete, process $p_i$ must abort, otherwise it selects its $\text{proposed}_i$ as the highest value. Process $p_i$ disregards the lost messages in the last round and optimistically assume a complete view for the senders of lost messages.

### III. Analysis of a Simple System

In order to provide an intuitive understanding of how the decision criterion influences the probability of the three possible outcomes of the decision process, i.e., \textit{agreement on a value}, \textit{disagreement}, \textit{agreement on abort}, we will in this section compare the outcomes of the optimistic and the pessimistic decision criteria for a 1-of-2 selection algorithm using two rounds of message exchange (i.e., a $1$-of-$n$ selection algorithm with $n=2$ and $R=2$). We assume that the two processes (called, $p_1$ and $p_2$) respectively propose 11 and 12 as their initial (ranking) values. The objective of the algorithm is to reach agreement on the highest value proposed by any of the two processes, i.e., the value 12 in our example.

In this algorithm, each of the two processes sends two messages, one in the first round and one in second round. Thus, in total four messages are exchanged during the execution of the algorithm. Since we assume that any number of messages can be lost due to communication failures, there are $2^4 = 16$ permutations of lost and successful messages, see Table I.

The columns denoted $T_1$ refer to transmissions from $p_1$ to $p_2$, while columns denoted $T_2$ refers to transmissions from $p_2$ to $p_1$. A successful message transmission is marked as 'OK' while a transmission failure is marked as 'Fail'.

The columns denoted $\text{msg}_1$ and $\text{msg}_2$ state the contents of the messages received by each process in the corresponding round. $\text{msg}_1$ (resp. $\text{msg}_2$) is the message received by $p_2$ (resp. $p_1$) from $p_1$ (resp. $p_2$). As explained before, $\text{msg}_i$ consists of $v_i$, the view vector of process $p_i$, and $\text{proposed}_i$, the value proposed by $p_i$. The content of a message is given within square brackets. The view vector is given with curly brackets. As an example, $\{(0,1), 1\}$ indicates that the view vector is $\{(0,1)\}$ (this shows that the process has not yet received a message from the other process) while the proposed value is 11. '[-]' denotes the loss of a message due to a transmission failure. As a further example consider Case 1 in Table I.

---

Table 1

<table>
<thead>
<tr>
<th>Case</th>
<th>round 1</th>
<th>round 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$T_1$</td>
<td>$T_2$</td>
</tr>
<tr>
<td>1</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>2</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>3</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>4</td>
<td>OK</td>
<td>OK</td>
</tr>
<tr>
<td>5</td>
<td>OK</td>
<td>Fail</td>
</tr>
<tr>
<td>6</td>
<td>OK</td>
<td>Fail</td>
</tr>
<tr>
<td>7</td>
<td>OK</td>
<td>Fail</td>
</tr>
<tr>
<td>8</td>
<td>OK</td>
<td>Fail</td>
</tr>
<tr>
<td>9</td>
<td>Fail</td>
<td>OK</td>
</tr>
<tr>
<td>10</td>
<td>Fail</td>
<td>OK</td>
</tr>
<tr>
<td>11</td>
<td>Fail</td>
<td>OK</td>
</tr>
<tr>
<td>12</td>
<td>Fail</td>
<td>OK</td>
</tr>
<tr>
<td>13</td>
<td>Fail</td>
<td>Fail</td>
</tr>
<tr>
<td>14</td>
<td>Fail</td>
<td>Fail</td>
</tr>
<tr>
<td>15</td>
<td>Fail</td>
<td>Fail</td>
</tr>
<tr>
<td>16</td>
<td>Fail</td>
<td>Fail</td>
</tr>
</tbody>
</table>

Note that the view vector actually consists of two bits. However, one of the bits represents the process’s view of its own value and this bit is always set to 1. For simplicity, we have omitted this bit in the example.
Table II
RESULTS FOR 1-of-2 SELECTION ALGORITHM
AG: AGREEMENT DG: DISAGREEMENT

<table>
<thead>
<tr>
<th>Case</th>
<th>Optimistic decision criterion</th>
<th>Pessimistic decision criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AG(12)</td>
<td>AG(12)</td>
</tr>
<tr>
<td>2</td>
<td>AG(12)</td>
<td>DG</td>
</tr>
<tr>
<td>3</td>
<td>AG(12)</td>
<td>DG</td>
</tr>
<tr>
<td>4</td>
<td>AG(12)</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>5</td>
<td>AG(12)</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>6</td>
<td>DG</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>7</td>
<td>AG(12)</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>8</td>
<td>DG</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>9</td>
<td>AG(12)</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>10</td>
<td>AG(12)</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>11</td>
<td>DG</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>12</td>
<td>DG</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>13</td>
<td>AG(12)</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>14</td>
<td>DG</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>15</td>
<td>DG</td>
<td>AG(abort)</td>
</tr>
<tr>
<td>16</td>
<td>AG(abort)</td>
<td>AG(abort)</td>
</tr>
</tbody>
</table>

Table III
COMPARISON OF DECISION CRITERIA, VIEW OF PROCESS \( p_1 \)

<table>
<thead>
<tr>
<th>Case</th>
<th>(msg_1) sent by ( p_1) to ( p_2) in round 2</th>
<th>(msg_2) received by ( p_1) from ( p_2) in round 2</th>
<th>State of ( msg_1) after compute phase in round 2</th>
<th>Optimistic decision by ( p_1)</th>
<th>Pessimistic decision by ( p_1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>([{1}, 12])</td>
<td>([{1}, 12])</td>
<td>([{1}, 12])</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>B</td>
<td>([{1}, 12])</td>
<td>([{0}, 12])</td>
<td>([{1}, 12])</td>
<td>12</td>
<td>abort</td>
</tr>
<tr>
<td>C</td>
<td>([{1}, 12])</td>
<td>([-]</td>
<td>([{1}, 12])</td>
<td>12</td>
<td>abort</td>
</tr>
<tr>
<td>D</td>
<td>([{0}, 11])</td>
<td>([{1}, 12])</td>
<td>([{1}, 12])</td>
<td>12</td>
<td>abort</td>
</tr>
<tr>
<td>E</td>
<td>([{0}, 11])</td>
<td>([{0}, 12])</td>
<td>([{1}, 12])</td>
<td>12</td>
<td>abort</td>
</tr>
<tr>
<td>F</td>
<td>([{0}, 11])</td>
<td>([-]</td>
<td>([{0}, 11])</td>
<td>abort</td>
<td>abort</td>
</tr>
</tbody>
</table>

in round 2 represents the message received by \( p_2\) and consists of \([\{1\}, 12]\). 12 is the value that \( p_1\) proposes in this round. (\( p_1\) receives 12 from \( p_2\) in round one, and since 12 is greater than its own value 11, \( p_1\) proposes 12 in the second round.) \( \{1\}\) is the view vector of \( p_1\) and is set to 1 because \( p_1\) received a message from \( p_2\) in the first round. Table II shows the outcomes of the decision process for the 16 cases shown in Table I. As we can see in Table II there are 9 cases of agreement on a value, 6 cases of disagreement, one case of agreement on abort for the optimistic decision criterion. For the pessimistic decision criterion there are one case of agreement on a value, two cases of disagreement, and 13 cases of agreement on abort.

We now explain the outcomes shown in Table II. To this end, we refer the reader to Table III and Table IV which describe the six possible variants of information that process \( p_1\) and \( p_2\) can have after the execution of round 2. These cases are denoted A to F for \( p_1\) in Table III and A’ to F’ for \( p_2\) in Table IV.

We will first look at the outcomes of disagreement for the optimistic decision criterion. We start with Case 6 in Table II.
In this section, the detailed analysis of both the optimistic and pessimistic decision criteria for the consensus algorithm is presented in order to efficiently compute the number of possible ways all the processes select the same value (i.e., agreement on a value) or all the processes decide to abort (agreement on abort) or all the processes can neither select the same value nor can decide to abort (disagreement). In particular, we compute $AG$, $AB$, and $DG$ for some given $n$ and $R$, where

- $AG$ is the total number of ways all processes select the same value
- $AB$ is the total number of ways some processes select one value and others select another value
- $DG$ is the total number of ways all processes select different values

In this case, the two messages sent by $p_2$ are both lost, whereas the two messages sent by $p_1$ are received by $p_2$. Since $p_1$ has not received any information from $p_2$, $p_1$ must abort according to Case F in Table III. $p_2$ receives both messages from $p_1$ and decides to select the value 12 according to Case B’ in Table IV. Hence, $p_1$ and $p_2$ decides differently, which leads to disagreement. The same the thing happens in Case 11, where the two messages sent by $p_1$ are both lost, whereas the two messages sent by $p_2$ are received by $p_1$.

In Case 8, $p_1$ decides to abort according to Case F in Table III, while $p_2$ decides on 12 according to Case C’ in Table IV. Case 12 is the same as Case 8, with $p_1$ and $p_2$ swapped. In Case 14, $p_1$ decides to abort according to Case F in Table III, while $p_2$ decides on 12 according to Case E’ in Table IV. In Case 15, $p_2$ decides to abort according to Case C’, while $p_1$ decides on 12 according to Case E. These examples illustrate why the decision criterion is called optimistic. Consider Case 15, where $p_1$ decides on 12 although it knows that $p_2$ did not receive the message it sent in round 1: $p_1$ optimistically assumes that the message it sent in round 2 is successfully received by $p_2$.

We now consider the two cases of disagreement for the pessimistic decision criterion shown in Table II. Using the pessimistic decision criterion a process $p_1$ will not decide on a value unless it has received confirmation that the other process $p_2$ has received the value $p_1$ sent in round 1. (In the general case of 1-of-$n$ selection, this is ensured by the second if statement in Alg. 1, which states that a process must have received complete views for all processes in order to decide on a value.) Thus, it is only for case A in Table III and case A’ in Table IV that the pessimistic decision criterion decides on a value. This implies that there is disagreement if one of the messages sent in the last round is lost and all other messages are successful. Clearly, there are two such cases, namely Case 2 and 3 in Table II.

### IV. Closed-Form Expressions for Symmetric Communication Failures

In this section, we derive closed-form expressions for calculating the probability of disagreement, $P_{DG}$, for the optimistic, pessimistic and moderately pessimistic decision criteria in the presence of symmetric communication failures. The derivation of closed-form expressions for the case of asymmetric communication failures is left for future work.

#### A. Sketch of Analysis and Useful Propositions

Safety-critical system that requires consensus needs to be predictable in the sense that appropriate level of assurance regarding any possible outcome of the consensus algorithm is guaranteed before the system is put in mission. To provide such assurance, the objective in this section is to efficiently count the number of different possibilities (out of total $2^{n-R}$ possibilities) such that all the $n$ processes agree/disagree under the assumption of symmetric communication failure. In addition, given the probability of a message loss, denoted by $q$ (i.e., the probability that a send operation is unsuccessful due to symmetric communication failure), the probability of agreement/abort/disagreement among all the $n$ processes is also computed. Such probabilistic analysis is useful, for example, to verify that the system’s probability of disagreement is not greater than some tolerable limit.

In this section, the detailed analysis of both the optimistic and pessimistic decision criteria for the 1-of-$n$ selection algorithm is presented in order to efficiently compute the number of possible ways all the processes select the same value (agreement on a value) or all the processes decide to abort (agreement on abort) or all the processes can neither select the same value nor can decide to abort (disagreement). In particular, we compute $AG$, $AB$, and $DG$ for some given $n$ and $R$, where

- $AG$ is the total number of ways all processes select the same value
- $AB$ is the total number of ways some processes select one value and others select another value
- $DG$ is the total number of ways all processes select different values

#### Table IV

<table>
<thead>
<tr>
<th>Case</th>
<th>$msg_2$ sent by $p_2$ to $p_1$ in round 2</th>
<th>$msg_1$ received by $p_2$ from $p_1$ in round 2</th>
<th>State of $msg_2$ after compute phase in round 2</th>
<th>Optimistic decision by $p_2$</th>
<th>Pessimistic decision by $p_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A’</td>
<td>{1},12</td>
<td>{1},12</td>
<td>{1},12</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>B’</td>
<td>{1},12</td>
<td>{0},11</td>
<td>{1},12</td>
<td>12</td>
<td>abort</td>
</tr>
<tr>
<td>C’</td>
<td>{1},12</td>
<td>[-]</td>
<td>{1},12</td>
<td>12</td>
<td>abort</td>
</tr>
<tr>
<td>D’</td>
<td>{0},12</td>
<td>{1},12</td>
<td>{1},12</td>
<td>12</td>
<td>abort</td>
</tr>
<tr>
<td>E’</td>
<td>{0},12</td>
<td>{0},11</td>
<td>{1},12</td>
<td>12</td>
<td>abort</td>
</tr>
<tr>
<td>F’</td>
<td>{0},12</td>
<td>[-]</td>
<td>{0},12</td>
<td>abort</td>
<td>abort</td>
</tr>
</tbody>
</table>

and [1] In this case, the two messages sent by $p_2$ are both lost, whereas the two messages sent by $p_1$ are received by $p_2$. Since $p_1$ has not received any information from $p_2$, $p_1$ must abort according to Case F in Table III. $p_2$ receives both messages from $p_1$ and decides to select the value 12 according to Case B’ in Table IV. Hence, $p_1$ and $p_2$ decides differently, which leads to disagreement. The same the thing happens in Case 11, where the two messages sent by $p_1$ are both lost, whereas the two messages sent by $p_2$ are received by $p_1$.

In Case 8, $p_1$ decides to abort according to Case F in Table III, while $p_2$ decides on 12 according to Case C’ in Table IV. Case 12 is the same as Case 8, with $p_1$ and $p_2$ swapped. In Case 14, $p_1$ decides to abort according to Case F in Table III, while $p_2$ decides on 12 according to Case E’ in Table IV. In Case 15, $p_2$ decides to abort according to Case C’, while $p_1$ decides on 12 according to Case E. These examples illustrate why the decision criterion is called optimistic. Consider Case 15, where $p_1$ decides on 12 although it knows that $p_2$ did not receive the message it sent in round 1: $p_1$ optimistically assumes that the message it sent in round 2 is successfully received by $p_2$.

We now consider the two cases of disagreement for the pessimistic decision criterion shown in Table II. Using the pessimistic decision criterion a process $p_1$ will not decide on a value unless it has received confirmation that the other process $p_2$ has received the value $p_1$ sent in round 1. (In the general case of 1-of-$n$ selection, this is ensured by the second if statement in Alg. 1, which states that a process must have received complete views for all processes in order to decide on a value.) Thus, it is only for case A in Table III and case A’ in Table IV that the pessimistic decision criterion decides on a value. This implies that there is disagreement if one of the messages sent in the last round is lost and all other messages are successful. Clearly, there are two such cases, namely Case 2 and 3 in Table II.
• \( AB \) is the total number of ways all processes decide to abort (i.e., agreement on abort)
• \( DG \) is the total number of ways some processes select a value while others decide to abort (i.e., disagreement).

In addition, by considering the probability of a message loss \( (q) \), we also compute

- \( P_{AG} \): the probability of agreement on value;
- \( P_{AB} \): the probability of agreement to abort; and
- \( P_{DG} \): the probability of disagreement

of the system where \( (P_{AG} + P_{AB} + P_{DG}) = 1 \).

According to the 1-of-n selection algorithm given in Algorithm 1, each of the \( n \) processes executes the send operation in each of the \( R \) rounds. Since each send operation can either be successful (message reaches to every other process) or unsuccessful (message reaches to no process), there are \( 2^n R \) possible combinations of all the views of the \( n \) processes at the end of the \( R^{th} \) round. Finding \( AG, AB \) and \( DG \) by considering the exponential number of possible views of all the processes (a trivial but exhaustive approach) can be computationally prohibitive when \( n \cdot R \) is very large (e.g., \( n = 20 \) cars execute the consensus algorithm for \( R = 5 \) rounds in a road intersection). Finding an efficient way to compute \( AG, AB \) and \( DG \) is non-trivial and challenging problem. Our endeavour in Section IV-B and Section IV-C is to address this challenge to efficiently compute \( AG, AB \), and \( DG \) for optimistic and pessimistic criteria, respectively. It will be evident that \( AG, AB \) and \( DG \) can be computed in linear time by conducting elegant analysis of each decision making criterion for the 1-of-n selection algorithm.

The following propositions will be useful in Section IV-B and Section IV-C

**Proposition 1.** Two or more processes fail to send during all the \( 1 \ldots K \) rounds if and only if all the \( n \) processes have incomplete view at the end of \( K^{th} \) round.

**Proof:** (**only if part**) Assume a contradiction that there is at least one process, say process \( p_x \), that has complete view at the end of \( K^{th} \) round. If process \( p_x \) has complete view at the end of \( K^{th} \) round, then each of the processes in set \( \{p_1, \ldots, p_{x-1}, p_{x+1}, \ldots, p_n\} \) successfully sends during one or more of the \( K \) rounds. This contradicts the fact that two or more processes fail to send in all \( K \) rounds since \( \{|p_1, \ldots, p_{x-1}, p_{x+1}, \ldots, p_n|\} = (n-1) \).

**(if part)** Assume a contradiction that zero or one process fails to send in all \( K \) rounds. If zero, i.e., no process fails to send in all \( K \) rounds, then the view of each of the \( n \) processes is complete at the end of \( K^{th} \) round. And, if exactly one process fails to send in all the \( K \) rounds, then each of the remaining \( (n-1) \) processes successfully sends during at least one of the \( K \) rounds. This implies that the only process that fails to send during all the \( K \) rounds has the complete view. Therefore, if zero or one process fails to send during all \( K \) rounds, then at least one process has complete view at the end of \( K^{th} \) round (contradiction!).

**Proposition 2.** Each process successfully sends message during at least one of the \( K \) rounds if and only if each of the \( n \) processes has complete view at the end of \( K^{th} \) round.

**Proof:** (**only if part**) Assume a contradiction that there is at least one process that has incomplete view at the end of \( K^{th} \) round. This can happen only if there is at least one process fails to send in all the \( K \) rounds (contradiction!).

**(if part)** Assume a contradiction that there is at least one process that fails to send during all the \( K \) rounds. This implies that the view of all the processes cannot be complete at the end of \( K^{th} \) round (contradiction!).

**Proposition 3.** Exactly one process fails to send in all \( K \) rounds if and only if one process has complete view and the remaining \( (n-1) \) nodes have incomplete view at the end of \( K^{th} \) round.

**Proof:** It is evident from the proof of Proposition 1 that at most one process can fail to send in all the \( K \) rounds if and only if at least one process has complete view at the end of \( K^{th} \) round. And, according to Proposition 2, no process fails to send in all the \( K \) rounds if and only if all processes have complete view. Combining these two observations, it is not difficult to see that exactly one process fails in all \( K \) rounds if and only if the view of this process is complete while the views of the remaining \( (n-1) \) nodes are incomplete.

Proposition 1–3 will be useful to find \( AG, AB \) and \( DG \) for the optimistic decision criterion of the 1-of-n selection algorithm. Each process select a value or decided to abort based on the different IF conditions of Algorithm 2 and Algorithm 3 respectively for the optimistic and pessimistic criteria. For ease of presentation, we denote C0, C1, C2 the various IF conditions present in Algorithm 2 and Algorithm 3. The semantics for the conditions C0, C1 and C2 being true or false for some process \( p \) are given below:

- C0 is true: The view of the process is complete.
- C0 is false: The view of process is incomplete.

The analysis of the optimistic decision criterion to find \( AG, AB \) and \( DG \) is now presented in Section IV-B
B. Analysis of Optimistic Decision Criterion

The optimistic decision criterion (given in Algorithm 2) for each process is simple: if the view of a process is complete (i.e., C0 is true) at the end of the $R^{th}$ round, then the process selects a value; otherwise, it decides to abort. We determine $\text{AG}$, $\text{AB}$, and $\text{DG}$ for optimistic decision criterion in next three subsections.

1) Finding $P_{\text{AG}}$ for Optimistic Criterion: We have to determine the number of ways the view of each of the $n$ processes can be complete at the end of $R^{th}$ round (i.e., the condition C0 is true for all processes). According to Proposition 2 the views of all the $n$ processes are complete if and only if each process successfully sends at least once during $R$ rounds. Since each send operation can either be successful or unsuccessful, there are $\sum_{i=1}^{R} \binom{R}{i}$ possible combinations for each of which at least one of the $R$ send operations by some process could be successful. Consequently, there are $n$ processes, we have

$$P_{\text{AG}} = \left( \sum_{i=1}^{R} \binom{R}{i} \right)^n = (2^R - 1)^n$$

Note that $i$ send operations of each process are successful while $(R - i)$ send operations of are unsuccessful for each $i$ in Eq. (1). If $q$ is the probability of a message loss, then the probability that all the nodes selects the a value (i.e., agreement on value) is given as follows:

$$P_{\text{AG}} = \left( \sum_{i=1}^{R} \binom{R}{i} \cdot (1 - q)^i \cdot q^{R-i} \right)^n = (1 - q^R)^n$$

2) Finding $P_{\text{AB}}$ for Optimistic Criterion: We have to find the number of ways the view of each of the $n$ processes can be incomplete at the end of $R^{th}$ round (i.e., the condition C0 is false for all processes). According to Proposition 1 the views of all the $n$ processes are incomplete if and only if at least two processes fail to send during all $R$ rounds. If there are $i$ processes that fail to send in all the $R$ rounds, where $2 \leq i \leq n$, then each of the remaining $(n - i)$ processes successfully sends in at least one of the $R$ rounds. Given some $i$, $2 \leq i \leq n$, there are $(\sum_{j=1}^{R} \binom{R}{j})^{n-i} = (2^R - 1)^{n-i}$ possibilities for each of which each of the $(n - i)$ processes successfully sends during at least one of the $R$ rounds for some given $i$. And, the $i$ processes from $n$ processes can be selected in $\binom{n}{i}$ ways, where $2 \leq i \leq n$. Consequently, the number of possibilities all the processes decide to abort is given as follows:

$$\text{AB} = \sum_{i=2}^{n} \binom{n}{i} \cdot (2^R - 1)^{n-i}$$

Given that the probability of a message loss is $q$, the probability that exactly $i$ processes fails to send in all $R$ rounds is $(q^R)^i$ while the probability that each of the $(n - i)$ processes successfully sends during at least one of the $R$ rounds is $\left( \sum_{j=1}^{R} \binom{R}{j} \right)^{n-i} = (1 - q^R)^{n-i}$, where $2 \leq i \leq n$. Consequently, the probability that all the processes agree to abort is given as follows:

$$P_{\text{AB}} = \sum_{i=2}^{n} \binom{n}{i} \cdot (q^R)^i \cdot (1 - q^R)^{n-i}$$

3) Finding $P_{\text{DG}}$ for Optimistic Criterion: Since $\text{AG} + \text{AB} + \text{DG} = 2^n R$, the value of $\text{DG}$ is computed as follows:

$$\text{DG} = 2^n R - \text{AG} - \text{AB}$$

where $\text{AG}$ and $\text{AB}$ are computed in Eq. (1) and Eq. (10), respectively. Similarly, the probability of disagreement is given as follows:

$$P_{\text{DG}} = 1 - P_{\text{AG}} - P_{\text{AB}}$$

where $P_{\text{AG}}$ and $P_{\text{AB}}$ are computed in Eq. (2) and Eq. (11), respectively. This completes the analysis of the optimistic decision criteria. Next sections present the analysis for pessimistic criterion.

C. Analysis of Pessimistic Decision Criterion

In this subsection, the closed-form expressions to compute $P_{\text{DG}}$ and $P_{\text{AG}}$ are presented by analyzing the pessimistic decision criterion (Algorithm 4) where the probability of a message loss is $q$.

1) Finding $P_{\text{DG}}$ for Pessimistic Decision Criterion: Disagreement occurs if some processes decide to abort while others decide on a value. According to the pessimistic decision criterion (given in Algorithm 4), if the view of some process, say process $p_i$, is not complete by round $(R - 1)$, then $p_i$ decides to abort. Notice that if process $p_i$ does not have complete view by round $(R - 1)$, it is also guaranteed that none of the other processes can receive a confirmation from process $p_i$ in any round. Consequently, all other processes (regardless whether their views are complete or not) also decide to abort
and there is no disagreement. The crucial observation is that having complete view by each of the processes at the end of round \((R - 1)\) is a necessary condition for disagreement. There are two possible ways the views of all the processes can be complete at the end of round \(r\), where \(1 \leq r \leq (R - 1)\):

- **Case (i):** all the processes have incomplete views by the end of round \((r - 1)\) and all the processes have complete views in round \(r\), and
- **Case (ii)** exactly \((n - 1)\) processes have incomplete views by the end of round \((r - 1)\) and all the processes have complete views in round \(r\).

Notice that other than these two cases, there is no other case in round \(r\) for which disagreement may occur. Given that the views of all the processes are complete at round \(r\), to compute the probability of disagreement, we have to consider that exactly one process, say process \(p_x\), receives confirmation from all other processes while others do not receive all confirmations. The probability of disagreement for pessimistic decision criterion is computed by analyzing each of the above two cases.

**Analysis of Case (i):** For this case, all \(n\) processes have incomplete views during rounds \((r - 1)\) and all \(n\) processes have complete views in round \(r\), where \(1 \leq r \leq R - 1\). We consider two different subcases for this case:

- **Subcase (i)** all the processors have complete views at round \(r = 1\),
- **Subcase (ii)** all the processors have incomplete views at round \((r - 1)\) and have complete views at round \(r\), where \(2 \leq r \leq (R - 1)\).

Subcase (i): The probability that all the processes have complete views at the end of round 1 is \((1 - q)^n\). Disagreement can occur if during rounds \(2\) \ldots \(R\), there is exactly one process, say \(p_x\), that does not send confirmation in any round while the other \((n - 1)\) processes send confirmation in at least one of the rounds \(2\) \ldots \(R\). The probability of not sending any confirmation by \(p_x\) in any round \(2\) \ldots \(R\) is \(q^{(R-1)}\) and the probability of sending confirmation by each of the processes in \(\Pi - \{p_x\}\) during at least one of the rounds \(2\) \ldots \(R\) is \((1 - q^{(R-1)})^{n-1}\). Since the process \(p_x\) can be selected in \(n\) possible ways, the probability of disagreement when all the processes have complete views at the end of 1st round is given as follows:

\[
(1 - q)^n \cdot q^{(R-1)} \cdot (1 - q^{(R-1)})^{n-1} \cdot n
\]

Subcase (ii): If the views of all the processes are complete at the end of round \((r - 1)\), where \(2 \leq r \leq (R - 1)\), then at least two or more processes have failed to send in all \(1, \ldots, (r - 1)\) rounds. For a given round \(r\), the probability that all the processes have incomplete view at the end of \((r - 1)\) rounds is \(\sum_{i=2}^{n} \binom{n}{i} (1 - q^{r-1})^{n-i} \cdot q^{(r-1)i}\) where \(i\) out of \(n\) processes fail to send in all \((r - 1)\) rounds and \((n - i)\) processes successfully send in at least one of the \((r - 1)\) rounds, where \(2 \leq i \leq n\). Because the view of all the processes are complete at the end of round \(r\) for this case, all these \(i\) processes must successfully send during round \(r\) and this has the probability \((1 - q)^i\). Therefore, for some given \(r\), the probability that all the processes have incomplete views at the end of round \((r - 1)\) and have complete views at the end of round \(r\) is \(\sum_{i=2}^{n} \binom{n}{i} (1 - q^{r-1})^{n-i} \cdot q^{(r-1)i} \cdot (1 - q)^i\).

After the view of all the processes are complete at round \(r\), disagreement occurs if exactly one process, say \(p_x\), does not send confirmation in any of the remaining \((R - r)\) rounds (has probability \(q^{(R-r)}\)) while each of the other \((n - 1)\) processes send confirmation in at least one of the remaining \((R - r)\) rounds (has probability \((1 - q^{R-r})^{n-1}\)). The process \(p_x\) can be selected in \(n\) possible ways. For a given \(r\), the probability of disagreement for this subcase is

\[
\sum_{i=2}^{n} \binom{n}{i} (1 - q^{r-1})^{n-i} \cdot q^{(r-1)i} \cdot (1 - q)^i \cdot n \cdot q^{(R-r)} \cdot (1 - q^{R-r})^{n-1}
\]

Since \(r\) ranges from 2 to \(R - 1\), the probability that disagreement occurs when all the processes have complete view at the end of any round \(2\) \ldots \(R - 1\) is given as follows:

\[3\text{Note that more than one process receive confirmation from all other processes if and only if each process receives confirmation from all other processes (i.e., all processes decide on a value).} \]
Analysis of Case (ii): For this case, exactly \((n - 1)\) processes have incomplete views during rounds \((r - 1)\) and all the processes have complete views in round \(r\), where \(2 \leq r \leq R - 1\). This can happen if exactly one process, say \(p_x\), fails to send in all the \((r - 1)\) rounds and other \((n - 1)\) processes send at least in one of the \((r - 1)\) rounds. Given a particular round \(r, 2 \leq r \leq (R - 1)\), the probability that any one of the \(n\) processes fails to send in all the \((r - 1)\) rounds is \(n \cdot q^{(r-1)}\) and the probability that each of the other \((n - 1)\) processes sends in at least one of the \((r - 1)\) rounds is \((1 - q^{(r-1)})^{n-1}\). The process \(p_x\) must send successfully at round \(r\) (has probability \((1 - q)\)) because the views of all the processes are complete at the end of round \(r\). The probability that the view of each of the processes is complete at the end of round \(r\) is \(n \cdot q^{(r-1)} \cdot (1 - q^{(r-1)})^{n-1} \cdot (1 - q)\). Since the view of process \(p_x\) is complete at the end of round \(r - 1\), the send operation by process \(p_x\) in round \(r\) is also the confirmation of \(p_x\) to all other processes.

After the view of all the processes are complete in round \(r\), disagreement occurs if exactly one process, say \(p_y\), where \(p_x \neq p_y\), does not send confirmation in any of the remaining \((R - r)\) rounds (has probability \(q^{(R-r)}\)) while each of the other \((n - 2)\) processes in \(\Pi - \{p_x, p_y\}\) send confirmation in at least one of the remaining \((R - r)\) rounds (has probability \((1 - q^{R-r})^{n-2}\)). The process \(p_y\) can be selected in \((n - 1)\) possible ways from set \(\Pi - \{p_x\}\) and the probability of disagreement, given that all processes have complete views at round \(r\), is equal to

\[
(1 - q^{r-1})^{n-1} \cdot (1 - q) \cdot (n - 1) \cdot q^{(R-r)} \cdot (1 - q^{R-r})^{n-2}
\]

Since \(r\) ranges from 2 to \(R - 1\), the probability of disagreement, given that all the processes have complete views in any of the \(2 \ldots R - 1\) rounds, is given as follows:

\[
\sum_{r=2}^{R-1} \left( n \cdot q^{r-1} \cdot (1 - q^{r-1})^{n-1} \cdot (1 - q) \cdot (n - 1) \cdot q^{(R-r)} \cdot (1 - q^{R-r})^{n-2} \right)
\]

Combining the probabilities for case (i) and case (ii), the probability of disagreement for the pessimistic decision criterion is computed as follows:
of sending confirmation by all the processes must successfully send during round \((r - 1)\), where \(1 \leq r \leq (R - 1)\):

- **Case (i):** all the processes have incomplete views by the end of round \((r - 1)\) and all the processes have complete views in round \(r\), and
- **Case (ii):** exactly \((n - 1)\) processes have incomplete views by the end of round \((r - 1)\) and all the processes have complete views in round \(r\).

Notice that other than these two cases, there is no other case in round \(r\) for which agreement may occur. Given that the views of all the processes are complete at round \(r\), to compute the probability of agreement, we have to consider that each process received confirmation from all other processes. The probability of agreement \(P_{AG}\) for pessimistic decision criterion is computed by analyzing each of the above two cases.

**Analysis of Case (i):** For this case, all \(n\) processes have incomplete views during rounds \((r - 1)\) and \(n\) processes have complete views in round \(r\), where \(1 \leq r \leq R - 1\). We consider two different subcases for this case:

- **Subcase (i):** all the processes have complete views at round \(r = 1\),
- **Subcase (ii):** all the processes have incomplete views at round \((r - 1)\) and have complete views at round \(r\), where \(2 \leq r \leq (R - 1)\).

Subcase (i): The probability that all the processes have complete views at the end of round \(1\) is \((1 - q)^n\). Agreement can occur if during rounds \(2 \ldots R\), each of the \(n\) processes successfully sends (confirmation) in at least one of the \(2 \ldots R\) rounds. The probability of sending any confirmation by one process in any round \(2 \ldots R\) is \((1 - q^{(R - 1)})\) and the probability of sending confirmation by all the \(n\) processes in \((1 - q^{(R - 1)})^n\). The probability of agreement when all the processes have complete views at the end of the 1st round is given as follows:

\[
(1 - q)^n \cdot (1 - q^{(R - 1)})^n
\]

Subcase (ii): If the views of the processes are incomplete at the end of round \((r - 1)\), where \(2 \leq r \leq (R - 1)\), then at least two or more processes have failed to send in all \(1, \ldots , (r - 1)\) rounds. For a given round \(r\), the probability that all the processes have incomplete view at the end of \((r - 1)\) rounds is \(\sum_{i=2}^{n} \left( {n\choose i} (1 - q^{r - 1})^{n-i} \cdot q^{(r-1)i} \right)\) where \(i\) out of \(n\) processes fail to send in all \((r - 1)\) rounds and \((n - i)\) processes successfully send in at least one of the \((r - 1)\) rounds, where \(2 \leq i \leq n\). Because the view of all the processes are complete at the end of round \(r\) for this case, all these \(i\) processes must successfully send during round \(r\) and this has the probability \((1 - q)^i\). Therefore, for some given \(r\), the probability that all the processes have complete views at the end of round \((r - 1)\) and have complete views at the end of round \(r\) is \(\sum_{i=2}^{n} \left( {n\choose i} (1 - q^{r - 1})^{n-i} \cdot q^{(r-1)i} \cdot (1 - q)^i \right)\).

After the view of all the processes are complete at round \(r\), agreement occurs if during rounds \((r + 1) \ldots R\), each of the \(n\) processes successfully sends (confirmation) in at least one of the \((r + 1) \ldots R\) rounds. The probability of sending any confirmation by one process in any round \((r + 1) \ldots R\) is \((1 - q^{(R - r)})\) and the probability of sending confirmation by all
the \( n \) processes in \((1 - q^{(R-r)})^n\). The probability of agreement when all the processes have complete views at the end of \( r^{th} \) round is given as follows:

\[
\sum_{i=2}^{n} \binom{n}{i} (1 - q^{r-1})^{n-i} \cdot q^{(r-1)i} \cdot (1 - q)^i \cdot (1 - q^{(R-r)})^n
\]

Since \( r \) ranges from 2 to \( R - 1 \), the probability that disagreement occurs when all the processes have complete view at the end of any round \( 2 \ldots R - 1 \) is given as follows:

\[
\sum_{r=2}^{R-1} \sum_{i=2}^{n} \binom{n}{i} (1 - q^{r-1})^{n-i} \cdot q^{(r-1)i} \cdot (1 - q)^i \cdot (1 - q^{(R-r)})^n
\]

Combining the probabilities for subcase (i) and subcase (ii), we have

\[
(1 - q)^n \cdot (1 - q^{R-1})^n + \\
\sum_{r=2}^{R-1} \sum_{i=2}^{n} \binom{n}{i} (1 - q^{r-1})^{n-i} \cdot q^{(r-1)i} \cdot (1 - q)^i \cdot (1 - q^{(R-r)})^n
\]

**Analysis of Case (ii):** For this case, exactly \((n - 1)\) processes have incomplete views during rounds \((r - 1)\) and all the processes have complete views in round \( r \), where \( 2 \leq r \leq R - 1 \). This can happen if exactly one process, say \( p_x \), fails to send in all the \((r - 1)\) rounds and other \((n - 1)\) processes send at least in one of the \((r - 1)\) rounds. Given a particular round \( r \), \( 2 \leq r \leq (R - 1) \), the probability that any one of the \( n \) processes fails to send in all the \((r - 1)\) rounds is \( n \cdot q^{(r-1)} \) and the probability that each of the other \((n - 1)\) processes sends in at least one of the \((r - 1)\) rounds is \((1 - q^{r-1})^{n-1}\).

The process \( p_x \) must send successfully at round \( r \) (has probability \((1 - q)\)) because the views of all the processes are complete at the end of round \( r \). The probability that the view of each of the processes is complete at the end of round \( r \) is \( \left( n \cdot q^{(r-1)} \cdot (1 - q^{r-1})^{n-1} \cdot (1 - q) \right) \). Since the view of process \( p_x \) is complete at the end of round \( r - 1 \), the send operation by process \( p_x \) in round \( r \) is also the confirmation of \( p_x \) to all other processes.

After the view of all the processes are complete in round \( r \), agreement occurs if each of the \((n - 1)\) other processes in \((\Pi - \{p_x\})\) successfully sends in at least one of the remaining \((R - r)\) rounds (has probability \((1 - q^{R-r})^{n-1}\)). The probability of agreement, given that all processes have complete views at round \( r \), is equal to

\[
n \cdot q^{r-1} \cdot (1 - q^{r-1})^{n-1} \cdot (1 - q) \cdot (1 - q^{R-r})^{n-1}
\]

Since \( r \) ranges from 2 to \( R - 1 \), the probability of disagreement, given that all the processes have complete views in any of the \( 2 \ldots R - 1 \) rounds, is given as follows:

\[
\sum_{r=2}^{R-1} \left( n \cdot q^{r-1} \cdot (1 - q^{r-1})^{n-1} \cdot (1 - q) \cdot (1 - q^{R-r})^{n-1} \right)
\]

Combining the probabilities for case (i) and case (ii), the probability of disagreement for the pessimistic decision criterion is computed as follows:
\[ P_{AG} = (1 - q)^n \cdot (1 - q^{R-1})^n + \]
\[
\sum_{r=2}^{R-1} \sum_{i=2}^{n} \binom{n}{i} (1 - q^{r-1})^{n-i} \cdot q^{(i-1)\cdot i} \cdot (1 - q)^i \cdot (1 - q^{(R-r)})^n + \]
\[
\sum_{r=2}^{R-1} n \cdot q^{r-1} \cdot (1 - q^{r-1})^{n-1} \cdot (1 - q) \cdot (1 - q^{R-r})^{n-1} \]

It is not difficult to see that the above equation can be computed in polynomial time. It is not difficult to see that the above equation can be computed in polynomial time. The probability of abort is \( P_{AB} = 1 - P_{DG} - P_{AG} \) where \( P_{DG} \) and \( P_{AG} \) are computed in Eq. (7) and Eq. (8), respectively.

D. Analysis of Moderately Pessimistic Decision Criterion

In this subsection, we present the closed-form expressions to compute \( P_{DG} \) and \( P_{AB} \) by analyzing the moderately pessimistic decision criterion (See Alg. 7) assuming symmetric message losses with the probability of \( q \).

1) Finding \( P_{DG} \) for Moderately Pessimistic Decision Criterion: Same as for other decision criteria, disagreement occurs if some processes select a value while some other decide to abort. We assume there is a set of processes \( \Pi_x \) which decide to select a value, while all other processes as \( p_x \) in \( \Pi - \Pi_x \) decide to abort. There are two conditions for a process as \( p_x \) to decide on selecting a value. First \( p_x \) should have complete view by the end of round \( R - 1 \). Second, process \( p_x \) should not receive any message from any process indicating that their view is incomplete at round \( R \).

We show that the set \( \Pi_x \) consists of exactly one process (we call this process as \( p_x \)). We prove this using contradiction. Our assumptions are as follows:

Assumption 1

There are two processes, \( p_x \) and \( p_x' \), in \( \Pi \) which decide to select a value.

Assumption 2

All processes in \( \Pi - \Pi_x \) decide to abort.

Based on Assumption 1 process \( p_x \) must have complete view at the end of round \( R - 1 \) (See Alg. 7). This means that all messages sent from the \( n - 2 \) processes in \( \Pi - \Pi_x \) and \( p_x \) are successfully delivered in at least one of the \( R - 1 \) rounds. Also from Assumption 1 we know that process \( p_x \) must have complete view at the end of round \( R - 1 \) and this shows that all messages sent from the \( n - 2 \) processes in \( \Pi - \Pi_x \) and \( p_x \) are successfully delivered in at least one of the \( R - 1 \) rounds. Therefore we can conclude that according to Assumption 1, messages sent from all processes in \( \Pi - \Pi_x \) \cup \{p_x\} \cup \{p_x'\} are successfully delivered in at least one of the \( R - 1 \) rounds. Obviously \( \Pi - \Pi_x \) \cup \{p_x\} \cup \{p_x'\} indicates the set of all processes (i.e., \( \Pi \)). In other words from Assumption 1 and Proposition 2 we can conclude that all processes have successfully delivered their messages in at least one of the \( R - 1 \) rounds, which results in complete views for all \( n \) processes by the end of round \( R - 1 \). This contradicts Assumption 2.

So using proof with contradiction we showed that there is exactly one process in \( \Pi_x \), as \( p_x \) which has a complete view at the end of round \( R - 1 \) while all other processes have incomplete views at this point and in order for \( p_x \) to decide to select a value it must not receive any message from other \( n - 1 \) processes in round \( R \). Eq. (9) shows the closed form solution to calculate the probability of disagreement for moderately pessimistic decision criterion.

\[ P_{DG} = n \cdot q^{R-1} \cdot (1 - q^{R-1})^{n-1} \cdot q^{n-1} \]

We explain Eq. (9) as follows. As exactly one process as \( p_x \) has a complete view at round \( R - 1 \), none of the other \( n - 1 \) processes have received a message from \( p_x \) during \( R - 1 \) rounds with the probability of \( q^{R-1} \cdot (1 - q^{R-1})^{n-1} \). On the other hand \( q^{n-1} \) refers to the probability that all messages sent from the \( n - 1 \) processes in \( \Pi - \Pi_x \) are lost in round \( R \). Finally the process \( p_x \) can be selected in \( n \) possible ways from \( n \) processes (See Eq. (9)).

2) Finding \( P_{AB} \) for Moderately Pessimistic Decision Criterion: In order to derive the closed form solution to calculate the probability of abort we consider two cases.

Case (i): All processes have incomplete views by the end of round \( R - 1 \).
Case (ii): Some processes in the set of $\Pi_x$ have complete views by the end of round $R-1$ but in round $R$ they receive incomplete views from all or some of the processes in $\Pi - \Pi_x$. As a result they decide to abort.

First we explain how we calculate the probability for Case (i). We have to find the number of ways the view of each of the $n$ processes can be incomplete at the end of round $R-1$. This case is similar to calculate the probability of abort in Section IV-B2 for the optimistic approach when the total number of rounds are $R-1$. So according to Proposition 1 the views of all the $n$ processes are incomplete if and only if at least two processes fail to send during all $R-1$ rounds. If there are $i$ processes that fail to send in all the $R-1$ rounds, where $2 \leq i \leq n$, then each of the remaining $(n-i)$ processes successfully sends in at least one of the $R-1$ rounds. Given some $i$, $2 \leq i \leq n$, there are $\left(\sum_{j=1}^{R} \binom{R}{i}\right)\cdot (1-q)^{n-i} \cdot (q^{R-1})^{n-i}$ possibilities for each of which each of the $(n-i)$ processes successfully sends during at least one of the $R-1$ rounds for some given $i$. And, the $i$ processes from $n$ processes can be selected in $\binom{n}{i}$ ways, where $2 \leq i \leq n$. Consequently, the number of possibilities that all the processes have incomplete views by round $R-1$ is given as follows:

$$P_{\text{Case}(i)} = \sum_{i=2}^{n} \binom{n}{i} \cdot (q^{R-1})^{n-i} \cdot (1-q)^{n-i}$$

(10)

Given that the probability of a message loss is $q$, the probability that exactly $i$ processes fail to send in all $R-1$ rounds is $\left(\sum_{j=1}^{R} \binom{R}{i}\right)\cdot (1-q)^{i} \cdot (q^{R-1})^{n-i} = (1-q^{R-1})^{n-i}$, where $2 \leq i \leq n$. Consequently, the probability that all the processes given in Case(i) agree to abort is given as follows:

$$P_{\text{Case}(i)} = \sum_{i=2}^{n} \binom{n}{i} \cdot (q^{R-1})^{i} \cdot (1-q^{R-1})^{n-i}$$

(11)

Now we explain how we calculate the probability of Case(ii). First we prove that the set of $\Pi_x$ consists of exactly one process as $p_x$. We show the proof by contradiction. We consider the following assumptions:

**Assumption 1**

Process $p_x$ and $p_x'$ in $\Pi_x$ decide to abort. They both have complete views by the end of round $R-1$ but in round $R$ they receive incomplete views from some or all processes in $\Pi - \Pi_x$.

**Assumption 2**

All processes in $\Pi - \Pi_x$ have incomplete views by round $R-1$ and as a result they all decide to abort.

Based on Assumption 1 process $p_x$ must have complete view at the end of round $R-1$ (See Alg. 7). This means that all messages sent from the $n-2$ processes in $\Pi - \Pi_x$ and $p_x$ are successfully delivered in at least one of the $R-1$ rounds. From Assumption 1 we know that process $p_x'$ has also complete view at the end of round $R-1$ and this shows that all messages sent from the $n-2$ processes in $\Pi - \Pi_x$ and $p_x$ are successfully delivered in at least one of the $R-1$ rounds. From Assumption 1 and Proposition 2 we can conclude that all processes have successfully delivered their messages in at least one of the $R-1$ rounds, which results in complete views for all $n$ processes by the end of round $R-1$. This contradicts Assumption 2.

So $\Pi_x$ consists of exactly one process as $p_x$ which fails to send its message in all $R-1$ rounds. This happens with the probability of $q^{R-1}$. As a result $n-1$ processes have incomplete views by the end of round $R-1$. All other $n-1$ processes successfully deliver their message at least once during $R-1$ rounds of execution, so that the view of $p_x$ is complete by round $R-1$. In round $R$, $p_x$ receives incomplete views from at least one of $n-1$ processes with the probability of $(1-q^{n-1})$. As a result $p_x$ decides to abort.

Eq. (12) shows the probability of case(ii). Then in Eq. (13) the probability of reaching to an agreement on abort is given, which is the sum of the probabilities of two cases, case (i) and (ii).

$$P_{\text{Case(ii)}} = n \cdot (1-q^{R-1})^{n-1} \cdot q^{R-1} \cdot (1-q^{n-1})$$

$$P_{(A_{AB})} = n \cdot (1-q^{R-1})^{n-1} \cdot q^{R-1} \cdot (1-q^{n-1}) + \sum_{i=2}^{n} \binom{n}{i} (1-q^{R-1})^{n-i} \cdot (q^{R-1})^{i}$$

(13)
Eq. [14] shows how we can calculate the probability of agreement on a value, considering that we have the given closed form solutions to calculate the probability of disagreement and abort.

\[ P_{AG} = 1 - P_{DG} - P_{AB} \]

(14)

V. PROBABILISTIC MODEL CHECKING OF THE 1-of-n SELECTION ALGORITHM

We model the given consensus algorithm in Section II-B using a probabilistic model checking tool, PRISM [4]. We assume the given system model and failure assumptions in Section II-A.

Model checking is the problem of automatically checking whether a model of a system satisfies its specifications or not. A model checker receives as an input a state transition model and a specification, and it verifies whether the given model satisfies the specification. In the case of probabilistic model checking, the state transition model contains stochastic behaviour, such as the probabilistic choice among enabled transitions. The probabilistic model checker performs the reachability analysis of the transition system and, in addition, it calculates the likelihoods of reaching the states using numerical methods.

PRISM supports four different classes of models: discrete time Markov chain \((dtmc)\), Markov decision process \((mdp)\), continuous time Markov chain \((ctmc)\), and probabilistic timed automata \((pa)\). Among these models, we do not use \(ctmc\) and \(pa\) since the consensus algorithm does not require the modelling of time intervals. Both \(dtmc\) and \(mdp\) allow the specification of the deterministic and probabilistic transitions. For probabilistic transitions, the choice of the next state is determined by a discrete probability distribution. The difference between \(mdp\) and \(dtmc\) is that \(mdp\) also allows the specification of non-deterministic transitions (not associated with any probability distribution), while \(dtmc\) does not. As the consensus algorithm does not require the use of non-deterministic transition, we define the models of the consensus algorithm as \(dtmc\).

Following we describe in detail the PRISM models which we designed in order to verify the correctness of the 1-of-n selection algorithm under the given failure assumptions in a probabilistic manner. Using PRISM we calculate the probability of reaching to an agreement on a value, the probability of all processes deciding to abort and finally the probability of having disagreement among the processes.

We start with explaining the PRISM model we designed for a system consisting of three processes with the assumption of having symmetric and asymmetric communication failures only. Then in the next sections, we show how the PRISM models can be modified for systems with more processes.

A. PRISM Model For Three Processes With Symmetric Failure Model

1) Model Overview: Our PRISM model is composed of three parts: declarations, modules and expressions. Declarations contain the list of constant values and global variables. The modules describe the behaviour of the processes; Each process is defined as a module. The message exchange among processes is modelled using global variables that are written/read by the modules. Expressions are the expressions that can be used to avoid repetition of code in the module definition. Synchronization among processes is achieved using a global variable (called token) and the decision criteria are embedded in expressions.

Before explaining each part in detail we look at an overview of a process module. Modules in general are divided in two parts: declaration of local variables and description of transitions. Fig. illustrates the module of a process. Table describes the transitions defined for a process. There are four states assumed for a process, \(S0, S1, S2\) and \(S3\). The probabilistic choice occurs between states \(S1\) and \(S2\) and corresponds to the cases where a message is successfully transmitted with the probability of \((1 - q)\) or is lost with the probability of \(q\). \(S3\) refers to the state in which the process is in its last round of execution. \(S0\) is the final state and a process in this state should decide either to select a value or to abort.

![Process model for symmetric failures](image-url)

Figure 1. Process model for symmetric failures
2) **Global Declarations:** **Constants:** The first line of a PRISM model declares its class which in our case is `dtmc`. Then we insert the list of constant values. The given constant values in Listing 2 define the system settings under which the 1-of-n selection algorithm is run: the number of processes (`N`), the number of rounds of execution of the algorithm (`RN`), the probability of losing a message (`q`) and finally the decision criterion (`DC`). Currently, the model supports two different decision criteria: optimistic (`DC=3`) and pessimistic (`DC=1`). Constants N1, N2 and N3 are the processes’ identifiers and are used to define which process has the token and shall fire the next transition (see expression `next` in Section V-B7). Constant `v_max` defines the highest value among the processes’ initial values and is used to limit the range of the variables that stores processes’ values. Constants `v1_ini`, `v2_ini` and `v3_ini` store the initial value for each process. These values can be chosen randomly between 1 and `v_max`. We know that the choice of the initial values does not affect the results of the algorithm. Regardless of the number of failures, if a process decides to select a value it selects the correct value. Therefore we leave the choice of defining the initial values to the user instead of implementing them as a probabilistic choice in the model, which unnecessarily increases the number of states and transitions.
### Declaration of Process Identity

```c
const N1=1; // Identity number of Process 1
const N2=2; // Identity number of Process 2
const N3=3; // Identity number of Process 3
```

**Listing 3. Declaring the processes’ identities**

### Declaration of Process Initial Values

```c
const v_max=2; // Maximum value of a process
const v1_ini=1; // Initial value of Process 1
const v2_ini=2; // Initial value of Process 2
const v3_ini=1; // Initial value of Process 3
```

**Listing 4. Declaring the processes’ initial values**

Finally, constants `not_last` and `last` are used to define which process receives the token after the current process (see expression `next` in Section [V-A4]).

```c
const not_last=1; // Auxiliary constant to define the next process
const last=0; // Auxiliary constant to define the next process
```

**Listing 5. Declaring auxiliary constants**

### Global Declarations

3) Global Declarations: Global Variables: We use global variables to model the message exchange among processes. At each round, each process writes its current value and view in the global variables and reads the values and views of the other processes.

Variable `v_i_ext` contains the value of Process `i` and must be defined within the range of `[0..v_max]`. As the range of acceptable values is between 1 and `v_max`, value 0 is used to indicate the loss of a message sent by a process as Process `i`.

```c
global v1_ext : [0..v_max] init 0; // Message value of Process 1
global v2_ext : [0..v_max] init 0; // Message value of Process 2
global v3_ext : [0..v_max] init 0; // Message value of Process 3
```

**Listing 6. Declaring auxiliary constants**

Variable `w_i_vj_ext` contains the Process `i`’s view of Process `j`. This variable is `boolean` and indicates whether or not Process `i` has received the value of Process `j` during the previous rounds.

```c
global w1_v2_ext : bool init false; // Process 1 view of Process 2
global w1_v3_ext : bool init false; // Process 1 view of Process 3
global w2_v1_ext : bool init false; // Process 2 view of Process 1
global w2_v3_ext : bool init false; // Process 2 view of Process 3
global w3_v1_ext : bool init false; // Process 3 view of Process 1
global w3_v2_ext : bool init false; // Process 3 view of Process 2
```

**Listing 7. Declaring auxiliary constants**

The global variable `token` is used to coordinate the processes and is within the range of `[1..N]`. `token`’s value indicates the identifier of a process that must perform the next transition. When `token=2`, only a transition of Process 2 can be enabled. When Process 2 performs the enabled transition it passes the token to the next process (Process 3) by assigning a new value to it `token=3`, then only a transition of Process 3 can be enabled.

The variable `m_lost` stores the number of lost messages during an execution of the algorithm. This variable is used for verification purposes (e.g. to determine the minimum number of lost messages which results in having disagreement among processes).

```c
global token : [1..N] init 1; // Token used to coordinate the processes
global m_lost : [0..(N*N)] init 0; // Number of lost messages
```

**Listing 8. Declaring auxiliary constants**
4) **Global Declarations: expressions:** The given expressions in the following are defined from the perspective of Process 1, i.e., with the appropriate variables for Process 1. Then, when we define the modules for other processes, we must indicate how the global and local variables are replaced, so that the same expression can be used by other processes. The first expression (next) determines the next value of the token. Table VI shows how the variables defined for the next expression vary for each process. In the case of Process 1, the variables are not replaced, as the expression is defined from the perspective of this process. For Process 2, N1 is replaced by N2 and not_last is not replaced, resulting in next=3. Finally, for Process 3, N1 is replaced by N3 and not_last is replaced by the constant last (defined as 0), which results in next=1.

The expression new computes the new value of a process after a round by comparing the current value (v1, which is an internal variable of the module Process 1) with the values received from other processes (v2_ext and v3_ext). Table VII presents how the variables of this expression are replaced when it is used by Process 2 and Process 3. Additional explanations about how to redefine variables for other processes are given in Section V-B7.

The boolean expressions w1_v2_new and w1_v3_new compute the view that Process 1 has of Process 2 and Process 3, respectively. For the case of w1_v2_new the result is true if at least one of the following conditions is satisfied:

1) It was already true in the previous round (w1_v2 is true, w1_v2 is an internal variable of the module Process 1);
2) Process 1 received the message of Process 2 in the current round (v2_ext!=0); or
3) Process 1 received the message of Process 3 and Process 3 has received the message from Process 2 in a previous round (w3_v2_ext is true).

We observe that, in the case of symmetric failure, the last condition is equal to the first one (when Process 3 receives the message, Process 1 also receives it). However, we leave this condition on the expression in order to be consistent with the given algorithm. Table VIII and IX present how the variables of these expressions are replaced when they are used by Process 2 and Process 3. We observe that in the case of Process 2, the expression w1_v2_new determines by Process 2 and Process 3.
Table VIII

<table>
<thead>
<tr>
<th>Process</th>
<th>w1_v2</th>
<th>v2_ext</th>
<th>w3_v2_ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table IX

<table>
<thead>
<tr>
<th>Process</th>
<th>w1_v3</th>
<th>v3_ext</th>
<th>w2_v3_ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

the view that Process 2 has of Process 3, and in the case of Process 3, it determines the view that Process 3 has of Process 1.

expression w1_c2_new determines whether or not Process 1 has received confirmation from Process 2 indicating that its view is complete. The result is a boolean value that is true if at least one of the following conditions is satisfied:

1) It was already true in the previous round (w1_c2 is true, w1_c2 is an internal variable of the module Process 1);
2) The message received from Process 2 in the current round shows that Process 2 has a complete view (w2_v1_ext and w2_v3_ext are true).

Table X presents how the variables of this expression are replaced when it is used by Process 2 and Process 3.

<table>
<thead>
<tr>
<th>Process</th>
<th>w1_c2</th>
<th>w2_v1_ext</th>
<th>w2_v3_ext</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Process3</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The next expressions verify the decision criteria and provide a boolean outcome: true means to decide on a value and false means decide to abort. For these expressions, the replacement of variables when the expression is called by Process 2 and Process 3 is the same as indicated in the previous tables (see also Section V-B7). The expression decision_OP verifies whether or not Process 1 has a complete view, i.e., has the view of Process 2 (w1_v2 is true) and Process 3 (w1_v3).

// Optimistic Decision

// Optimistic Decision
formula decision_OP = w1_v2 & w1_v3; // Process 1 has complete view at RN

It returns true if the view of Process 1 is complete and the received messages are also complete. It is important to observe that the requirement that the view of Process 1 must be complete at RN-1 is not explicitly embedded in the expression. This requirement is satisfied by not updating the view of Process 1 in the last round (see Section V-A6).

The expression decision_MP verifies whether or not the view of Process 1 is complete (w1_v2 and w1_v3 are true) and Process 1 has received a confirmation that Process 2 and Process 3 also have complete view (w1_c2 and w1_c3 are true). Finally, the expression decision combines the three decision criteria in a single expression using the value of the constant DC.

5) Module Description: Internal Variables: In PRISM, the definition of a module starts with the word module, followed by its name (Process 1). The internal variables of Process 1 are: its current stage in the execution of the algorithm...
(S1), its current round (RN1), its decision (d1), its current value (v1, initiated as v1_ini), its current views of other processes’ values (w1_v2 and w1_v3), and its current confirmations that other processes view are complete (w1_c2 and w1_c3).

The variable S1 assumes its value as described in Table V and Fig. 1 (S1=0 is equivalent to S0, and so on). It is important to observe that S1 is defined in order to help the organization and understanding of the module. The actual state of the module results from the combination of the value of all its variables, not only S1.

6) Module Description: Transitions: The definition of a transition starts with square brackets. They are used to specify the synchronisation of transitions between modules. As the models we describe here have no synchronisation of transitions, all the transitions described here start with empty brackets [].

A transition is composed of a guard and an action separated by an arrow: ([ guard ] → action). The guard is a boolean expression that specifies the condition under which the transition can be executed. The action specifies how internal and global variables are updated when the transition is performed. The update of each variable must be included in parentheses. The parentheses are combined with an & operator. Example: [] guard → (update1) & (update2) & (update3).

In the case of probabilistic transitions, the action is composed of a set of possible actions with the corresponding probabilities, and separated by plus signals. Example: [] guard → p1: action 1 + p2: action 2 + p3: action 3. As for the case of deterministic transitions, each action may be composed of one or more updates. We observe that here we broke the text of a transition in many lines to have a better understanding. In PRISM a transition must be specified in a single line.

The first transition of the module performs the message sending. The guard of this transition specifies that the module must be in the initial state (S1=1), have the token (RN1=1), and must not have complete the last round (RN1<RN). The condition m_lost<(RN+N) is added to avoid compilation errors, it assures that the update of m_lost (m_lost'= m_lost + 1) will not violate the variable range ([0..(RN+N)]). With probability of (1−q) the message is sent successfully: the global variables associated with Process 1’s value (v1_ext) and view (w1_v2_ext and w1_v3_ext) are updated with the current values of the internal variables. With probability of q the message is lost: the global variable v1_ext receives 0, w1_v2_ext and w1_v3_ext are set to false, and the number of lost messages (m_lost) is incremented.

In both cases, the current round of Process 1 is incremented (RN1'=RN1+1), the token is passed to the next process (token'=next), and Process 1 moves to state S1=2.

The following transition describes the computation of the round when Process 1 is not in the last round (RN1<RN). When this transition is enabled, all processes have already sent their messages and the token has returned to Process 1 (token=N1). The round computation consists of updating the internal variables of Process 1 (value:v1; views: w1_v2
and w1_v3; and confirmations: w1_c2 and w1_c3), using the corresponding expressions. Then, the token is passed to the next process and Process 1 returns to S1=1. The following transition describes the computation of the last round

Listing 18. The message sent from Process 1 is either lost or delivered successfully

(RN1=RN). The main difference between this and the previous transition is that the value and views of Process 1 are updated if and only if the decision criterion is the optimistic one (DC=1). The expression \( x' = (c)?a : b \) means that if \( c \) is true \( x = a \), otherwise \( x = b \). Also differently from the previous transition, in this case Process 1 goes to S1=3 and does not pass the token. After computing the last round, Process 1 is at S1=3 and makes a decision using the corresponding expression. It then goes to the final state S1=0 and passes the token to the next process. The module is closed with end_module.

Listing 19. The message sent from Process 1 is either lost or delivered successfully

7) Modules of other processes: Process 2 and 3 are defined as a copy of Process 1. In this case, PRISM imposes that all the internal variables must be renamed. The external variables may be replaced or not by other external variables that have already been defined in the appropriate section.

The basic idea is that, in the definition of Process 1, a reference to Process 2 (which is the next process after Process 1), shall be replaced by a reference to Process 3 for the case of Process 2 (Process 3 is the next process after Process 2). Likewise for the case of Process 3 it should be referenced by Process 1 (which is the next process after Process 3). Similarly, in the definition of Process 1, a reference to Process 3 (second next process after Process 1), shall be replaced, in the case of Process 2, by a reference to Process 1 (second next process after Process 2) and, in the case of Process 3 by a reference to Process 2 (second next process after Process 3).

The general rule adopted in this work for the definition of a new Process \( i \) (1 < \( i \) <= \( N \)) as a copy of Process 1 is:

For each internal variable, global variable or constant used by Process 1 and named \( X_j \) or \( X_j_Y \) or \( X_j_Yw \), where \( X \) and \( Y \) are the variable names and \( j \) and \( w \) are references to other processes in the interval [1..N]:

- If \( (j + i - 1 <= n) \) and/or \( (w + i - 1 <= n) \) then replace it with \( (j + i - 1) \) and/or \( (w + i - 1) \).
- If \( (j + i - 1 > n) \) and/or \( (w + i - 1 > n) \) then replace it with \( (j + i - n) \) and/or \( (w + i - n) \).

Only for the definition of Process \( i \), add the replacement not_last=last. The application of the general rule results in the following definition of Process 2 and 3:

Listing 20. The process makes the decision

Listing 21. The process makes the decision

8) Verification of properties: We specify the verification properties using an extension of probabilistic temporal logic, which combines temporal relationships between events with probabilistic quantifiers. The specification language used by PRISM is named the Probabilistic Computation Tree Logic (PCTL), derived from the well-known Computation Tree Logic (CTL). We used PCTL to calculate the probability that all processes reach the final state in a given condition (agreement, abort or disagreement). For this purpose, the following properties are specified: To better understand the given properties we explain Property (1) given in Listing 22. This property refers to the probability \((P=?)(\) that eventually \((F)\) the system
reaches a state where all the processes are in the final state \((s_1=1) \& (s_2=1) \& (s_3=1) \& (s_4=1)\) and have reached different decisions \((d_1! = d_2) | (d_2! = d_3)\) (disagreement). One limitation of probabilistic model checking is that due to the problem of state explosion, we are only able to calculate probabilities for a network with 3 processes. For a larger number of processes, PRISM is can estimate the value of property using simulation, with a given tolerance and interval of confidence, or with a fixed number of runs.

B. PRISM Model For Three Processes With Asymmetric Failure Model

1) Model overview: In the case of asymmetric failure, each process may receive or lose a message independently from the other processes. As a consequence, the number of states and transitions of the process’s module depends on the number of processes in the network. Generally, the process’s module is composed of \(N+3\) transitions and \(N+3\) states. Fig. illustrates the module of a process for the case of \(N=3\), while Table describes its transitions. For larger number of processes, the probabilistic transition shall be repeated \(N-1\) times.

2) Global Declarations: Constants : The only modification introduced in the constants’ declaration is the replacement of constant \(q\) with constant \(Q\).

```
const double Q=0.5; // Probability of losing a message (0<=q<=1)
```

Listing 26. Probability of losing a message

3) Global Declarations: Global Variables : The only modification introduced in the declaration of global variables is the range of \(m\_lost\), which is enlarged to \([0..RN*N*(N-1)]\).

```
global m_lost : [0..(RN*N*(N-1))] init 0; // Number of lost messages
```

Listing 27. Number of lost messages

4) Global Declarations: expressions: We assume that in the case of having asymmetric failures, all messages are always sent, which means that their values and views are always copied the global variables. In order to register the occurrence of a failure, each Process \(i\) has a set of internal variables, named \(ni\_nfj\), that indicates whether Process \(i\) has received \((ni\_nfj = true)\) or not \((ni\_nfj= true)\) the message of Process \(j\) in the last round. The expressions are redefined in order to consider other processes’ values and views only when they have been received by Process 1. The expression \(v1\_new\) for computing the new value of Process 1 uses the condition operator \(?\) to replace the value of \(v2\_ext\) and \(v3\_ext\) with 0 when the corresponding message has not been received by Process 1.

```
formula v1_new = max(v1,(n1_nf2?v2_ext:0),(n1_nf3?v3_ext:0)); // Process 1 compute new value
```

Listing 28. Process 1 computes new value
The expressions that compute the views of Process 1 (\(w_{1\text{\_v2\_new}}\) and \(w_{1\text{\_v3\_new}}\)) are also modified. Taking as an example \(w_{1\text{\_v2\_new}}\), instead of using \(v_{2\_next}\) to check if the message of Process 2 has been received, \(n_{1\text{-nf2}}\) is used. Also, the view of Process 2 can only be acquired through Process 3 (\(w_{3\text{\_v2\_ext}}\)), when the message from Process 3 has been received (\(n_{1\text{-nf3}}\) is true).

\[
\begin{align*}
\text{formula } w_{1\text{\_v2\_new}} &= w_{1\text{\_v2}} \mid n_{1\text{-nf2}} \mid (n_{1\text{-nf3}} \& w_{3\text{\_v2\_ext}}); // Process 1 update its view of Process 2 \\
\text{formula } w_{1\text{\_v3\_new}} &= w_{1\text{\_v3}} \mid n_{1\text{-nf3}} \mid (n_{1\text{-nf2}} \& w_{2\text{\_v3\_ext}}); // Process 1 update its view of Process 3
\end{align*}
\]

Listing 29. Process 1 updates its view

Similarly, the expressions that compute the confirmations of Process 1 (\(w_{1\text{\_c2\_new}}\) and \(w_{1\text{\_c3\_new}}\)) consider whether or not the messages have been received by checking \(n_{1\text{-nf2}}\) and \(n_{1\text{-nf3}}\). The expressions associated with the decision criteria are not modified as they use only internal variables. The only exception is for the expression \(\text{received\_message\_complete}\), where the condition (\(v_{i\_ext}=0\)) is replaced by \(n_{1\text{-nf1}}\).

5) Module Description: Internal Variables: The range of variable \(s_{1}\) is modified to \([0..N+2]\). Variables \(n_{1\text{-nf2}}\) and \(n_{1\text{-nf3}}\) are added to the list of internal variables.

\[
s_{1} : [0..N+2] \text{ init 1}; // Process 1 current state
\]

Listing 31. Process 1 current state

\[
n_{1\text{-nf3}} : \text{ bool init true}; // Process 1 has not received the message of Process 3
\]

Listing 32. Status of the message from the other processes

6) Module Description: Transitions: The first transition is modified so that Process 1 always sends its message, i.e., copies its value and views to the global variables. A set of \(N-1\) probabilistic transitions are added in order to define, at
### Table XI
**TRANSITIONS FOR THE ASYMMETRIC FAILURE MODEL N=3**

<table>
<thead>
<tr>
<th>Transition</th>
<th>From</th>
<th>To</th>
<th>Probability</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>S1</td>
<td>S2</td>
<td>1</td>
<td>Process 1 sends its message</td>
</tr>
<tr>
<td>T2</td>
<td>S2</td>
<td>S3</td>
<td>1-q</td>
<td>Process succeed in receiving the message of Process 2</td>
</tr>
<tr>
<td>T2</td>
<td>S2</td>
<td>S3</td>
<td>q</td>
<td>Process fails in receiving the message of Process 2</td>
</tr>
<tr>
<td>T3</td>
<td>S3</td>
<td>S4</td>
<td>1-q</td>
<td>Process succeed in receiving the message of Process 3</td>
</tr>
<tr>
<td>T3</td>
<td>S3</td>
<td>S4</td>
<td>q</td>
<td>Process fails in receiving the message of Process 3</td>
</tr>
<tr>
<td>T4</td>
<td>S4</td>
<td>S1</td>
<td>1</td>
<td>Not last round: process receives the messages and computes the round (not last round)</td>
</tr>
<tr>
<td>T5</td>
<td>S4</td>
<td>S5</td>
<td>1</td>
<td>Last round: process receives the messages and computes the round (last round)</td>
</tr>
<tr>
<td>T6</td>
<td>S5</td>
<td>S0</td>
<td>1</td>
<td>Make a decision (agree or abort)</td>
</tr>
</tbody>
</table>

Listing 34. Process 1 either receives or loses the message sent by other processes.

7) **Modules of other Processes:** The variables n1_nfi is set to false and m_lost is incremented. Finally, for the next transitions, the initial and final values of variable s1 are set according to Fig. 2.

8) **Verification of Properties:** No modification is introduced in the list of properties.
C. PRISM Model For More Than Three Processes With Symmetric Failure Model

This section describes how to expand the consensus algorithm model described in Section V-A for the case of 4 processes. The same procedure should be repeated to obtain models with more than 4 processes.

1) Model overview: In the case of symmetric failure no additional transition or state is added to the model of a process.

2) Global Declarations: Constants: The following constants are added or modified: number of processes, identity number of Process 4, initial value of Process 4.

```
3 const N4=4; // Number of processes in the network (N cannot be modified)
4 const RN=2; // Number of rounds in the protocol (RN>=2)
5 const double q; // Probability of losing a message (0<=q<=1)
6 const DC=3; // Decision criterion: OP=1; PS=3;
7 const v_max=2; // Maximum value of a process
8 const v3_ini=1; // Initial value of Process 3
9 const v2_ini=2; // Initial value of Process 2
10 const v1_ini=1; // Initial value of Process 1
11 const DC=3; // Decision criterion: OP=1; PS=3;
12 const RN=2; // Number of rounds in the protocol (RN>=2)
13 const N=4; // Number of processes in the network (N cannot be modified)
```

3) Global Declarations: Global Variables: The following variables must be added to the list of global variables: the value of Process 4, the view each other process has of Process 4, and the view and confirmation of Process 4. The expression for computing the value, views and confirmations of Process 1 are updated to consider the value and views of Process 4. Two new expressions are created for computing Process 1 view and confirmation of Process 4. The decision criteria expressions are also updated to include the view and confirmation of Process 4.
the expression received_message_complete must check if all other processes have the view of Process 4 and if Process 4 has the view of all other processes.

4) Module Description: Internal Variables: Process 1 view and confirmation of Process 4 are added to the list of internal variables.

5) Module Description: Transitions: The first transition is modified so that Process 1 also sends or fails to send its view of Process 4. Furthermore, at the compute phase, Process 1 view and confirmation of Process 4 must be updated.

6) Modules of other Processes: In the definition of Process 2 and 3, the new internal and global variables related to Process 4 are introduced in the list of variables that are renamed or replaced. Moreover, all the renaming and replacing previously defined for the case of N=3 must be revised in order to follow the general rule defined in Section V-B7. One example is the variable w1_v3 in the definition of Process 3: in the case of N=3 it is renamed as w2_v1, while for N=4 it is renamed as w2_v4. Finally, the definition of Process 4 as a copy of Process 1 is added to the model.

7) Verification of properties: The properties are updated to include the final state and the decision of Process 4:

D. PRISM Model For More Than Three Processes With Asymmetric Failure Model

The extension of the asymmetric model for the case of N=4 follows the same steps of the symmetric model. Additionally, a transition is added to Process 1 in order to include the reception of the message from Process 4 (See Appendix A)
VI. COMPARISON OF THE OPTIMISTIC AND THE PESSIMISTIC DECISION CRITERION

In this section, we present several graphs showing how the probability of disagreement, $P_{DG}$, varies with respect to the main system parameters: $n$ the number of nodes involved in the decision process, and $R$ the number of rounds of message exchange.

For symmetric failures, we assume a fixed probability of message loss, $q$, for all messages. For asymmetric failures we assume that all receivers have a fixed probability, $Q$, of not receiving a message. The results for the symmetric failure model are obtained using the closed-form expressions derived in Section IV, while the results for the asymmetric failure model have been calculated by means of the PRISM models described in Section ??.

A. Symmetric versus asymmetric failures

Fig. 3 shows, for symmetric failures, the probability of agreement, disagreement and abort, as a function of $q$, for a 1-of-3 selection algorithm with two rounds of message exchange ($R = 2$). Fig. 4 shows the corresponding results for asymmetric failures, as a function of $Q$. We see that the probability of agreement on a value drops much more rapidly towards zero for the pessimistic decision criterion compared to the optimistic decision criterion as $q$ and $Q$ approach one. Similarly, and as expected, we see that the probability of agreement to abort increase more rapidly for the pessimistic decision criterion (hence the name pessimistic!) as $q$ and $Q$ approach one. The probability of disagreement, $P_{DG}$, shows, as expected, a distinct peak in all the curves. For both failure models, the maximum probability of disagreement is considerably higher for the optimistic criterion than for the pessimistic criterion. On the other hand, the peak occurs for much lower values of $q$ and $Q$ for the pessimistic criterion. These results show that there are pros and cons to both decision criteria. If we compare Fig. 3 and Fig. 4 we see that the asymmetric failure model have higher peaks for $P_{DG}$ compared to the symmetric failure model. It is interesting to see that $P_{DG}$ is as high as 68% for the optimistic criterion for asymmetric failures.

B. Observations for Symmetric Failures

In this section, we investigate variations in the probability of disagreement in the presence of symmetric failures only.

Fig. 5 shows $P_{DG}$ for a 1-of-3 consensus algorithm as a function of $q$. The solid curves show the results for the optimistic decision criterion with $R = 2, 3, 4$ and 6. The dotted curves show the corresponding results for the pessimistic criterion. We see that the peak values of $P_{DG}$ for the optimistic criterion are considerably higher than those for the pessimistic criterion. We also see, as expected from the results shown in Fig. 5, that the $P_{DG}$ for the optimistic criterion peaks at higher values of...
Figure 3. Probability of agreement (AG), abort (AB) and disagreement (DG) as a function of $q$ for 1-of-3 selection algorithm with symmetric failures ($R = 2$), a comparison of optimistic and pessimistic criteria.

Figure 4. Probability of agreement (AG), abort (AB) and disagreement (DG) as a function of $Q$ for 1-of-3 selection algorithm with asymmetric failures ($R = 2$), a comparison of optimistic and pessimistic criteria.

$q$ compare to the pessimistic approach. For both decision criteria, the peaks of the curves move to the right if we increase the number of rounds. This is expected because when the number rounds increases so does the probability that all processes have a complete view at the last round.

Fig. 5 shows the probability of disagreement for a 1-of-$n$ selection algorithm with $R= 3$ rounds. The different curves represent the results for systems with different number of processes ($n = 2, 3, 4$ and $6$). For both decision criteria, with increasing $n$, the peak of the curves move to the left. Hence, if the number of processes increase in a system with a fixed number of rounds, then the peak of $P_{DG}$ moves towards points with lower probabilities of message loss. This implies that

Figure 5. Probability of disagreement ($P_{DG}$) for a 1-of-3 selection algorithm with ($n = 3, R = 2, 3, 4$ and $6$) for symmetric failures. Comparison of the optimistic decision criteria (solid curves) and the pessimistic decision criteria (dotted curves).
the probability of agreement to abort increases when the number of processes increases.

1) An analysis and comparison of moderately pessimistic decision criterion: Fig. 9 shows the probabilities of the three outcomes of the given decision criteria for a 1-of-3 selection algorithm with 2 rounds of execution as a function of the probability of message loss \( Q \). As we see in Fig. 9 for any value of \( Q \), the optimistic decision criterion has the highest probability of agreement while the pessimistic one has the lowest. On the other hand, the pessimistic decision criterion has the highest probability of abort compare to the other two decision criteria.
For all decision criteria the probability of disagreement shows a distinct peak. The maximum probability of disagreement varies significantly for different decision criteria. The optimistic decision criterion has the highest maximum probability of disagreement which is which is on the most right side of the x-axis (a large value of $Q$). The peak of disagreement for the pessimistic decision criterion on the other hand is on the most left side of x-axis (small values of $Q$) and is the smallest maximum probability of disagreement of all decision criteria, although not much larger than the maximum probability of disagreement for the moderately pessimistic decision criterion. Fig. 10 illustrates how the probability of disagreement is affected by varying the number of processes $n$ and rounds $R$. In this figure the probability of disagreement for three different system configurations ($n=3, 4, 6$) each running the 1-of-$n$ selection algorithm in two, three and four rounds of execution. The three given sub graphs 10(a), 10(b) and 10(c) correspond to the three decision criteria, optimistic, pessimistic and moderately pessimistic respectively.

Considering a fixed number of processes, a larger number of rounds means that a process has more chance to complete its view, therefore the probability of agreement increases for all the decision criteria. Consequently the peak of the probability of disagreement moves to the right side of the x-axis and is achieved for larger values of $Q$. For the optimistic decision criterion, the maximum value of disagreement decreases slightly with increasing $R$, but for the pessimistic and moderately pessimistic decision criteria, it increases significantly. An immediate conclusion is that, for all decision criteria, increasing the number of rounds does not guarantee lower probabilities of disagreement if the probability of message loss $Q$ cannot be limited. Varying $n$ affects the probability of disagreement in a different way. In the case of the optimistic decision criterion (Fig. 10(a)), when we increase the number of processes, the maximum probability of disagreement increases significantly (For $n=6$ the probability of disagreement becomes larger than 80%). However, with increasing $n$ the peak of disagreement does not move to the left or right side of the x-axis unlike the results we get with increasing $R$.

In the case of the pessimistic and moderately pessimistic decision criteria (Fig. 10(b) and Fig. 10(c)), we distinguish two different behaviours according to the number of rounds. For $R=2$, in order to have an outcome of agreement or disagreement, all the messages should be successfully transmitted in the first round. When we increase the number of processes, the agreement region is reduced because it is less likely that a process completes its view in the first round. As a consequence, with increasing $n$ the curve of disagreement moves to the left w.r.t. the x-axis when we have $R=2$. In the case of the pessimistic decision criterion, when we increase $n$, the maximum probability of disagreement remains around the same value, that is 0.25. However, for the moderately pessimistic decision criterion, it declines significantly (not larger than 10% for $n=6$). For $R>2$, with increasing $n$ the maximum probability of disagreement increases significantly for both the pessimistic and moderately pessimistic decision criteria. Fig. 11 shows a comparison of the three decision criteria for system configurations of three, four and six processes executing the 1-of-$n$ selection algorithm in $R=2$ rounds. For any number of processes, the optimistic decision criterion have the highest peak of the probability of disagreement. For a given application with fixed $n$, if $Q$ is unknown, disagreement can be minimized by adopting the moderately pessimistic decision criterion with $R=2$. If the range of $Q$ can be estimated, then we have a threshold and for lower values of $Q$ the minimum disagreement is achieved by the optimistic decision criterion, while for higher values of $Q$ the pessimistic decision criterion has the lowest probabilities of disagreement. For example the case of $n=3$, this threshold is around $Q=0.24$, while for $n=6$ it is around $Q=0.15$. Fig. 12 illustrates a comparison of probability of disagreement ($P_{DG}$) for three decision criteria, for a system executing the 1-of-$n$ selection algorithm with $n=3, 4$ and $R=2, 3$ assuming symmetric failures. As we see in this figure, in most cases the maximum value of $P_{DG}$ is significantly lower when compared with the corresponding results for asymmetric failure. For the moderately pessimistic decision criterion, when we increase the number of processes with $R=2$, the probability of disagreement is reduced. Furthermore, for all decision criteria, when we increase the number of rounds, the peak of $P_{DG}$ moves to the right w.r.t. the x-axis but does not decrease, similar to the case of asymmetric failure.
Figure 10. Probability of disagreement for 1-of-\(n\) selection algorithm for \((n = 3, 4, 6)\) with \(R = 2, 3, 4\) under asymmetric failures.

C. Observations for Asymmetric Failures

In this section, we study how the probability of disagreement varies for different values of \(Q\), \(n\) and \(R\) in the presence of asymmetric failures. Fig. 7 shows \(P_{DG}\) for a 1-of-3 consensus algorithm as a function of \(Q\), the probability of message loss in the asymmetric failure model.

As for the symmetric failures, we see that the peak values of \(P_{DG}\) for the optimistic criterion are considerably higher than those for the pessimistic criterion, while for the pessimistic criterion the \(P_{DG}\) peaks occur at lower values of \(Q\) compared to the \(P_{DG}\) peaks for the optimistic criterion. Interestingly, we see that the peak values increase as the number of rounds increase for pessimistic criterion, while trend for the optimistic criterion is the reverse.

Comparing the results in Fig. 7 with those in Fig. 5 as expected, we see higher probabilities of disagreement for the asymmetric failure model than for the symmetric failure model. Similar to the results for the symmetric failure model, the
peak of the curves move to the right for both decision criteria when the number of rounds increases.

Fig. 8 shows $P_{DG}$ as a function of $Q$ with $R = 3$ and $n = 3, 4, 6$ and 8. We see that with increasing $n$, we obtain higher peak values of $P_{DG}$ for both decision criteria. The peak of the curves move to the left with increasing $n$ for the pessimistic criterion, while they move to the right with increasing $n$ for the optimistic criterion. We also see that the peak values for the pessimistic criterion is higher than for the optimistic criterion, which is the opposite to what we observed for the symmetric failures in Fig. 5. Hence, for systems with many processes, the pessimistic criterion is worse than the optimistic criterion for asymmetric failures.

VII. RELATED WORK

The problem of reaching agreement among the processes of a fault-tolerant distributed system has been investigated widely since thirty years ago [8]–[10]. The consensus problem has been proved to be solvable under different failure assumptions such as in [8], [9], [11]. Most of previous research were based on different classes of process failures only, with assuming reliable communication links among processes. Some were simply associating communication failures to process failures rather than investigating them explicitly as an independent phenomenon (e.g., see [12], [13]) which may lead to incorrect characterization of systems. There are also perception-based hybrid failure models proposed in literature such as in [2] in which the sender-caused link faults are considered as process faults and the term link fault is used to denote the receiver-caused failures. Such failure models also may lead to undesirable conclusions for a system. Perfect communications abstraction with retransmission schemes is used in data-link layer protocols [14], but due to the high execution time needed in these solutions they are only useful for asynchronous systems when real-time properties are not so important.

On the other hand, considering highly unpredictable wireless environments, it is important to consider the communication failures explicitly in order to assure dependability and safety of critical distributed systems. Therefore our focus is to study the synchronous consensus problem for systems subject to transient and dynamic communication failures. In such a model, failures may occur on any communication link at any time when there are no limitations on the number or pattern of the lost messages. We define our failure model based on the model introduced by Santoro and Widmayer in [1] denoted as the transmission fault model. We are mainly considering dynamic and transient omission faults. We know from the results given in [1] and [15] that any non-trivial form of agreement is impossible to solve if $n1$ or more messages can be lost per communication round in a system with $n$ processes. The impossibility result given by Santoro and Widmayer is indeed a

---

For example attributing the transmission faults to the sending or receiving processes one may reach to incorrect conclusions such as assuming the entire processes to be faulty due to only a single failure of a message broadcast.
generalization of the given results by Akkoyulu et al. in [16] and later in 1978 by Gray [17] in which they show that there is no deterministic solution to the consensus problem between two processes with unreliable communication links.

Santoro and Widmayer in [15] define a faulty transmission resulting in omission, corruption or addition of a message and then provide an extensive map of possible and impossible computations in the presence of transmission faults. Later in [18] they define bounds on the number of dynamic faults with expressing the connectivity requirements to achieve any non-trivial agreement.

In [14], Afek et al. employed randomization techniques to solve the k-consensus problem in presence of communication failures. In a k-consensus problem, at least \( k \) processes among \( n \) processes decide on the same value such that \( k > n/2 \). They show that the safety properties of consensus (i.e., validity and agreement) are ensured in presence of even unrestricted communication failures, however in order to satisfy the liveness property (i.e., termination) of the consensus algorithm the number of faults in a round should be restricted.

There are a large number of methods suggested in literature to circumvent the given impossibility result in synchronous consensus systems with dynamic omission faults such as [18]–[20]. However, most of the suggested methods take a preventive approach toward this problem, such as restricting the communication failure patterns or limiting the number of failures in a round.

Nevertheless, it is possible to design protocols that have a low probability of failing to reach consensus, so as to meet specific requirements on reliability and availability. This intuition has been explored to build protocols that maximize the probability of correctness by accumulating more information over a larger duration of the execution [21]. Researchers have also focused on stochastic models of verifying the probability of transition into an incorrect state [22].

Our goal in this paper is to design decision making algorithms to run on top of a simple consensus protocol with the main purpose of minimizing the probability of failing to reach consensus. We evaluate and compare the effectiveness of different decision algorithms by means of using probabilistic model checking tools as well as deriving closed form expressions to calculate the probability of disagreement among processes. Our results may be applied also for on-line verification and adaptation to cope with variable probabilities of communication failures. Our work focuses on probabilistic analysis of round-based consensus protocols in which processes communicate in rounds of message exchange in order to decide on a consistent output [9]. Our system model is inspired by a general computational model named the heard-of model introduced by Schiper and Charron-Bost [23] and is used to specify systems with any type of benign failures.

VIII. Conclusion And Future Work

We have presented closed-form expressions for calculating the probability of disagreement in the presence of symmetric message losses for a family of simple synchronous consensus algorithms. Our work is motivated by the need to develop fast and reliable consensus algorithm for distributed cooperative systems for the transportation sector. Since it is impossible to construct an algorithm that solves the consensus problem for a system that uses wireless, and thereby, unreliable communication, we are interested in exploring the design of adaptive consensus algorithms that are equipped with an on-line mechanism which can temporarily shut down the algorithm in situations where the likelihood for disagreement becomes unacceptably high. We are therefore interested in finding computational effective ways of calculating the probability of disagreement on-line. The next step in our pursuit for such algorithms will be to find closed-form expressions for calculating the probability of disagreement for simple algorithms under the assumption of asymmetric message losses.

REFERENCES


APPENDIX
module Process_1;

const N=3; // Number of processes in the network (N cannot be modified)
const RN=2; // Number of rounds in the protocol (RN>=2)
const double q; // Probability of losing a message (0<=q<=1)
const DC=3; // Decision criterion: OP=1; PS=3;

const ID=1; // Identity number of Process 1
const ID=2; // Identity number of Process 2
const ID=3; // Identity number of Process 3

const V_max; // Maximum value of a process
const s1=1; // Initial value of Process 1
const s1=2; // Initial value of Process 2
const s1=3; // Initial value of Process 3

const not_last=1; // Auxiliary constant to define the next process
const last=0; // Auxiliary constant to define the current process

global v1; // Process 1 value
global v2; // Process 2 value
global v3; // Process 3 value

global v1_ext; // Process 1 view of other processes
global v2_ext; // Process 2 view of other processes
global v3_ext; // Process 3 view of other processes

global RN1; // Identity number of Process 1

global v1_ini; // Process 1 value
global v2_ini; // Process 2 value
global v3_ini; // Process 3 value

const v2_max; // Maximum value of a process
const m_lost; // Number of lost messages

const init; // Initial state

// Decision Formulas
formula decision = (DC=1) & decision_OP | ((DC=3) & decision_PS); // Combine all decision criteria in a single formula

// Optimistic Decision
formula decision_OP = w1_v2 & w1_v3; // Process 1 has complete view at RN

// Pessimistic Decision
formula decision_PS = w1_v2 & w1_v3 & w1_c2 & w1_c3; // Process 1 has complete view at (RN-1) and has received complete view from all processes at RN

// General Decision Formula
formula decision = (DC=1) & decision_OP | (DC=3) & decision_PS; // Combine all decision criteria in a single formula

module Process_2;

endmodule

module Process_3;

endmodule

module Process_1;

endmodule

Listing 48. N=3 with Symmetric Failures
// Last round, Process 1 computes the messages of other processes: updates its confirmations and, only for OP, updates value and views;

// Not last round, Process 1 computes the messages of other processes: updates its value, views and confirmations;

// Process 1 sends or loses its message;

// Process 1 view of other Processs

// Process 1 current state

// Process 1 decision

// Process 1 has confirmation from Process 3

// Process 1 has confirmation from Process 4

// Process 1 has confirmation from Process 2

// Process 1 has the view of Process 4

// Process 1 has the view of Process 3

// Process 1 has the view of Process 2

// Process 1 has the view of other Processes

// Process 1 has complete view at (RN-1) and has received complete view from all processes at RN

// General Decision Formula

// Combine all decision criteria in a single formula
module N=3; // Number of processes in the network (N cannot be modified)
module N=2; // Number of rounds in the protocol (N=2)
constant double P=0.5; // Probability of losing a message (0<P<1)
constant int Q=0.5; // Decision criterion: OP=1, PS=Q
constant int N; // Identity number of Process
constant int RN; // Identity number of Process
constant int RN1; // Identity number of Process 1
constant bool max; // Maximum value of a process
constant int v_init; // Initial value of Process 1
constant int v2_init; // Initial value of Process 2
constant int v3_init; // Initial value of Process 3
constant bool not_last; // Auxiliary constant to define the next process
constant bool last; // Auxiliary constant to define the next process
constant bool init; // Initialize constant to define the next process

module Process_1;
const int N1=1; // Process 1 identity number
const int v1_ini=1; // Process 1 initial value
const int v2_ini=2; // Process 2 initial value
const int v3_ini=3; // Process 3 initial value
const int v1_ext=v1_ini; // Process 1 external value
const int v2_ext=v2_ini; // Process 2 external value
const int v3_ext=v3_ini; // Process 3 external value
const int N2=2; // Identity number of Process 2
const int N3=3; // Identity number of Process 3
const int Q=0.5; // Probability of losing a message (0<=Q<=1)
const int RN=2; // Number of rounds in the protocol (RN>=2)
const int N=3; // Number of processes in the network (N cannot be modified)

// Process 1 has the view of Process 2
w1_v2_ext : // Process 1 view of other processes
w1_v3_ext : // Process 1 view of other processes
w2_v1_ext : // Process 1 view of other processes
w2_v3_ext : // Process 1 view of other processes
w3_v1_ext : // Process 1 view of other processes
w3_v2_ext : // Process 1 view of other processes

// Process 1 has confirmation from Process 2
n1_nf2 : // Status of the message from the other process
n1_nf3 : // Status of the message from the other process

// Process 1 has complete view at RN
v1 : [0..v_max]; // Process 1 value
RN1 : [0..RN]; // Process 1 round

d1 : // Status of the message from the other process
v1_ext : [0..v_max]; // Process 1 extension value
w1_v2_ext : [0..v_max]; // Process 1 view of other processes extension
w1_v3_ext : [0..v_max]; // Process 1 view of other processes extension

// Process 1 has not received the message of Process 2
n1_nf2 : bool init false; // Process 1 has not received the message of Process 2
n1_nf3 : bool init false; // Process 1 has not received the message of Process 3
n1_nf4 : bool init false; // Process 1 has not received the message of Process 3

// Process 1 has confirmation that other processes have complete view
n1_nf2 : bool init true; // Process 1 has received the message of Process 2
n1_nf3 : bool init true; // Process 1 has received the message of Process 3

// Process 1 has complete view at RN
v1-ext = v1; // Process 1 has complete view at RN
w1-v2-ext = w1-v2; // Process 1 has complete view at RN
w1-v3-ext = w1-v3; // Process 1 has complete view at RN

// Process 1 decides -> agree or abort
formula decision = ((DC=1) & decision_OP) | ((DC=3) & decision_PS); // Combine all decision criterea in a single formula

// General Decision Formula
formula decision_PS = w1_v2 & w1_v3 & w1_c2 & w1_c3; // Process 1 has complete view at (RN-1) and has received complete view from all processes at RN
formula decision_OP = w1_v2 & w1_v3; // Process 1 has complete view at RN

formula w1_c2_new = w1_c2 | (n1_nf2 & (w2_v1_ext & w2_v3_ext)); // Process 1 knows that Process 2 view is complete
formula w1_v3_new = w1_v3 | n1_nf3 | (n1_nf2 & w2_v3_ext); // Process 1 update its view of Process 3
formula w1_v2_new = w1_v2 | n1_nf2 | (n1_nf3 & w3_v2_ext); // Process 1 update its view of Process 2

formula v1_new = // Process 1 computes the messages of other processes: updates its value, views and confirmations;

formula next = N1*not_last+1; // Define the next Process in the network

global token : [1..R]; // Token used to coordinate the processes

global RN1 : [0..RN]; // Identity number of Process 1

global s1 : N+2 & token=N1 -> 1: (s1'=0) & (token'=next) & (d1'= decision); // Process 1 decides -> agree or abort

global s1 : N+1 & token=N1 & RN1=RN -> 1: (s1'=N+2) & (w1_c2'=w1_c2_new) & (w1_c3'=w1_c3_new) & (v1'=(DC=1)?v1_new:v1) & (w1_v2'==(DC=1)?w1_v2_new:w1_v2) & (w1_v3'==(DC=1)?w1_v3_new:w1_v3); // Process 1 has complete view at RN

Listing 50. N=3 with Asymmetric Failures
Listing 51. N>3 with Asymmetric Failures- part 1
// Pessimistic Decision
formula decision_PS = w1_v2 & w1_v3 & w1_v4 & w1_c2 & w1_c3 & w1_c4; // Process 1 has complete view at (RN-1) and received complete view from all processes at RN

// General Decision Formula
formula decision = (DC=1) & decision_OP | (DC=3) & decision_PS; // Combine all decision criteria in a single formula

module Process_1
s1 : [0..N+2] init 1; // Process 1 current state
RN1: [0..RN] init 0; // Current round
v1 : [0..v_max] init v1_ini; // Process 1 value
d1: bool init false; // Process 1 decision
n1_nf2: bool init true; // Process 1 has not received the message of Process 2
n1_nf3: bool init true; // Process 1 has not received the message of Process 3
n1_nf4: bool init true; // Process 1 has not received the message of Process 4
w1_v2 : bool init false; // Process 1 has the view of Process 2
w1_v3 : bool init false; // Process 1 has the view of Process 3
w1_v4 : bool init false; // Process 1 has the view of Process 4
w1_c2 : bool init false; // Process 1 has confirmation from Process 2
w1_c3 : bool init false; // Process 1 has confirmation from Process 3
w1_c4 : bool init false; // Process 1 has confirmation from Process 4

// Process 1 sends its message;
[] s1=1 & token=N1 & RN1<RN -> 1:(s1'=2) & (token'=next) & (v1_ext'=v1) & (w1_v2_ext'=w1_v2) & (w1_v3_ext'=w1_v3) & (w1_v4_ext'=w1_v4) & (RN1'=RN1+1);

// Process 1 receives or loses the message of each other process
[] s1=2 & token=N1 & RN1<=RN & (m_lost<(RN*N*(N-1))) -> (1-Q): (s1'=3) & (n1_nf2'=true) + Q: (s1'=3) & (n1_nf2'=false) & (m_lost'=m_lost+1);
[] s1=3 & token=N1 & RN1<=RN & (m_lost<(RN*N*(N-1))) -> (1-Q): (s1'=4) & (n1_nf3'=true) + Q: (s1'=4) & (n1_nf3'=false) & (m_lost'=m_lost+1);
[] s1=4 & token=N1 & RN1<=RN & (m_lost<(RN*N*(N-1))) -> (1-Q): (s1'=5) & (n1_nf4'=true) + Q: (s1'=5) & (n1_nf4'=false) & (m_lost'=m_lost+1);

// Not last round, Process 1 computes the messages of other processes: updates its value, views and confirmations;
[] s1=N+1 & token=N1 & RN1<RN -> 1: (s1'=1) & (v1'=v1_new) & (w1_v2'=w1_v2_new) & (w1_v3'=w1_v3_new) & (w1_v4'=w1_v4_new) & (w1_c2'=w1_c2_new) & (w1_c3'=w1_c3_new) & (w1_c4'=w1_c4_new) & (token'=next);

// Last round, Process 1 computes the messages of other processes: updates its confirmations and, only for OP, updates value and views;
[] s1=N+1 & token=N1 & RN1=RN -> 1: (s1'=N+2) & (w1_c2'=w1_c2_new) & (w1_c3'=w1_c3_new) & (w1_c4'=w1_c4_new) & (v1'=(DC=1)?v1_new:v1) & (w1_v2'=(DC=1)?w1_v2_new:w1_v2) & (w1_v3'=(DC=1)?w1_v3_new:w1_v3) & (w1_v4'=(DC=1)?w1_v4_new:w1_v4) & (w1_c2'=w1_c2_new) & (w1_c3'=w1_c3_new) & (w1_c4'=w1_c4_new) & (token'=next);

// Process 1 decides -> agree or abort
[] s1=N+2 & token=N1 -> 1: (s1'=0) & (token'=next) & (d1'= decision);

eendmodule