Interactive Consistency Algorithms based on Voting and Error-correcting Codes

Thijs Krol

University of Twente, Department of Computer Science
P.O. Box 217, NL7500 AE Enschede, The Netherlands
email: krol@cs.utwente.nl

Abstract

This paper[1] presents a new class of synchronous deterministic non-authenticated algorithms for reaching interactive consistency (Byzantine agreement). The algorithms are based on voting and error-correcting codes and require considerably less data communication than the original algorithm, whereas the number of rounds and the number of modules meet the minimum bounds. These algorithm based on voting and coding will be defined and proved on the basis of a class of algorithms, called the Dispersed Joined Communication algorithms.

1 Introduction

In fault-tolerant systems data are always replicated in some way or are encoded by means of error-correcting codes. Keeping the data consistent, i.e. correct modules contain correct data can easily be obtained if the data results from a source which is replicated too. However fault-tolerant systems will always be connected to other systems based on different methods for reliability improvement and in any case will be connected to basically unreliable input devices. These unreliable sources might cause a fault-tolerant system to break down even if the fault-tolerant system does contain no more faults than it is designed to tolerate. The root of the problem is in the broadcasting of the data by the external source to the N modules of the fault-tolerant system.

In the end of the seventies this observation led to the definition of the interactive consistency problem or Byzantine Generals problem [2, 12] and a first solution of the Byzantine Generals algorithm in the early eighties, [11].

If an algorithm runs on a system consisting of N modules of which one is the source, and if in this system at most T modules behave maliciously, this algorithm is said to fulfill the *interactive consistency requirements* when the following conditions are fulfilled, regardless of which modules are faulty and what data was sent by the source:

- *The well-functioning modules agree among each other on the data they think they have received from the source.*
- *If the source is well-functioning, the above-mentioned agreement should equal the data actually sent by the source.*

The prime parameters that characterize interactive consistency algorithms are the number of modules, N, the maximum number of faulty modules, T, that can be tolerated for the algorithm fulfilling its requirements, the number of *rounds* K, i.e. the maximum number of times a message is relayed, the connectivity of the graph which represents the communication possibilities and the total amount of data which has to be transmitted between the modules.

Next to these parameters there are several other parameters that characterize interactive consistency algorithms, such as the synchrony of the system, the fault behavior of the modules, and the way in which the algorithm terminates.

In *synchronous* systems, modules relay messages within a commonly known limited time span, which allows receivers to detect modules which refused to relay data.

When messages are *authenticated*, modules which mutilate the message instead of relaying it can always be detected. In case the messages are not authenticated, faulty modules may mutilate the message arbitrarily and may send different messages in different directions.

In this paper we will focus on **synchronous deterministic non-authenticated algorithms**.

[1]The work described in this paper has been carried out at the Philips Research Laboratories Eindhoven as a part of the Philips Research program

Synchronous deterministic non-authenticated algorithms for reaching interactive consistency can only exist if $N \geq 3T+1$,[11] and $K \geq T+1$, [6, 5]. The connectivity of the graph which represents the communication possibilities should be at least $2T+1$. Existing algorithms are the Pease-algorithm [11] which requires $\mathcal{O}(N^T)$ messages to be transmitted and the Dolev-algorithm [4] which requires $\mathcal{O}(N.T+T^3 \log T)$ messages but $2T+3$ rounds.
Synchronous deterministic authenticated algorithms have to satisfy $N \geq T+1$, $K \geq T+1$ and graph connectivity $T+1$. A complete overview of the existing interactive consistency algorithms is presented in [1].

In the following we will present a class of synchronous deterministic algorithms which solves the interactive consistency problem for all values $N \geq 3T+1$ and $K \geq T+1$. The algorithms in this class have similarities with the algorithm which we will call the Pease-algorithm presented in [11, 3] and with (N,K)-concept fault-tolerance, [9, 8, 7]. The Pease algorithm is a member of this class. The amount of messages which needs to be transmitted between the modules is reduced considerably by reducing the number of directions in which a message is forwarded and by replacing the broadcast functions by the encoder functions of error-correcting codes and simultaneously replacing the voting function in the decision-making process by the decoder functions of the error-correcting codes applied in the broadcast process.
This new class of Interactive Consistency algorithms which is based on voting and error-correcting codes will be derived from a new class of algorithms which we will call Dispersed Joined Communication (DJC) algorithms, cf. figure 1. The latter class of algorithms satisfies more liberal properties than those which are required for the Interactive Consistency (IAC) algorithms.

2 The Dispersed Joined Communication Algorithms

2.1 Introduction

In this section we will define a class of algorithms which will be called Dispersed Joined Communication (DJC) algorithms. These aim to transmit a message from a single source module to all modules in the system in the presence of a number of maliciously behaving modules.

A Dispersed Joined Communication algorithm satisfies the following properties:

1. If the source and destination are both functioning correctly, then the decision calculated by the decision-making process in the destination equals the original message in the source.
2. For any two destinations which are both functioning correctly it holds that if the results calculated in these destinations are unequal, then the number of maliciously behaving modules in the system is at least K.

These requirements are similar to the interactive consistency requirements mentioned above, except that inconsistency may occur if the number of faulty modules in the system exceeds the number of rounds.

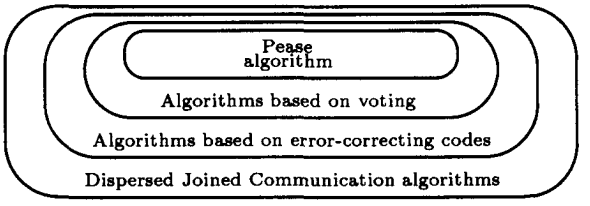

Figure 1: The classes of algorithms discussed in this paper.

A DJC algorithm prescribes the way in which the message is forwarded through the network from the source to the other modules, the way in which the messages are modified by the modules, and the way in which the final result is calculated in the destinations.
DJC algorithms are based on the following ideas:
In order to be able to tolerate modules which behave maliciously, the communication between the source and the destinations is *dispersed*, i.e. the message which needs to be transmitted is sent possibly in differently modified versions, via different paths from the source to the destinations.
This means that the message is encoded by means of an error-correcting code and each symbol of the codeword is transmitted via a different path to the destination. So let a source module be identified by a and let two particular destinations be identified by d and e respectively. Furthermore the system contains modules $b_0, b_1, \ldots, b_{n-1}$. The message $m(a)$ in the source a is encoded into n symbols and these symbols are sent via the paths (a, b_0, d), (a, b_1, d),...., (a, b_{n-1}, d) to the destination d and similarly via the paths (a, b_0, e), (a, b_1, e),...., (a, b_{n-1}, e) to the destination e. So the message is *dispersed* but the first step of communication from a to d and from a to e is *joined*. The set $\{b_0, b_1, \ldots, b_{n-1}\}$ of modules to which a sends its message, is called the next-set $\mathbf{B}(a)$ of module a. Modification of the original message $m(a)$ is implemented by

encoding the message by means of a T-symbol-error-correcting code, for instance a Reed-Solomon code [10]. Let the encoding function performed in module a be represented by $\mathcal{Y}$, then the message (symbol) $m(a, b_i)$ which is sent from the source to module b_i and which is received by b_i is denoted by

$$m(a, b_i) = \mathcal{Y}(b_i)(m(a)) \quad (1)$$

in which $\mathcal{Y}(b_i)$ is called the partial encoder function of the encoder function $\mathcal{Y}$ which delivers the symbol which has to be sent to module b_i.
The modules b_i relay the received messages unchanged to the destinations d and e, provided the modules b_i are functioning correctly. So if a, b_i and d are functioning correctly, the message $m(a, b_i, d)$ received by d via b_i from a is described by

$$m(a, b_i, d) = \mathcal{Y}(b_i)(m(a)) \text{ for all } i \in \mathbf{B}(a) \quad (2)$$

and similarly if a, b_i an e are functioning correctly

$$m(a, b_i, e) = \mathcal{Y}(b_i)(m(a)) \text{ for all } i \in \mathbf{B}(a) \quad (3)$$

So the destination d and e both receive the n symbols of the T-symbol-error-correcting codeword.
In module d the decoder function $\mathcal{Y}^{(-1)}$ of the T-symbol-error-correcting code is applied on the received messages $m(a, b_i, d)$ with $b_i \in \mathbf{B}(a)$. The result of this decoding (and error-correction) function is denoted by $dec((a), d)$, i.e. the decision taken in d about what a has sent.
The same is done in module e resulting in $dec((a), e)$.
From the preceding description and the properties of the symbol-error-correcting code we observe:

- If the number of faulty modules in the system is less than or equal to T and the source module a and a destination d are both functioning correctly then the result $dec((a), d)$ calculated in module d equals the original message $m(a)$ in the source.

Assume that the number of faulty modules in the system is T or less and that two destinations d and e are functioning correctly, then from the preceding observation we find that if module a is functioning correctly then for the decisions calculated in the modules d and e holds:

$$m(a) = dec((a), d) = dec((a), e) \quad (4)$$

And thus $dec((a), d) \neq dec((a), e)$ implies that module a is faulty.
If a module b_i is functioning correctly it relays identical symbols to the modules d and e. So if b_i is functioning correctly then $m(a, b_i, d) = m(a, b_i, e)$. In the destinations d and e the **same** decoder function $\mathcal{Y}^{(-1)}$ is applied on the received messages $m(a, b_i, d)$ and $m(a, b_i, e)$ with $b_i \in \mathbf{B}(a)$ and thus if all modules b_i are functioning correctly, it must hold that the decisions calculated in d and e are equal. Hence $dec((a), d) = dec((a), e)$ regardless whether module a is functioning correctly or not. Consequently, if $dec((a), d) \neq dec((a), e)$ then at least one module b_i must be faulty. This leads to the second observation

- If the number of faulty modules in the system is less than or equal to T, then for two destinations d and e which are both functioning correctly it holds that if the results $dec((a), d)$ and $dec((a), e)$ are unequal, both the source module and at least one intermediate module is faulty and thus the number of faulty modules in the system is at least two.

The DJC algorithms and the IAC algorithms derived from them, are based on a recursive implementation the preceding system model and its observations.

2.2 The construction of the Dispersed Joined Communication Algorithms

Dispersed Joined Communication algorithms are defined on a set **Ns** of N fully interconnected modules of which no more than T are behaving maliciously. A particular DJC algorithm aims at sending a particular message from a particular *source module* a to all modules in **Ns**, by means of K rounds of information exchange, cf. figure 2.

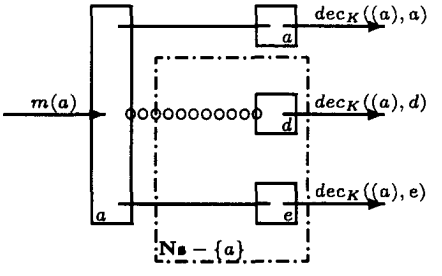

Figure 2: A pictural representation of an algorithm in the class $\mathcal{A}(T, K, a, \mathbf{Ns})$. Direct communication is indicated by —, communication via other modules by o o o.

In general there will exist many DJC algorithms which have the same properties. Therefore we define classes

$$\mathcal{A}(T, K, a, \mathbf{Ns}) \quad (5)$$

of DJC algorithms.
These classes are only defined if

$$K \geq 1 \text{ and } a \in \mathbf{Ns} \text{ and } |\mathbf{Ns}| \geq K + 1 \quad (6)$$

The first two requirements are obvious, the last $|\mathbf{Ns}| \geq K+1$ is added in order to exclude some pathological classes. If the constraints in (6) are not satisfied the class $\mathcal{A}(T, K, a, \mathbf{Ns})$ is empty by definition.

A particular algorithm in a class $\mathcal{A}(T, K, a, \mathbf{Ns})$ lays down in detail the way in which a message travels from the source a to the destinations via different parallel paths in K rounds of information exchange. Moreover the algorithm prescribes the way in which the messages are modified during their journey though the network and the way in which in each destination d a decision is calculated starting from all data received by d.

The original message in the source is denoted by

$$m(\underline{\mathbf{s}}, a) \quad \text{or by} \quad m(a) \tag{7}$$

The prefix $\underline{\mathbf{s}}$ to the source module identifier a is only used if we need to distinguish between different messages in the same module a and in that case denotes the path along which the message traveled to module a.

If a message $m(\underline{\mathbf{s}}, a)$ (or $m(a)$) is sent to the modules in the set $\mathbf{Ns}$ by means of an algorithm from the class $\mathcal{A}(T, K, a, \mathbf{Ns})$, then the results calculated in the modules d, with $d \in \mathbf{Ns}$, are denoted by

$$dec_K((\underline{\mathbf{s}}, a), d) \quad (\text{or by} \quad dec_K((a), d) \tag{8}$$

The algorithms in the classes $\mathcal{A}(T, K, a, \mathbf{Ns})$ will be defined recursively with respect to K. The basis of the recursion is the case $K = 1$.

The construction of the algorithms in the class $\mathcal{A}(T, 1, a, \mathbf{Ns})$

An algorithm in the class $\mathcal{A}(T, 1, a, \mathbf{Ns})$ is based on only one round of information exchange and is only defined if $a \in \mathbf{Ns}$ and $|\mathbf{Ns}| \geq 2$, cf. (6).

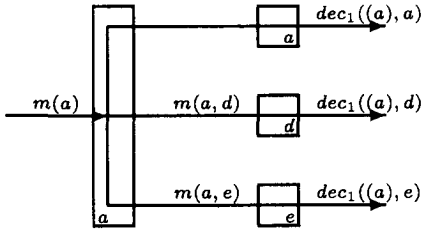

Figure 3: A pictural representation of an algorithm in the class $\mathcal{A}(T, 1, a, \mathbf{Ns})$.

The class $\mathcal{A}(T, 1, a, \mathbf{Ns})$ contains the following algorithm:
During round 0, the source module a sends the original message $m(a)$ directly and unchanged to all modules in $\mathbf{Ns} - \{a\}$. Furthermore module a keeps a copy of the message $m(a)$ itself in order to be used in the decision-making process. The messages received by the modules d in $\mathbf{Ns} - \{a\}$ from module a are denoted by $m(a, d)$, cf. figure 3.
After round 0 the decision-making process is executed in which in each module d, with $d \in (\mathbf{Ns} - \{a\})$, the message $m(a, d)$ received from a is taken as decision $dec_1((a), d)$ and the decision $dec_1((a), a)$ in module a will be equal to the stored message $m(a)$.
So the behavioral aspects of the algorithms in the class $\mathcal{A}(T, 1, a, \mathbf{Ns})$ (starting from correctly functioning modules) are defined by:

$$\begin{aligned} d \in (\mathbf{Ns} - \{a\}) &\Longrightarrow m(a, d) = m(a) \\ d \in (\mathbf{Ns} - \{a\}) &\Longrightarrow dec_1((a), d) = m(a, d) \\ dec_1((a), a) &= m(a) \end{aligned} \tag{9}$$

The recursive construction of the algorithms in the class $\mathcal{A}(T, K, a, \mathbf{Ns})$ with $K > 1$ by means of algorithms from the set of classes

$$\mathcal{A}(T, K-1, b, \mathbf{Ns} - \{a\}).$$

The construction of the algorithms in the class $\mathcal{A}(T, K, a, \mathbf{Ns})$ with $K > 1$ consist of two part, viz. a *broadcast process* in which the message is transferred in K rounds $0, \ldots, K-1$, from the source to the destinations and in the meantime modified by the involved modules and a *decision making process* which is executed after round $K-1$ in each module of the system.

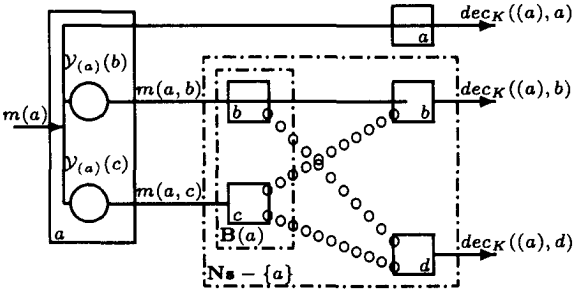

Figure 4: The construction of an algorithm in the class $\mathcal{A}(T, K, a, \mathbf{Ns})$.

The *broadcast process* is based on encoding the original message $m(a)$ in the source into symbols of a T-error-correcting code, thereafter transmitting during round 0 each symbol to a different module b in the *next-set* $\mathbf{B}(a)$, $b \in \mathbf{B}(a)$ and $\mathbf{B}(a) \subset (\mathbf{Ns} - \{a\})$, which forwards the received symbol during the rounds $1 \cdots K-1$, to the destinations in $\mathbf{Ns} - \{a\}$ by means of an algorithm from the class $\mathcal{A}(T, K-1, b, \mathbf{Ns} - \{a\})$. Furthermore $m(a)$ is kept stored in a in order to be used later in the decision making process in a, cf. figure 4.

The algorithms from the class $\mathcal{A}(T, K-1, b, \mathbf{Ns}-\{a\})$ result in decisions $dec_{K-1}((a,b),d)$, i.e. the decision taken in d about what b received from a. The decision $dec_K((a),d)$ in d about the message $m(a)$ is calculated in each d by applying the decoder function of the error-correcting code on these symbols $dec_{K-1}((a,b),d)$ with $b \in \mathbf{B}(a)$. In module a, the decision $dec_K((a),a)$ gets directly the value of the stored message $m(a)$, cf. figures 5 and 6.

Let $\mathcal{Y}_{(a)}$ be the encoder function of some T-error-correcting code. The corresponding decoder function is denoted by $\mathcal{Y}_{(a)}^{(-1)}$, cf section 2.1. Let the original message be $m(a)$ and the symbol which is sent to b be denoted by $m(a,b)$, then this symbol is related to $m(a)$ by

$$m(a,b) = \mathcal{Y}_{(a)}(b)(m(a)) \tag{10}$$

in which the $\mathcal{Y}_{(a)}(b)$ is called the partial encoder function which delivers the symbol which has to be sent to module b.

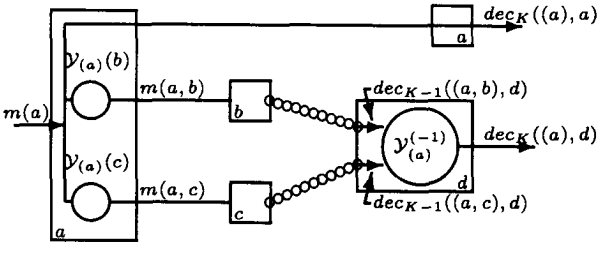

Figure 5: The way in which the decision $dec_K((a),d)$ in a module d with $d \notin \mathbf{B}(a)$ is obtained from the partially encoded messages sent by module a to the modules in $\mathbf{B}(a)$.

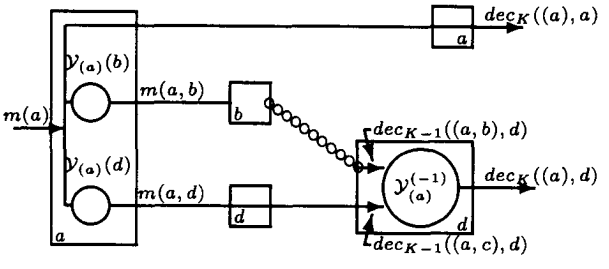

Figure 6: The way in which the decision $dec_K((a),d)$ in a module d with $d \in \mathbf{B}(a)$ is obtained from the partially encoded messages sent by a to the modules in $\mathbf{B}(a)$.

The behavioral aspects of the algorithms in the class $\mathcal{A}(T, K, a, \mathbf{Ns})$ can thus be summarized as follows:

$$\begin{aligned}
&m(a,b) = \mathcal{Y}_{(a)}(b)(m(a)) \quad \text{for all} \quad b \in \mathbf{B}(a) \\
&dec_{K-1}((a,b),d) \quad \text{follows from} \quad m(a,b) \\
&\qquad \text{based on an algorithm from the class} \\
&\qquad\qquad \mathcal{A}(T, K-1, b, \mathbf{Ns}-\{a\}) \\
&d \neq a \Longrightarrow dec_K((a),d) = \mathcal{Y}_{(a)}^{(-1)} \quad \text{applied on the} \\
&\qquad \text{values} \quad dec_{K-1}((a,b),d) \quad \text{with} \quad b \in \mathbf{B}(a) \\
&d = a \Longrightarrow dec_K((a),d) = m(a)
\end{aligned} \tag{11}$$

2.3 The existence of Dispersed Joined Communication Algorithms in the classes $\mathcal{A}(T, K, a, \mathbf{Ns})$

The next step is to investigate for which parameters DJC algorithms can be constructed or stated in a different way, for which parameters the classes $\mathcal{A}(T, K, a, \mathbf{Ns})$ are non-empty.

Theorem 1 *For any fully interconnected system consisting of* $|\mathbf{Ns}|$ *modules and* $T \geq 1$*:*
the class of algorithms $\mathcal{A}(T, 1, a, \mathbf{Ns})$ *is non-empty if and only if*

$$a \in \mathbf{Ns} \quad \textit{and} \quad |\mathbf{Ns}| \geq 2$$

and if $K \geq 2$ *the class of algorithms* $\mathcal{A}(T, K, a, \mathbf{Ns})$ *is non-empty if and only if*

$$a \in \mathbf{Ns} \quad \textit{and} \quad |\mathbf{Ns}| \geq 2T + K$$

□

Proof:

The first part of the theorem, i.e. $(K = 1)$ trivially follows from the definition. In a fully interconnected system a message can always be sent directly from the source to the destination.
The second part $(K \geq 2)$ will be proved by induction on K. The basis of the induction will be $K = 2$.

If $K \geq 2$ then from the definition of the algorithms of the class $\mathcal{A}(T, K, a, \mathbf{Ns})$ we recall that if any of the constraints $T \geq 1$, $a \in \mathbf{Ns}$, $|\mathbf{Ns}| \geq K+1$ is not satisfied, the class $\mathcal{A}(T, K, a, \mathbf{Ns})$ is empty. Because $T \geq 1$ and $|\mathbf{Ns}| \geq 2T + K$ implies $|\mathbf{Ns}| \geq K + 1$, we will only have to prove that class $\mathcal{A}(T, K, a, \mathbf{Ns})$ is non-empty if and only if $|\mathbf{Ns}| \geq 2T + K$

Let

$$T \geq 1 \text{ and } K \geq 2 \text{ and } a \in \mathbf{Ns} \tag{12}$$

The construction of algorithms is possible if and only if

1. A T-error-correcting code exists of which the code words consist of $|\mathbf{B}(a)|$ symbols.
2. A next-set $\mathbf{B}(a)$ can be found such that $\mathbf{B}(a) \subset \mathbf{Ns} - \{a\}$.
3. The classes $\mathcal{A}(T, K-1, b, \mathbf{Ns} - \{a\})$ of DJC algorithms with $b \in \mathbf{B}(a)$ are all non-empty.

The first requirement can always be fulfilled if $|\mathbf{B}(a)| \geq 2T + 1$, [10]. Hence, the second requirement can be satisfied if and only if

$$|\mathbf{Ns}| \geq 2T + 2 \tag{13}$$

The third requirement is fulfilled only if the general constraints of the classes $\mathcal{A}(T, K-1, b, \mathbf{Ns} - \{a\})$ are satisfied, i.e.

$$b \in (\mathbf{Ns} - \{a\}) \text{ and } |\mathbf{Ns} - \{a\}| \geq K \tag{14}$$

The first part of predicate (14) trivially follows from the construction of the algorithms in the class $\mathcal{A}(T, K, a, \mathbf{D}, \mathbf{Ns})$ which requires both $b \in \mathbf{B}(a)$ and $\mathbf{B}(a) \subset \mathbf{Ns} - \{a\}$.
The second part is satisfied if we are able to prove that $|\mathbf{Ns}| \geq 2T + K$.
Let $K = 2$, then from (13) we recall the only remaining requirement

$$|\mathbf{Ns}| \geq 2T + 2 \tag{15}$$

Which proves Theorem 1 for $K = 2$.

Suppose Theorem 1 holds for $K - 1$ with $K \geq 3$. So suppose if $K \geq 3$ the class of algorithms $\mathcal{A}(T, K-1, a, \mathbf{Ns})$ is non-empty if and only if

$$a \in \mathbf{Ns} \text{ and } |\mathbf{Ns}| \geq 2T + K - 1 \tag{16}$$

Recall that the class of algorithms $\mathcal{A}(T, K, a, \mathbf{Ns})$ is non-empty if and only if (13), i.e. $|\mathbf{Ns}| \geq 2T + 2$, is satisfied and the classes $\mathcal{A}(T, K-1, b, \mathbf{Ns} - \{a\})$ of DJC algorithms with $b \in \mathbf{B}(a)$ are all non-empty.
The latter holds if the general constraints in (14) are satisfied, which already has been proved and if and only if $|\mathbf{Ns} - \{a\}| \geq 2T + K - 1$ according to (16). So $|\mathbf{Ns}| \geq 2T + K$ which satisfies (13).

Hence the assumption (16) implies that for all $T, a, \mathbf{Ns}$ and $K \geq 3$ the class of algorithms $\mathcal{A}(T, K, a, \mathbf{Ns})$ is non-empty if and only if

$$a \in \mathbf{Ns} \text{ and } |\mathbf{Ns}| \geq 2T + K \tag{17}$$

We already proved the theorem for $K = 2$ and thus by induction on K we obtain that (17) holds for any $K \geq 2$.
Which completes the proof of Theorem 1. □

2.4 Some behavioral properties of the DJC algorithms

DJC algorithms satisfy the following behavioral properties:

Theorem 2 *Let a message $m(a)$ be transmitted by means of any DJC algorithm from the class $\mathcal{A}(T, K, a, \mathbf{Ns})$ to the destinations given by the set $\mathbf{Ns}$ in the presence of at most T faulty modules then:*

1. *If the source module a and a destination d are both functioning correctly then the result $dec_K((a), d)$ of the algorithm calculated in module d equals the original message $m(a)$ in module a.*
2. *For any two destinations d and e which are both functioning correctly it holds that if the results $dec_K((a), d)$ and $dec_K((a), e)$ are unequal, then the number of maliciously behaving modules in the system is at least K.* □

Proof:

We start with property 1.
Suppose the source module a and a destination d are both functioning correctly.
If $a = d$ then according to the construction of the algorithms $\mathcal{A}(T, K, a, \mathbf{Ns})$, it holds that $dec_K((a), d) = m(a)$.
So we need only to consider the case $d \neq a$, i.e. $d \in (\mathbf{Ns} - \{a\})$. For these cases we prove $dec_K((a), d) = m(a)$ by induction with respect to K.
Basis: $K = 1$.
Algorithms from the class $\mathcal{A}(T, 1, a, \mathbf{Ns})$ send the message $m(a)$ directly and unchanged to the destination d and the decision taken in module d equals the message received from a. So because we assume that a and d are functioning correctly it holds that $dec_1((a), d) = m(a)$.

Induction step: $K > 1$
The algorithms in a class $\mathcal{A}(T, K, a, \mathbf{Ns})$ have been constructed from the algorithms in the class $\mathcal{A}(T, K-1, b, \mathbf{Ns} - \{a\})$ with $b \in \mathbf{B}(a)$ and $\mathbf{B}(a) \subset (\mathbf{Ns} - \{a\})$.
For the latter algorithms we know from the induction hypothesis that if a message $m(a, b)$ is communicated by a correctly functioning source module b to a correctly functioning destination d, then

$$dec_{K-1}((a, b), d) = m(a, b) \tag{18}$$

From the construction we know that in module a by means of the encoder function $\mathcal{Y}_{(a)}$, the message $m(a)$ is encoded into $|\mathbf{B}(a)|$ symbols which during round 0

each are sent to a different module b, with $b \in \mathbf{B}(a)$. So

$$m(a,b) = \mathcal{Y}(a)(b)(m(a)) \qquad (19)$$

cf. (11). Each module b thereafter forwards the message $m(a,b)$ (is the code word symbol) to the destinations represented by the set $\mathbf{Ns} - \{a\}$ by means of an algorithm from the class $\mathcal{A}(T, K-1, b, \mathbf{Ns} - \{a\})$.
As the result of these algorithms, in each destination d of the set $\mathbf{Ns} - \{a\}$, $|\mathbf{B}(a)|$ decisions $dec_{K-1}((a,b),d)$ will become available. If module b is functioning correctly then according to (18) and (19) it holds that

$$dec_{K-1}((a,b),d) = \mathcal{Y}(a)(b)(m(a)) \qquad (20)$$

In each module d the decoder function $\mathcal{Y}_{(a)}^{(-1)}$ is applied to these decisions $dec_{K-1}((a,b),d)$ with $b \in \mathbf{B}(a)$. At most T modules b behave maliciously so at most T decisions do not satisfy (20). The code $\mathcal{Y}_{(a)}$ is T-error-correcting and thus $dec_K((a),d)$ which is the result of applying the decoder function on the values $dec_{K-1}((a,b),d)$ must be equal to $m(a)$.
This completes the proof of property 1.

Next we prove property 2.
Let us assume that two destination d and e, with $d, e \in \mathbf{Ns}$ behave correctly and that $dec_K((a),d) \neq dec_K((a),e)$.
If the source module a is functioning correctly then, by property 1 it holds that

$$dec_K((a),d) = m(a) = dec_K((a),e)$$

conflicting the assumption $dec_K((a),d) \neq dec_K((a),e)$. So we conclude that module a is behaving maliciously and thus $d \neq a$ and $e \neq a$, or $d, e \in (\mathbf{Ns} - \{a\})$.
Again we use induction with respect to K.
Basis: K=1
We already concluded from our assumption $dec_K((a),d) \neq dec_K((a),e)$ and d and e both behaving correctly, that module a is behaving maliciously. Hence the system contains at least one maliciously behaving module.

Induction step: $K > 1$
Recall that during round 0 the symbols are sent by module a to the modules b with $b \in \mathbf{B}(a)$ and thereafter forwarded by means of algorithms from the classes $\mathcal{A}(T, K-1, b, \mathbf{Ns} - \{a\})$. The results of the latter algorithms calculated in the modules d with $d \in (\mathbf{Ns} - \{a\})$ are denoted by $dec_{K-1}((a,b),d)$.

Let $dec_{K-1}((a,b),d)$ and $dec_{K-1}((a,b),e)$ with $b \in \mathbf{B}(a)$ be decisions calculated in modules d and e. Since $d, e \in (\mathbf{Ns} - \{a\})$, the decision $dec_K((a),d)$ is based on applying the function $\mathcal{Y}_{(a)}^{(-1)}$ on the decisions $dec_{K-1}((a,b),d)$ with $b \in \mathbf{B}(a)$, whereas the decision $dec_K((a),e)$ is based on applying the same function $\mathcal{Y}_{(a)}^{(-1)}$ on the decisions $dec_{K-1}((a,b),e)$ with $b \in \mathbf{B}(a)$. It follows that

$$\forall b \,:\, b \in \mathbf{B}(a) \Longrightarrow dec_{K-1}((a,b),d) = dec_{K-1}((a,b),e) \qquad (21)$$

would imply $dec_K((a),d) = dec_K((a),e)$. The latter however is conflicting with the assumption $dec_K((a),d) \neq dec_K((a),e)$ so we must conclude

$$\exists b \,:\, b \in \mathbf{B}(a) \wedge dec_{K-1}((a,b),d) \neq dec_{K-1}((a,b),e) \qquad (22)$$

Recall that our assumption implies $d, e \in (\mathbf{Ns} - \{a\})$. So from the definition of the construction of the algorithms in the class $\mathcal{A}(T, K, a, \mathbf{Ns})$ we know that the decisions $dec_{K-1}((a,b),d)$ are the result of the algorithms from the classes $\mathcal{A}(T, K-1, b, \mathbf{Ns} - \{a\})$ with $b \in \mathbf{B}(a)$.
According to the induction hypothesis it holds for the latter classes that if the modules d and e are both functioning correctly and $dec_{K-1}((a,b),d) \neq dec_{K-1}((a,b),e)$ then the number of maliciously behaving modules in the set $\mathbf{Ns} - \{a\}$ must be at least $K-1$. And thus with (22) we conclude that the set $\mathbf{Ns} - \{a\}$ must contain at least $K-1$ maliciously behaving modules. We already concluded from the assumption that module a behaves maliciously. Hence the set $\mathbf{Ns}$ must contain at least K maliciously behaving modules.
Which completes the proof of Theorem 2. □

3 A class of algorithms for reaching interactive consistency based on voting and coding

The class of algorithms for reaching interactive consistency based on voting and coding is a subclass of the DJC algorithms defined in the previous section and is defined by:

$$\mathcal{A}(T, K, a, \mathbf{Ns}) \text{ with } T \geq 1 \text{ and } K = T + 1 \qquad (23)$$

According to theorem 1 the class $\mathcal{A}(T, K, a, \mathbf{Ns})$ is non-empty if and only if $|\mathbf{Ns}| \geq 2T + K$, hence the class of algorithms $\mathcal{A}(T, T+1, a, \mathbf{Ns})$ is non-empty if and only if $|\mathbf{Ns}| \geq 3T + 1$.

Theorem 3 *The Interactive Consistency algorithms in a class* $\mathcal{A}(T, T+1, a, \mathbf{Ns})$ *with* $|\mathbf{Ns}| \geq 3T+1$, *satisfy*

the interactive consistency requirements, i.e.:

$$\forall d \ : \ a, d \in \overline{\mathbf{F}} \Longrightarrow dec_{T+1}((a), d) = m(a) \quad \textit{and}$$
$$\forall d, e \ : \ d, e \in \overline{\mathbf{F}} \Longrightarrow dec_{T+1}((a), d) = dec_{T+1}((a), e)$$

in which $\overline{\mathbf{F}}$ *is any set of correctly functioning modules such that*

$$|\overline{\mathbf{F}}| \geq |\mathbf{Ns}| - T \quad \textit{and} \quad \overline{\mathbf{F}} \subset \mathbf{Ns}$$

and $dec_{T+1}((a), d)$ *with* $d \in \mathbf{Ns}$ *denotes the decision in a module* d *about what module* a *tried to send.* □

Proof:

Consider Interactive Consistency algorithms which are defined by the non-empty class of DJC algorithms $\mathcal{A}(T, K, a, \mathbf{Ns})$ with

$$T \geq 1 \ \wedge \ |\mathbf{Ns}| \geq 3T + 1 \ \wedge \ K = T + 1 \qquad (24)$$

Moreover let $\overline{\mathbf{F}}$ be any set of correctly functioning modules such that

$$|\overline{\mathbf{F}}| \geq |\mathbf{Ns}| - T \quad \text{and} \quad \overline{\mathbf{F}} \subset \mathbf{Ns}$$

From Theorem 2 we know that if the source module a and a destination d are both functioning correctly then the result $dec_K((a), d)$ of the algorithm calculated in module d equals the original message $m(a)$ in module a. Thus:

$$\forall d \ : \ a, d \in \overline{\mathbf{F}} \Longrightarrow dec_K((a), d) = m(a) \qquad (25)$$

Which proves the first part of the interactive consistency property.

From the second part of Theorem 2 we know that for any two destinations d and e which are both functioning correctly it holds that if the results $dec_K((a), d)$ and $dec_K((a), e)$ are unequal, then the number of maliciously behaving modules is at least K.
However this conflicts with the constraint $K = T + 1$ and the assumption that at most T modules behave maliciously. Thus if both modules d and e are behaving correctly, the decisions $dec_K((a), d)$ and $dec_K((a), e)$ must be identical. So

$$\forall d, e \ : \ d, e \in \overline{\mathbf{F}} \Longrightarrow dec_K((a), d) = dec_K((a), e) \qquad (26)$$

Which completes the proof of Theorem 3. □

4 Some remarks on the construction of IAC algorithms which are based on voting and coding

In this section we will first discuss the design process and the design freedom which is left by the definition of these IAC algorithms.
In the introduction to this paper we claimed that the reduction of the number of messages which needs to be transmitted between the modules can be obtained by minimizing the number of directions in which the possibly modified messages are broadcast each round, and by replacing the voting function by an error-correcting code.
From the construction we immediately see that replacing the voting function by an error-correcting code causes an increase in the number of modules to which a modified message has to be sent. However, the size of these messages decreases. We will show that this decrease is more efficient than reducing the number of directions. It however has to be paid for by a larger minimal size of the original message in the source and by the fact that the implementation of the decoding function of an error-correcting code is more complex than the implementation of a majority voter.
For these reasons we will focus on two subclasses of algorithms, i.e

- The *Minimal-Voting* algorithms, in which the number of directions in which a message is sent, is minimized and in which in the decision-making process only majority voting is applied.
- The *Maximal-Coding* algorithms, in which the messages are broadcast to as many modules as is possible and so maximizing the size reduction of the messages.

Another way of minimizing the amount of messages is to reduce the amount of destinations in which the decisions are calculated to $2T + 1$ and thereafter to broadcast the results in an additional round to the other modules. This so-called *Subset Method* will remain beyond the scope of this paper, see [9].

Consider an IAC-algorithm from the class $\mathcal{A}(T, T + 1, a, \mathbf{Ns})$ and let the number of modules be N, $N = |\mathbf{Ns}|$. The source module is always one of the destinations and thus during round 0 the original message $m(a)$ is kept stored in the source module. Furthermore $m(a)$ is encoded by means of a T-error-correcting code $\mathcal{Y}_{(a)}$ consisting of $n_{(a)}$ symbols of $b_{(a)}$ bits and of which $k_{(a)}$ symbols form the data part. The message size reduction thus is $k_{(a)}^{-1}$. In case of a minimal-voting algorithm $n_{(a)} = 2T+1$ and $k_{(a)} = 1$ and each symbol

equals $m(a)$. Each of these symbols is sent to a different module b with $b \in \mathbf{B}(a)$ and $\mathbf{B}(a) \subset (\mathbf{Ns} - \{a\})$. So in general $n_{(a)} = |\mathbf{B}(a)|$ and $2T+1 \leq n_{(a)} \leq N-1$. A module b forwards these symbols during round $1, \ldots, K-1$ to the destinations by means of an algorithms of the class $\mathcal{A}(T, T, b, \mathbf{Ns} - \{a\})$. So in b the received symbol is kept stored and furthermore $m(a,b)$ is encoded by means of a T-error-correcting code $\mathcal{Y}_{(a,b)}$ consisting of $n_{(a,b)}$ symbols of $b_{(a,b)}$ bits of which $k_{(a,b)}$ symbols form the data part. These symbols are forwarded to modules c with $c \in \mathbf{B}(a,b)$ and $\mathbf{B}(a,b) \subset (\mathbf{Ns} - \{a,b\})$. So $n_{(a,b)} = |\mathbf{B}(a,b)|$ and $2T+1 \leq n_{(a,b)} \leq N-2$.
In general during round t, with $0 \leq t \leq T$ the message $m(a_0, a_1, \ldots, a_t)$ is encoded by $\mathcal{Y}_{(a_0,a_1,\ldots,a_t)}$ and forwarded to modules a_{t+1} from the set $\mathbf{B}(a_0, a_1, \ldots, a_t)$ and $\mathbf{B}(a_0, a_1, \ldots, a_t) \subset (\mathbf{Ns} - \{a_0, a_1, \ldots, a_t\})$. So $n_{(a_0,a_1,\ldots,a_t)} = |\mathbf{B}(a_0, a_1, \ldots, a_t)|$ and $2T+1 \leq n_{(a_0,a_1,\ldots,a_t)} \leq N-t-1$.
A message $m(a_0, a_1, \ldots, a_T)$ received during round T is again kept stored and forwarded to the destinations given by the set $(\mathbf{Ns} - \{a_0, a_1, \ldots, a_T\})$ by means of an algorithm of the class $\mathcal{A}(T, 1, a, \mathbf{Ns} - \{a_0, \ldots, a_T\})$. In this algorithm no coding takes place and the message is directly sent to the destinations.

Obviously the design freedom of the algorithm given N and T is in the choice of the next-sets $\mathbf{B}(a_0, a_1, \ldots, a_t)$ in combination with the choice of the T-error-correcting codes $\mathcal{Y}_{(a_0,a_1,\ldots,a_t)}$. A more detailed discussion can be found in [9].

During the decision-making process executed in a module d, first the decisions $dec_1((a_0, \ldots a_T, d), d)$ are calculated according to the rules of the algorithm from the class $\mathcal{A}(T, 1, a, \mathbf{Ns} - \{a_0, \ldots, a_T\})$, i.e. if $a_T \neq d$ then $dec_1((a_0, \ldots a_T), d) = m(a_0, \ldots a_T, d)$ and $dec_1((a_0, \ldots, a_{T-1}, d), d) = m(a_0, \ldots a_{T-1}, d)$ for all messages received by d during round $T+1$ or kept stored during round T. Next the decisions are calculated according to the rules of the algorithm from the class $\mathcal{A}(T, 2, a, \mathbf{Ns} - \{a_0, \ldots, a_{T-1}\})$, i.e.: if $a_{T-1} \neq d$ then $dec_2((a_0, \ldots, a_{T-1}), d)$ is calculated by applying the decoder function $\mathcal{Y}^{(-1)}_{(a_0,\ldots,a_{T-1})}$ on the decisions $dec_1((a_0, \ldots, a_{T-1}, a_T), d)$ with $a_T \in \mathbf{B}(a_0, \ldots a_{T-1})$ and if $a_{T-1} = d$ then $dec_2((a_0, \ldots, a_{T-2}, d), d) = m((a_0, \ldots, a_{T-2}, d)$.
This process is continued until $dec_{T+1}((a_0), d)$ is found, being the final result of the algorithm in module d.

5 Examples

In this section we will present a few examples and compare them with existing IAC-algorithms. Only synchronous deterministic algorithms without authentication will be taken into account, i.e. the algorithm published by Pease et al., [11] which is a member of our class of algorithms and the one published by Dolev [4]. Starting from the parameters N and T we will compare these algorithms firstly on the number of message bits, *#mess*, that needs to be transmitted between the modules related to the size of the original message and secondly on the minimum size, *msize*, of the original message.

The relative number message bits follows from

$$\#\text{mess} = (N-K).\prod_{t=0}^{K-2}\left(\frac{n_c(t)}{k_c(t)}\right) + \sum_{t=0}^{K-2}\prod_{i=0}^{t}\left(\frac{n_c(i)}{k_c(i)}\right) \tag{27}$$

In which the codes used during round t are all the same and thus can be expressed by the parameters $n_c(t)$, $k_c(t)$ and $b_c(t)$ denoting the number of codeword symbols, the number of data symbols and the number of bits in a symbol respectively. The last term in (27) expresses the message content of the first $K-1$ rounds, the first term the content of round $K-1$.

Because each time a symbol is encoded again and the minimum symbol size of a T-error-correcting code is determined by $\lceil \log_2(n_c - 1) \rceil$ bits, the minimum size of $m(a)$ can be calculated from the recurrent relations:

$$\begin{aligned} msize &= k_c(0).b_c(0) \\ b_c(t-1) &= k_c(t).b_c(t) \\ k_c = 1 &\Longrightarrow b_c(t) \geq 1 \\ k_c \geq 2 &\Longrightarrow b_c(t) \geq \log_2(n_c(t) - 1) \end{aligned} \tag{28}$$

with $1 \leq t \leq K-2$

The number of messages used by the Dolev algorithm can be calculated to be

$$(3T+1)(N-3T)-1+3T(3T+1)(3T+2)\lceil \log_2(3T+2) \rceil \tag{29}$$

This figure differs slightly from the figure calculated in [4], because Dolev also counts the messages sent by a module to itself and ignores the messages sent in round 0.

In the following tables the parameters of a number of IAC-algorithms are shown for some practical values of N and T and with $K = T+1$ for all algorithms except the Dolev algorithm in which $K = 2T+3$. The applied codes in the successive rounds $0, 1, \cdots, T$ are represented by their parameters $[n_c, k_c, b_c]$.
From these figures we observe that for $T = 1$ and $N = 4$ the Pease-algorithm, the minimal-voting and

algorithm	N	T	*msize*	*#mess*	applied codes
Dolev	4	1	1	183	
Pease	4	1	1	9	[3,1,1]
MinVot	4	1	1	9	[3,1,1]
MaxCod	4	1	1	9	[3,1,1]
Dolev	6	1	1	191	
Pease	6	1	1	25	[5,1,1]
MinVot	6	1	1	15	[3,1,1]
MaxCod	6	1	6	8.3	[5,3,2]
Dolev	16	1	1	231	
Pease	16	1	1	225	[15,1,1]
MinVot	16	1	1	45	[3,1,1]
	16	1	4	30	[4,2,2]
MaxCod	16	1	52	17.3	[15,13,4]
Pease	64	1	1	3969	[63,1,1]
Dolev	64	1	1	423	
MinVot	64	1	1	189	[3,1,1]
	64	1	4	126	[4,2,2]
MaxCod	64	1	336	65.1	[63,61,6]
Dolev	7	2	1	1014	
Pease	7	2	1	156	[6,1,1][5,1,1]
MinVot	7	2	1	130	[5,1,1][5,1,1]
MaxCod	7	2	6	78	[6,2,3][5,1,3]
Pease	16	2	1	2955	[15,1,1][14,1,1]
Dolev	16	2	1	1077	
MinVot	16	2	1	355	[5,1,1][5,1,1]
MaxCod	16	2	440	28.1	[15,11,40][14,10,4]
Pease	64	2	1	24.10^4	[63,1,1][62,1,1]
Dolev	64	2	1	1413	
MinVot	64	2	1	1555	[5,1,1][5,1,1]
MaxCod	64	2	20532	72	[63,59,348] [62,58,6]
Dolev	10	3	1	3969	
Pease	10	3	1	3609	[9,1,1][8,1,1][7,1,1]
MinVot	10	3	1	2457	[7,1,1][7,1,1][7,1,1]
MaxCod	10	3	18	603	[9,3,6][8,2,3][7,1,3]
Pease	16	3	1	35715	[15,1,1][14,1,1] [13,1,1]
Dolev	16	3	1	4029	
MinVot	16	3	1	4515	[7,1,1][7,1,1][7,1,1]
	16	3	256	212	[10,4,64][10,4,16] [10,4,4]
MaxCod	16	3	2016	75	[15,9,224][14,8,28] [13,7,4]
Dolev	64	3	1	4509	
MinVot	64	3	1	20979	[7,1,1][7,1,1][7,1,1]

the maximal-coding algorithm are the same. For larger values of N or T, the maximal-coding algorithm is always the most favorable in terms of the number of messages. However if N or T become large the minimal message size becomes impractical. In some cases the subset method which requires one additional round gives better results. In those cases where large messages have to be broadcasted the use of authenticated algorithms is more economical. An other approach in that case is transferring the messages directly and applying an IAC algorithm on the cyclic redundancy check of these messages. The Dolev algorithm, although its message content is of polynomial order, only uses less messages for $T \geq 4$. For $T = 3$ and large N the subset method prevails.

6 Conclusion

This paper describes and proves a new class of algorithms from which a new class of interactive consistency algorithms based on error-correcting codes is derived. The original IAC algorithm described by Pease et al. is a member of this new class. We showed that for practical values of N and T the new algorithms require considerably less communication than the other known synchronous non-authenticated IAC-algorithms.

References

[1] M.Barborak, M.Malek, "The consensus Problem in Fault-Tolerant Computing" *ACM Computing Surveys*, Vol.25, No 2, June 1993.

[2] D.Davies, J.Wakerly, "Synchronization and matching in redundant systems",*IEEE Tr. on Comp. C-27,6* pp.531-539, June 1978.

[3] D.Dolev,"The Byzantine Generals strike again",*Journal of algorithms,*Vol.3, pp.14-30.

[4] D.Dolev, M.J.Fischer, R.Fowler, N.A.Lynch and H.R.Strong,
"An Efficient Algorithm for Byzantine Agreement without Authentation" *Information and Control* Vol.52, No.2, pp257-274, 1982.

[5] D.Dolev and H.R.Strong, "Autenticated algorithms for Byzantine agreement" *SIAM COMP*, Vol.12, No.4, pp.656-666, Nov. 1983.

[6] M.Fischer and N.Lynch, "A lower bound for the time to assure interactive consistency" *Inform. Proc. Letters,* Vol.14, pp.183-186, 1982.

[7] "The (4,2) concept fault-tolerant computer" *12-th Annual Symposium on Fault-Tolerant Computing*, pp49-54, Santa Monica, CA, June, 1982.

[8] Th.Krol, "(N,K) Concept Fault tolerance" *IEEE Tr. on Comp.*, Vol.C35, No 4, April 1986, pp339-349.

[9] Th.Krol, "A Generalization of fault-Tolerance Based on Masking", *Ph.D. dissertation*, Eindhoven University of Technology, Eindhoven, the Netherlands, Sept. 1991.

[10] F.J.MacWilliams and N.J.A.Sloane, "The Theory of Error-Correcting Codes." Amsterdam, The Netherlands: North Holland, 1978.

[11] M.Pease, R.Shostak, and L.Lamport, "Reaching agreement in the presence of faults" *J. Assoc. Comput. Mach.*, Vol.27, No.2, pp228, 1980.

[12] J.H.Wensley *et al.*, "SIFT: design and analysis of a fault-tolerant computer for aircraft control " *Proc. IEEE*, Vol. 66, pp.1240-1255, Oct. 1978.