Parametric Verification of a Group Membership Algorithm *

Ahmed Bouajjani and Agathe Merceron

LIAFA, Univ. of Paris 7, Case 7014, 2 place Jussieu, F-75251 Paris 5, France.
E-mail: {Ahmed.Bouajjani,Agathe.Merceron}@liafa.jussieu.fr

Abstract. We address the problem of verifying clique avoidance in the TTP protocol. TTP allows several stations embedded in a car to communicate. It has many mechanisms to ensure robustness to faults. In particular, it has an algorithm that allows a station to recognize itself as faulty and leave the communication. This algorithm must satisfy the crucial 'non-clique' property: it is impossible to have two or more disjoint groups of stations communicating exclusively with stations in their own group. In this paper, we propose an automatic verification method for an arbitrary number of stations N and a given number of faults k. We give a faithful abstraction that allows to model the algorithm by means of unbounded (parametric) counter automata. We have checked the non-clique property on this model in the case of one fault, using the ALV tool as well as the LASH tool.

keywords: Formal verification, fault-tolerant protocols, parametric counter automata, abstraction.

1 Introduction

The verification of complex systems, especially of software systems, requires the adoption of powerful methodologies based on combining, and sometimes iterating, several analysis techniques. A widely adopted approach consists in combining abstraction techniques with verification algorithms (e.g., model-checking, symbolic reachability analysis, see, e.g., [16, 1, 23]). In this approach, non-trivial abstraction steps are necessary to construct faithful abstract models (typically finite-state models) on which the required properties can be automatically verified. The abstraction steps can be extremely hard to carry out depending on how restricted the targeted class of abstract models is. Indeed, many aspects in the behavior of complex software systems cannot (or can hardly) be captured using finite-state models. Among these aspects, we can mention, e.g., (1) the manipulation of variables and data-structures (counters, queues, arrays, etc.) ranging over infinite domains, (2) parameterization (e.g., sizes of the data structures, the number of components in the system, the rates of errors/faults/losses,

* This work was supported in part by the European Commission (FET project ADVANCE, contract No IST-1999-29082).

etc.). For this reason, it is often needed to consider abstraction steps which yield infinite-state models corresponding to extended automata, i.e., a finite-control automata supplied with unbounded data-structures (e.g., timed automata, push-down automata, counter automata, FIFO-channel automata, finite-state transducers, etc.) [1]. Then, symbolic reachability analysis algorithms (see, e.g., [14, 7–9, 18, 12, 24, 10, 4, 2, 3]) can be applied on these (abstract) extended automata-based models in order to verify the desired properties of the original (concrete) system. Of course, abstraction steps remain non-trivial in general for complex systems, even if infinite-state extended automata are used as abstract models.

In this paper, we consider verification problems concerning a protocol used in the automotive industry. The protocol, called TTP/C, was designed at the Technical University of Vienna in order to allow the communication between several devices (micro-processors) embedded in a car, whose function is to control the safe execution of different driving actions [20, 19].

The protocol involves many mechanisms to ensure robustness to faults. In particular, the protocol involves a mechanism which allows to discard devices (called stations) which are (supposed to be) faulty. This mechanism must ensure the crucial property: *all active stations form one single group of communicating stations, i.e., it is impossible to have two (or more) disjoint groups of active stations communicating exclusively with stations in their own group.*

Actually, the algorithm is very subtle and its verification is a real challenge for formal and automatic verification methods. Roughly, it is a parameterized algorithm for N stations arranged in a ring topology. Each of the stations broadcasts a message to all stations when it is its turn to emit. The turn of each station is determined by a fixed time schedule. Stations maintain informations corresponding to their view of the global state of the system: a membership vector, consisting of an array with a parametric size N, telling which stations are active. Stations exchange their views of the system and this allows them to recognize faulty stations. Each time a station sends a message, it sends also the result of a calculation which encodes its membership vector. Stations compare their membership vectors to those received from sending stations. If a receiver disagrees with the membership vector of the sender, it counts the received message as incorrect. If a station disagree with a majority of stations (in the round since the last time the station has emitted), it considers itself as faulty and leaves the active mode (it refrains from emitting and skips its turn). Stations which are inactive can return later to the active mode (details are given in the paper). Besides the membership vector, each station s maintains two integer counters in order to count in the last round (since the previous emission of the station s) (1) the number of stations which have emitted and from which s has received a correct message with membership vector equal to its own vector at that moment (the stations may disagree later concerning some other emitting station), and (2) the number of stations from which s received an incorrect message (the incorrect message may be due to a transmission fault or to a different membership vector). The information maintained by each station s depends tightly on its position in the ring relatively to the positions of the faulty stations and relatively to the stations which agree/disagree with s w.r.t. each fault.

The proof of correction of the algorithm and its automatic verification are far from being straightforward, especially in the parametric case, i.e., for any number of stations, and any number of faults.

The first contribution of the paper is to prove that the algorithm stabilizes to a state where all membership vectors are equal after precisely two rounds from the occurrence of the last fault in any sequence of faults.

Then, we address the problem of verifying automatically the algorithm. We prove that, for every fixed number of faults k, it is possible to construct an exact abstraction of the algorithm (parameterized by the number of stations N) by means of a parametric counter automaton. This result is surprising since (1) it is not easy to abstract the information related to the topology of the system (ordering between the stations in the ring), and (2) each station (in the concrete algorithm) has local variables ranging over infinite domains (two counters and an array with parametric bounds). The difficulty is to prove that it is possible to encode the information needed by all stations by means of a finite number of counters. Basically, this is done as follows: (1) We observe that a sequence of faults induces a partition of the set of active stations (classes correspond to stations having the same membership vector) which is built by successive refinements: Initially, all stations are in the same set, and the occurrence of each fault has the effect of splitting the class containing the faulty station into two subclasses (stations which recognizes the fault, and the other ones). (2) We show that there is a partition of the ring into a finite number of regions (depending on the positions of the faulty stations) such that, to determine at any time whether a station of any class can emit, it is enough to know how many stations in the different classes/zones have emitted in the last two rounds. This counting is delicate due to the splitting of the classes after each fault.

Finally, we show that, given a counter automaton modeling the algorithm, the stabilization property (after 2 rounds following the last fault) can be expressed as a constrained reachability property (in CTL with Presburger predicates) which can be checked using symbolic reachability analysis tools for counter automata (e.g., ALV [13] or LASH [21]). We have experimented this approach in the case of one fault. We have built a model for the algorithm in the language of ALV, and we have been able to verify automatically that it converges to a single clique after precisely two rounds from the occurrence of the fault. It is worth noticing that we have provided a refinement of the abstraction given in the general case which allows to build a simpler automaton. This refinement is based on properties specific to the 1 fault case that have been checked automatically using ALV.

The paper is organized as follows. Section 2 presents the protocol. In Section 3, we prove the crucial non-clique property for n stations: the stations that are still active do have the same membership vector at the end of the second round following fault k. Considering the 1 fault case, section 4 presents how to abstract faithfully the protocol parameterized by the number of stations n as an automaton with counters that can be symbolically model checked. Section 5 generalizes the approach for a given number of faults k. Section 6 concludes the paper.

2 Informal description of the protocol

TTP is a time-triggered protocol. It has a finite set S of N stations and allows them to communicate via a shared bus. Messages are broadcast to all stations via the bus. Each station that participates in the communication sends a message when it is the *right* time to do so. Therefore, access to the bus is determined by a time division multiple access (TDMA) schema controlled by the global time generated by the protocol. A TDMA round is divided into *time slots*. The stations are statically ordered in a ring and time slots are allocated to the stations according to their order. During its time slot, a station has exclusive message sending rights. A TDMA round for three stations is shown in Figure 1. When one round is completed, a next one takes place following the same pattern.

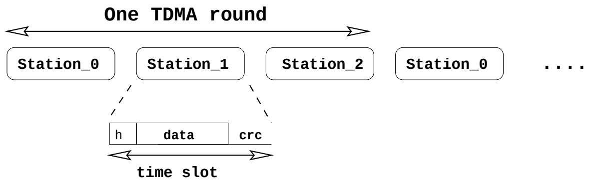

Fig. 1. A TDMA round for 3 stations.

TTP is a fault-tolerant protocol. Stations may fail while other stations continue communicating with each other. TTP provides different services to ensure robustness to faults, such as replication of stations, replication of communication channels, bus guardian, fault-tolerant clock synchronization algorithm, implicit acknowledgment, clique avoidance mechanism, [20, 19, 5]. Several classes of faults are distinguished. A symmetric fault occurs when a station is *send faulty*, i.e., no other station can receive it properly, or *receive faulty*, i.e., it cannot receive properly any message. Asymmetric faults occur when an emitting station is received properly by more than 1 station, but less then all stations. In this paper, we allow asymmetric faults to occur and consider symmetric faults as a special case of asymmetric faults. TTP provides special mechanisms to deal with other failures like processor faults, transient faults, but we do not consider them here. For the protocol to work well, it is essential that (asymmetric) faults do not give rise to cliques. In [20, 19] *cliques* are understood as disjoint sets of stations communicating exclusively with each other. In this paper, we focus on implicit acknowledgment and clique avoidance mechanism, to be introduced shortly, and show that they prevent the formation of different cliques, clique is cast in its graph theoretical meaning.

When it is working or in the `active` state, a station sends messages in its time slot, listens to messages broadcast by other stations and carries local calculations.

2.1 Local information

Each station s stores locally some information, in particular a *membership vector* m_s and two counters, $CAcc_s$ and $CFail_s$. A *membership vector* is an array of booleans indexed by S. It indicates the stations that s receives correctly (in a sense that will be made precise below). If s received correctly the last message, also called *frame*, sent by s', then $m_s[s'] = 1$, otherwise $m_s[s'] = 0$. A sending station is supposed to receive herself properly, thus $m_s[s] = 1$ for a working station s. The counters $CAcc_s$ and $CFail_s$ are used as follows. When it is ready to send, s resets $CAcc_s$ and $CFail_s$ to 0. During the subsequent round, s increases $CAcc_s$ by 1 each time it receives a correct frame (this includes the frame it is sending itself) and it increases $CFail_s$ by 1 each time it receives an incorrect frame. When no frame is sent (because the station that should send is not working), neither $CFail_s$ nor $CAcc_s$ are increased.

2.2 Implicit acknowledgment

Frames are broadcast over the bus to all stations but they are not explicitly acknowledged. TTP has *implicit acknowledgment*.

A frame is composed of a header, denoted by `h` in Figure 1, a data field, denoted by `data` and a CRC field denoted by `crc`. The data field contains the data, like sensor-recorded data, that a station wants to send to all other stations. The CRC field contains the calculation of the Cyclic Redundancy Check done by the sending station. CRC is calculated over the header, the data field and the individual membership vector. When station s is sending, it puts in the CRC field the calculation it has done with its own membership vector m_s. Station s' receiving a frame from station s recognizes the frame as *valid* if all the fields have the expected lengths. If the frame is valid, station s' performs a CRC calculation over the header and the data field it has just received, and its own membership vector $m_{s'}$. It recognizes the frame as *correct* if it has recognized it as valid and its CRC calculation agrees with the one put by s in the CRC field. Therefore, a correct CRC implies that sender s and receiver s' have the same membership vector. *We also assume a converse:* if s and s' do not have the same membership vector, the CRC is not correct. The CRC check justifies *implicit acknowledgment*. Receiver s' has $m_{s'}[s'] = 1$. A correct CRC implies $m_{s'} = m_s$, thus $m_s[s'] = 1$. Hence s has correctly received the last frame sent by s', i.e., s' is implicitly acknowledged by s. Implicit acknowledgment in TTP contains an additional feature involving first and second successors [19]. Our result on cliques is established for a version where these complications are not present.

2.3 Clique avoidance mechanism

The *clique avoidance mechanism* reads as follows: Once per round, at the beginning of its time slot, a station s checks whether $CAcc_s > CFail_s$. If it is the case, it resets both counters as already said above and sends a message. If it is not the case, the stations fails. It puts its own membership vector bit to 0, i.e., $m_s[s] = 0$, and leaves the `active` state, thus will not send in the subsequent rounds. The intuition behind this mechanism is that a station that fails to recognize a majority of frames as correct, is most probably not working properly. Other working stations, not receiving anything during the time slot of s, put the bit of s to 0 in their own membership vector.

2.4 Reintegration

Faulty stations that have left the active state can re-integrate the active state [19, 5]. The integration algorithm works as follows. An integrating station s copies the membership vector from some active station. As soon as the integrating station has a copy, it updates its membership vector listening to the traffic following the same algorithm as other working stations. During its first sending slot, it resets both counters, $CAcc_s$ and $CFail_s$ to 0, without sending any frame. During the following round, it increases its counters and keeps updating its membership vector as working stations do. At the beginning of its next sending slot, s checks whether $CAcc_s > CFail_s$. If it is the case, it puts $m_s[s]$ to 1 and sends a frame, otherwise it leaves the `active` state again. Receiving stations, if they detect a valid frame, put the membership of s to 1 and then perform the CRC check as described above.

2.5 Example

Consider a set S composed of 4 stations and suppose that all stations received correct frames from each other for a while. This means that they all have identical membership vectors and $CFail = 0$. After station s_3 has sent, the membership vectors as well as counters $CAcc$ and $CFail$ look as follows. Remember that there is no global resetting of $CAcc$ and $CFail$. Resetting is relative to the position of the sending station.

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	$CAcc$	$CFail$
s_0	1	1	1	1	4	0
s_1	1	1	1	1	3	0
s_2	1	1	1	1	2	0
s_3	1	1	1	1	1	0

We suppose that a fault occurs while s_0 is sending and that no subsequent fault occurs for at least two rounds, calculated from the time slot of s_0. We assume also that the frame sent by s_0 is recognized as correct by s_2 only. So the set S is split in two subsets, $S_1 = \{s_0, s_2\}$ and $S_0 = \{s_1, s_3\}$.

1. Membership vectors and counters after s_0 has sent:

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	$CAcc$	$CFail$
s_0	1	1	1	1	1	0
s_1	0	1	1	1	3	1
s_2	1	1	1	1	3	0
s_3	0	1	1	1	1	1

2. Membership vectors and counters after s_1 has sent. Notice that, because s_0 and s_2 do not have the same membership vector as s_1, the CRC check does not pass, so they don't recognize the frame sent by s_1 as correct.

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	*CAcc*	*CFail*
s_0	1	0	1	1	1	1
s_1	0	1	1	1	1	0
s_2	1	0	1	1	3	1
s_3	0	1	1	1	2	1

3. Membership vectors and counters after s_2 has sent:

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	*CAcc*	*CFail*
s_0	1	0	1	1	2	1
s_1	0	1	0	1	1	1
s_2	1	0	1	1	1	0
s_3	0	1	0	1	2	2

4. Memberships and counters after the time slot of s_3, which cannot send by the clique avoidance mechanism and leaves the `active` state:

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	*CAcc*	*CFail*
s_0	1	0	1	0	2	1
s_1	0	1	0	0	1	1
s_2	1	0	1	0	1	0
s_3	0	0	0	0	0	0

5. Memberships and counters after s_0 has sent again:

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	*CAcc*	*CFail*
s_0	1	0	1	0	1	0
s_1	0	1	0	0	1	2
s_2	1	0	1	0	2	0
s_3	0	0	0	0	0	0

6. Memberships and counters after the time slot of s_1, which cannot send by the clique avoidance mechanism and leaves the `active` state:

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	*CAcc*	*CFail*
s_0	1	0	1	0	1	0
s_1	0	0	0	0	0	0
s_2	1	0	1	0	2	0
s_3	0	0	0	0	0	0

Membership vectors are coherent again at this point of time.

3 Proving Clique Avoidance

In this section, we prove that if k faults occur at a rate of more than 1 fault per two TDMA rounds and if no fault occur during two rounds following fault k, then at the end of that second round, all active stations have the same membership vector, so they form a single *clique* in the graph theoretical sense.

Let us denote by W the subset of S that contains all working stations. We may write $m_s = S'$ for a station s with $S' \subseteq S$ as a short hand for $m_s[s'] = 1$ iff $s' \in S'$. To prove coherence of membership vectors we start with the following situation. We suppose that stations of W have identical membership vectors and all have $CFail_s = 0$. Because $m_s[s] = 1$ for any working station, this implies that $m_s = W$ for any $s \in W$. Faults occur from this initial state.

Let us define a graph as follows : the nodes are the stations, and there is an arc between s and s' iff $m_s[s'] = 1$. We recall that, in graph theory, a *clique* is a complete subgraph, i.e., each pair of nodes is related by an arc. Thus initially, W forms a single clique in the graph theoretical sense.

3.1 Introductory example

Let us illustrate how things might work in the case of two faults. The first fault occurs when s_0 sends. We suppose that only s_1 fails to receive correctly the frame sent by s_0. S is split as $S_1 = \{s_0, s_2, s_3\}$ and $S_0 = \{s_1\}$. Membership vectors and counters after s_0 has sent:

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	*CAcc*	*CFail*
s_0	1	1	1	1	1	0
s_1	0	1	1	1	3	1
s_2	1	1	1	1	3	0
s_3	1	1	1	1	2	0

Membership vectors and counters after s_1 has sent:

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	*CAcc*	*CFail*
s_0	1	0	1	1	1	1
s_1	0	1	1	1	1	0
s_2	1	0	1	1	3	1
s_3	1	0	1	1	2	1

Membership vectors and counters after s_2 has sent. At this point, we suppose that a second fault occurs. Neither s_3 nor s_0 recognize the frame sent by s_2 as correct. S_1 is split in $S_{11} = \{s_2\}$ and $S_{10} = \{s_0, s_3\}$:

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	*CAcc*	*CFail*
s_0	1	0	0	1	1	2
s_1	0	1	0	1	1	1
s_2	1	0	1	1	1	0
s_3	1	0	0	1	2	2

Membership vectors and counters after the time slot of s_3, which is prevented from sending by the clique avoidance mechanism:

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	*CAcc*	*CFail*
s_0	1	0	0	0	1	2
s_1	0	1	0	0	1	1
s_2	1	0	1	0	1	0
s_3	0	0	0	0	0	0

One notices that s_0, then s_1 are prevented from sending by the clique avoidance mechanism. Membership vectors and counters after the time slot of s_1:

stations	$m[s_0]$	$m[s_1]$	$m[s_2]$	$m[s_3]$	*CAcc*	*CFail*
s_0	0	0	0	0	0	0
s_1	0	0	0	0	0	0
s_2	0	0	1	0	1	0
s_3	0	0	0	0	0	0

Coherence is achieved again after the time slot of s_1, where s_2 remains the only active station. Though S_{11} is smaller that S_{10}, the position of s_2 in the ring as the first station of the round with the second fault allows it to capitalize on frames accepted in the round before and to win over the set S_{10}.

3.2 Proving a single clique after k faults

The proof proceeds as follows. First we show a preliminary result. If W is divided into subsets S_i is such a way that all stations in a subset have the same membership vector, then stations inside a subset behave similarly : if there is no fault, they recognize the same frames as correct or as incorrect. Then we show that the occurrence of faults does produce such a partitioning, i.e., after fault k, W is divided into subsets S_w, where $w \in \{0,1\}^k$. Indeed, as illustrated by the example at the end of section 2, after 1 fault, W is split in S_1, the stations that recognize the frame as correct, and S_0, the stations that do not recognize the frame as correct. Because any station recognizes itself as correct, S_1 is not empty. Now, suppose that a second fault occurs. Assume that the second fault occurs when a station from set S_1 sends. As before, set S_1 splits into S_{11}, the stations that recognize the frame as correct, and S_{10}, the stations that do not recognize the frame as correct, as illustrated in the example in Subsection 3.1. Again S_{11} is not empty. And the process generalizes. If a station s from a set S_w sends when fault k occurs, S_w splits into S_{w1} and S_{w0} with $S_{w1} \neq \emptyset$. Then, we show that two stations s and s' have the same membership vector if and only if they belong to the same set S_w. Using the preliminary lemma, we have a result about the incrementation of the counters *CAcc* and *CFail*, namely, all stations from a set S_w increment *CAcc* if a station from S_w sends, and increment *CFail* if a station from $S_{w'}$ sends, where $w \neq w'$. From this, we can deduce our main result: in the second round after fault k, only stations from a single set S_w can send. It follows that, at the end of the second round, there is only one clique and it is not empty.

The preliminary result reads as follows.

Lemma 1 *Suppose that W is divided into m subsets $S_1, \ldots, S_m$ such that s and s' have the same membership vector iff s and s' belong to the same subset S_i, $1 \leq i \leq m$. Let $s \in S_i$, $1 \leq i \leq m$. Suppose no fault occurs. Then, each time some other station s' from S_i is sending, s increases $CAcc_s$ by 1 and keeps the membership bit of s' to 1. Each time some station $s' \in S_j$, $j \neq i$ is sending, s increases $CFail_s$ by 1 and puts the membership bit of s' to 0.*

Proof. We prove this lemma for the general case where integrating stations are allowed.
First notice that, for $s, s' \in S_i$ with $s \neq s'$, $m_s[s'] = 1$ and $m_{s'}[s] = 1$. Indeed, $s, s' \in W$ mean $m_s[s] = 1$ and $m_{s'}[s'] = 1$. Because s and s' have the same membership vector, one has $m_s[s'] = 1$ and $m_{s'}[s] = 1$.

Let $s \in S_i$ and s' be the station which is sending.
First suppose $s' \in S_i$ or, if s' is an integrating station, s' has copied its membership vector from some active station $s'' \in S_i$. Following the integration policy, both s and s' put the membership bit of s' to 1 when s' sends. Because s and s' have the same membership vector and because no other fault occurs, s recognizes as correct the frame sent by s'. So it increases CAcc_s by 1 when s' sends and keeps the membership bit of s' to 1.
Suppose now $s' \in S_j$, $j \neq i$. As above, if s' is an integrating station, s' has copied its membership vector from some active station $s'' \in S_j$. Because s and s' do not have the same membership vector, s does not recognize the frame sent by s' as correct. Hence, it increases CFail_s by 1 when s' sends and puts the membership bit of s' to 0. ■

Now we show that faults partition the set W.

Proposition 2 *At the end of the time slot of s^k, the station where fault k occurs, $k \geq 1$, W is partitioned into subsets S_w, with $w \in \{0, 1\}^k$, such that:*

1. *there exists at least one w with $S_w \neq \emptyset$,*
2. *any two stations $s \in S_w$ and $s' \in S_{w'}$ have the same membership vector iff $w = w'$,*
3. *for any $w \in \{0 \mid 1\}^k$ with $S_w \neq \emptyset$, for any $s, s' \in S_w$, $m_s[s'] = 1$.*

Proof. We proceed by induction on k.
Basis : $k = 1$. Let s^1 be the station which is sending when the first fault occurs. Before the time slot of s^1, all stations have the same membership vector, namely W, by hypothesis on the initial state. We denote by S_0 the subset of stations that *failed* to receive correctly the frame sent by s^1 while we denote by S_1 the subset of stations that *accepted* the frame as correct.
Because the sending station s^1 is considered as receiving itself properly, we have $S_1 \neq \emptyset$. This proves the first item.
All stations in S_1 could receive correctly the frame sent by s^1, consequently their membership vector does not change. None of the station in S_0 could receive correctly the frame sent by s^1 because of the fault. Hence they all put the membership bit of s^1 to *false*. Thus they all have the same membership vector, namely $W \setminus \{s^1\}$. This proves the second and third item, since, before the fault, $m_s = W$ for any $s \in W$.
Suppose the result is true till fault $k - 1$.
Induction step : we prove that the result is true for k, $k > 1$. Let s^k be the station which is sending when fault k occurs. By induction hypothesis, before the time slot of s^k, W is partitioned into sets S_w with $w \in \{0|1\}^{k-1}$ and at least one subset S_w is not empty. Exists a unique $w \in \{0|1\}^{k-1}$ with $s^k \in S_w$. By induction hypothesis $m_s[s^k] = 1$ for any station $s \in S_w$. Let us denote by S_{w1}

the subset of S_w that could receive s^k correctly and by S_{w0} the subset of S_w that could not receive s^k correctly. $s^k \in S_{w1}$, thus $S_{w1} \neq \emptyset$.
Obviously, S_{w1} and S_{w0} partition S_w. After the time slot of s^k, all stations in S_{w1} keep the membership of s^k to 1, while all stations in S_{w0} put it to 0. Using the induction hypothesis, this proves second and third item for S_{w1} and S_{w0}.
Consider some $w' \in \{0|1\}^{k-1}$, with $w' \neq w$ and $S_{w'} \neq \emptyset$. Let $s' \in S_{w'}$. By induction hypothesis s^k and s' do not have the same membership vector. So, when it performs the CRC check, s' does not recognize as correct the frame sent by s^k. Thus $S_{w'}$ can be denoted by $S_{w'0}$. Using the induction hypothesis, it is easy to deduce that the second and third item hold for $w \in \{0|1\}^k$. ■

Now, observe that in the second round following fault k, the last fault, at least one station can send.

Lemma 3 *In the second round following fault k, at least one station is sending.*

Proof. We know that at least one station has sent in the first round following fault k (that station was possibly station s^k). Let s be the last station which is sending in that round. Because s recognizes itself as correct, after sending one has $CAcc_s = 1$ and $CFail_s$ has been reset to 0. If no other station sends before it, station s is certainly able to send in the second round since it has $CAcc_s = 1 > 0 = CFail_s$. ■

Finally, we show that only stations from a unique set S_w are able to send in the second round following fault k.

Theorem 1. *Let $s \in S_w$ be the first station to send in the second round following fault k. Then, only stations from set S_w can send in the second round following fault k.*

Proof. Let $CAcc_s$ and $CFail_s$ when s performs the clique avoidance mechanism. By Lemma 1, $CAcc_s = \mid \{s' \in S_w \mid s' \text{ sent in the first round following fault } k\} \mid$ and $CFail_s = \Sigma_{w' \neq w} \mid \{s' \in S_{w'} \mid s < s', \text{ and } s' \text{ sent in the first round following fault } k\} \mid$, where $<$ refers to the statical order among stations. Because s is able to send, one has $CAcc_s > CFail_s$ at the beginning of the time slot of s in the second round.
Let t be the follower station of s in the statical order that have sent in the first round following fault k. Is t able to send in the second round, or is it prevented from sending by the clique avoidance mechanism? We show that t is able to send in the second round if and only if it belongs to S_w.
Suppose $t \in S_w$. When its time slot comes in the second round, it has increased $CAcc_t$ by 1 when s has sent in the second round. Because, in the first round following fault k, t increases its counters as s does by Lemma 1, one has $CAcc_t = CAcc_s$ and $CFail_t = CFail_s$ at the beginning of the time slot of t in the second round. Thus t is able to send as well.
Suppose now $t \in S_{w'}$ for some $w' \neq w$. $CAcc_t = \mid \{s' \in S_{w'} \mid t \leq s' \leq s^k \text{ and } s' \text{ sent in the first round following fault } k\} \mid$. However, by Lemma 1, $CAcc_t \leq CFail_s$. Furthermore, also by Lemma 1, $CAcc_s \leq CFail_t$. It follows that, at the beginning of its time slot, $CAcc_t < CFail_t$ and t is prevented from sending.

A similar argument can be repeated to all stations that have sent in the first round following fault k giving the result. ■

From Theorem 1, one deduces that, at the end of the second round following fault k, for any station $s \in S_w$: $m_s = S_w$. This gives our safety property about cliques.

Corollary 4 *At the end of the second round following fault k, all working stations form a single clique in the graph theoretical sense.*

By Lemma 3, we know that the clique formed at the end of the second round is not empty. This implies that faults never prevent all stations from sending.

Corollary 5 *At the end of the second round following fault k, the set of working stations is not empty.*

3.3 Integrating stations

Proposition 2 and Theorem 1 can be generalized to include integrating stations. Indeed, an integrating station s copies the membership vector from some active station s' which belongs to some S_w and updates it as active stations do. So it keeps having the same membership vector as stations of some set $S_{w'}$, with w being a prefix of w', as faults occur. This is shown in Proposition 6, which is a sharper version of Proposition 2.

Proposition 6 *At the end of the time slot of s^k, the station where fault k occurs, $k \geq 1$, W is partitioned into subsets S_w, with $w \in \{0,1\}^k$, such that:*

1. *there exists at least one w with $S_w \neq \emptyset$,*
2. *any two stations $s \in S_w$ and $s' \in S_{w'}$ have the same membership vector iff $w = w'$,*
3. *for any $w \in \{0 \mid 1\}^k$ with $S_w \neq \emptyset$, for any $s, s' \in S_w$, $m_s[s'] = 1$.*
4. *This partition stays stable till the occurrence of fault $k+1$.*
5. *The station s^{k+1} ready to send when fault $k+1$ occurs belongs to some set S_w, $w \in \{0,1\}^k$, or has its membership vector identical to a station of some set S_w, $w \in \{0,1\}^k$.*

Proof. We proceed by induction on k.
Basis : $k = 1$. Let s^1 be the station which is sending when the first fault occurs. Before the time slot of s^1, all active stations, plus s^1 if s^1 is an integrating station, have the same membership vector, namely W, by hypothesis on the initial state. In case s^1 is an integrating station, all stations put the membership bit of s^1 to 1. We denote by S_0 the subset of W that *failed* to receive correctly the frame sent by s^1 while we denote by S_1 the subset of stations that *accepted* the frame as correct.
Because the sending station s^1 is considered as receiving itself properly, we have $S_1 \neq \emptyset$. This proves the first item.

All stations in S_1 could receive correctly the frame sent by s^1, consequently their membership vector does not change. None of the station in S_0 could receive correctly the frame sent by s^1 because of the fault. Hence they all put the membership bit of s^1 to *false*, also in the case where s^1 was an integrating station. Thus they all have the same membership vector, namely $W \setminus \{s^1\}$. This proves the second and third item.
As long as no fault occurs, stations in S_1 keep the membership bit of s' to 1 if the emitting station s' belongs to S_1 by Lemma 1. This also applies in case s' is an integrating station which has copied its membership vector from a station of S_1, or in case s' has copied its membership vector before the fault, but has recognized s^1 as correct, i.e., behaves as stations of S^1 do. Stations in S_1 put also the membership bit of s' to 0 if the emitting station s' belongs to S_0. This also applies in case s' is an integrating station which has copied its membership vector from a station of S_0, or behaves as stations from S_0 do. This proves the item 4 concerning the stability of the partition.
Consider the station s^2 ready to send when the next fault, fault 2, occurs. If $s^2 \in W$, then $s^2 \in S^1$ or $s^2 \in S_0$. Suppose s^2 is an integrating node. If s^2 has copied its membership vector from S_1 or from S_0, then we are done. Otherwise, s^2 has copied W for its membership vector. Because s^2 updates its membership vector as working stations do, after fault 1 it has updated its membership vector and got a membership vector identical to S_1, or to S_0, depending on its reaction to the fault. By Lemma 1, its membership vector remains identical to the membership vector of stations in S_1, or to the membership vector stations in S_0.
Suppose the result is true till fault $k-1$.
Induction step : we prove that the result is true for k, $k > 1$. Let s^k be the station which is sending when fault k occurs. By induction hypothesis, at the end the time slot of s^{k-1}, W is partitioned into sets S_w with $w \in \{0|1\}^{k-1}$ and at least one subset S_w is not empty. Exists a unique $w \in \{0|1\}^{k-1}$ with $s^k \in S_w$ or such that the membership vector of s^k is identical to S_w.

By induction hypothesis $m_s[s^k] = 1$ for any station $s \in S_w$. Let us denote by S_{w1} the subset of S_w that could receive s^k correctly and by S_{w0} the subset of S_w that could not receive s^k correctly. $s^k \in S_{w1}$, thus $S_{w1} \neq \emptyset$.
Obviously, S_{w1} and S_{w0} partition S_w. After the time slot of s^k, all stations in S_{w1} keep the membership of s^k to 1, while all stations in S_{w0} put it to 0. Using the induction hypothesis, this proves second and third item for S_{w1} and S_{w0}.
Consider some $w' \in \{0|1\}^{k-1}$, with $w' \neq w$ and $S_{w'} \neq \emptyset$. Let $s' \in S_{w'}$. By induction hypothesis s^k and s' do not have the same membership vector. So, when it performs the CRC check, s' does not recognize as correct the frame sent by s^k. Thus $S_{w'}$ can be denoted by $S_{w'0}$. Using the induction hypothesis, it is easy to deduce that the second and third item hold for $w \in \{0|1\}^k$.
Items four and five are proved in a similar way as has been done for the case 1.
■

At its first time slot, an integrating station s resets its counters but does not send. During the subsequent round, it increments $CAcc_s$ and $CFail_s$ as working stations do.When its time slot comes again, s, having the same membership vector as stations of S_w, performs the clique avoidance mechanism. $CAcc_s > CFail_s$ means that stations from S_w, without s, have emitted more than stations from any other set. Thus s can emit and, that way, contributes to reinforce S_w. If $CAcc_s \leq CFail_s$, then s cannot emit and nothing will be changed concerning the size of S_w.

Theorem 2. *Only stations from a unique set S_w can send in the second round following fault k.*

Proof. Let s be the first station that emits in the second round following fault k. Such a station exists by Lemma 3.
If s is a station that emitted in the first round following fault k, the beginning of proof 1 can be repeated.
Suppose s is an integrating station. s has a copy of the membership vector of some active station. Let us assume that s has the membership vector of some station from S_w. s is the first station to send in the second round, this means that, in the statical order, s comes before any other active station from S_w. Consider $CAcc_s$ and $CFail_s$ when s performs the clique avoidance mechanism. $CAcc_s = \mid \{s' \in S_w \mid s' \text{ sent in the first round following fault } k\} \mid$ because $CAcc_s$ has been put to 0 in the last round. Also $CFail_s = \Sigma_{w' \neq w} \mid \{s' \in S_{w'} \mid s < s'$, and s' sent in the round following fault $k\} \mid$. One knows $CAcc_s > CFail_s$, since s is able to emit.
Consider t, the first station following s in the statical order, that is either active or integrating. If t is integrating, it has copied its membership vector from some active station. Suppose either $t \in S_w$ or t has copied its membership vector from some active station of S_w. Because t increases its counters as s does by Lemma 1, $CAcc_t = CAcc_s + 1$, if s was an integrating node – because s did not send in the previous round–, or $CAcc_t = CAcc_s$ otherwise. Furthermore, $CFail_t = CFail_s$, thus t is able to send as well.
Suppose $t \in S_{w'}$ or t has copied its membership vector from some station of $S_{w'}$, $w' \neq w$. A reasoning similar to the one used in the proof 1 holds showing that t is prevented from sending in the second round.
Repeating the argument for follower stations gives the result. ■

Corollaries 4 and 5 hold as well.

4 Automatic verification: the 1 fault case

In the case of a single fault, the set W of active stations is divided into two subsets, S_1 and S_0. The set S_1 is not empty as it contains s^1, the station that was sending when the fault occurs. We assume that no other fault occurs for the next two rounds, a round is taken with the beginning of the time slot of s^1. We want to prove automatically for an arbitrary number N of stations that, at the end of the second round following the fault, all working stations form a single non-empty clique. To achieve this goal, we need a formalism to model the protocol and a formalism to specify the properties that the protocol must satisfy.

To model the protocol, we take an automaton with parameters and counters. To specify the properties, we take the temporal logic CTL.

We have seen that each station maintains two counters, *CAcc* and *CFail* and a membership vector m. To be able to model the protocol with an arbitrary number N of stations by an automaton, we need an abstraction that allows us to forget all individual membership vectors and counters, and, at the same time, allows us to represent correctly the emission of frames.

We divide the presentation in two main parts: first round, and second and later rounds following the fault.

4.1 First round following the fault: abstraction

In the case of 1 fault, Propositions and Lemmata of section 3 take a simpler form given by the Corollaries below. Essentially, stations from S_1 increment *CAcc* when stations from S_1 send, and they increment *CFail* and do $m[s] = 0$ when some station s from S_0 sends. Stations from S_0 behave similarly.

Corollary 7 *At the end of the time slot of s^1, all stations in S_1 have the same membership vector, namely W and all stations in S_0 have the same membership vector, namely $W \setminus \{s^1\}$.*

Corollary 8 *In the round following the time slot of s^1, after a station s of S_0 has sent, any station s' of S_1 which is still in the* `active` *state, puts the membership bit of s to* false*, i.e. $m_{s'}[s] = 0$, and increases $CFail_{s'}$ by 1.*

Corollary 9 *In the round following the time slot of s^1, after a station s of S_1 has sent, any station s' of S_1, which is still in the* `active` *state, keeps the membership bit of s to* true *and increases $CAcc_{s'}$ by 1.*

With these corollaries, it is equivalent to know which set, S_1 or S_0, a station belongs to, or to know its membership vector. Hence, we may abstract away individual membership vectors and keep only the two sets.

Let us see now that we can abstract away all individual counters *CAcc* and *CFail*. Let $d1$ be a counter to count how many stations of S_1 have sent so far in the round since the fault occurred. Let $d0$ be a similar counter for S_0. These two global counters are enough to calculate $CAcc_s$ and $CFail_s$ for each station s ready to send. Indeed, let s be a station ready to send. Assume $s \in S_1$. How much is $CFail_s$? It is exactly given by $d0$. How much is $CAcc_s$? Generally, it is more than $d1$. One has to add all stations that have emitted before the fault since the last time slot of s, because s has recognized them all as correct, see Figure 2. However, this number can be calculated exactly with the help of $d1$ and $d0$ only as Proposition 3 shows.

Theorem 3. *Let s a station ready to send in the round following fault 1.*

1. *If $s \in S_1$, then $CAcc_s = |W| - d0$ and $CFail_s = d0$.*
2. *If $s \in S_0$, then $CAcc_s = |W| - d1$ and $CFail_s = d1$.*

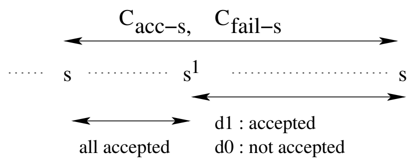

Fig. 2. Illustrating proof of Theorem 3.

Proof. We prove the first item only, the second one being dual. Since there was no fault in the last round, s has recognized as correct all stations between itself and s^1 in the last round. By Corollaries 7 and 9, it has recognized as correct all stations of S_1 that have sent so far in the round. This is illustrated in Figure 2. Thus $CAcc_s = \mid \{s' \mid s \leq s' < s^1\} \cap W \mid + d1$, where $\leq$ and $<$ refer the statical order among stations. $\{s' \mid s \leq s' < s^1\} \cap W = W \setminus (\{s' \mid s^1 \leq s' < s\} \cap W)$. Thus $\mid \{s' \mid s \leq s' < s^1\} \cap W \mid = \mid W \mid - d1 - d0$. This gives $CAcc_s = \mid W \mid - d0$. By Corollary 8, $CFail_s = d0$. ■

Corollaries 7 to 9 and Theorem 3 give us a powerful abstraction: we abstract away the N individual membership vectors, we abstract away the statical order and the individual *CAcc* and *CFail*. Instead we fix two sets, S_1 and S_2, and two counters $d1$ and $d0$, and we can model correctly stations sending and failing.

4.2 First round following the fault: the model

The protocol with N stations including the first round following 1 fault is modeled by the automaton with parameters and counters shown in the `TOP` part of Figure 3. Let us read it from left to right.

Transition from initial state `init` to state `normal` initializes counters. The counter of working stations, CW, takes as initial value the parameter N which is the total number of stations. The counter of stations that are not working anymore, or have failed, CF is set to 0 as well as three other counters, Cp, d and dF. Cp counts how many time slots in total have elapsed in the round. d counts how many working stations have sent in the round. dF counts how many slots of non-working stations have elapsed in the round.

State `normal` models the protocol when there are no faults. Transition with guard $d < CW$ models the sending of a frame by a working station while transition with guard $dF < CF$ corresponds to the time slot of a non-working station, where no frame is sent. Notice the non-determinism as we abstract the statical order among stations. Transition with guard $Cp = N$ is the starting of a new round.

Transition with guard *1fault* from `normal` to `round1` is taken when a fault occurs. The input *1fault* can be seen as a signal that can have the value *true* at random. This transition sets the items defined in the abstraction: $C1$ is the number of stations belonging to S_1, i.e., the number of stations that recognize the frame of the faulty station as correct and $C0$ is similar for S_0. These parameters must fulfill 3 constraints as given in the abstraction: $C1 \geq 1$, as the faulty

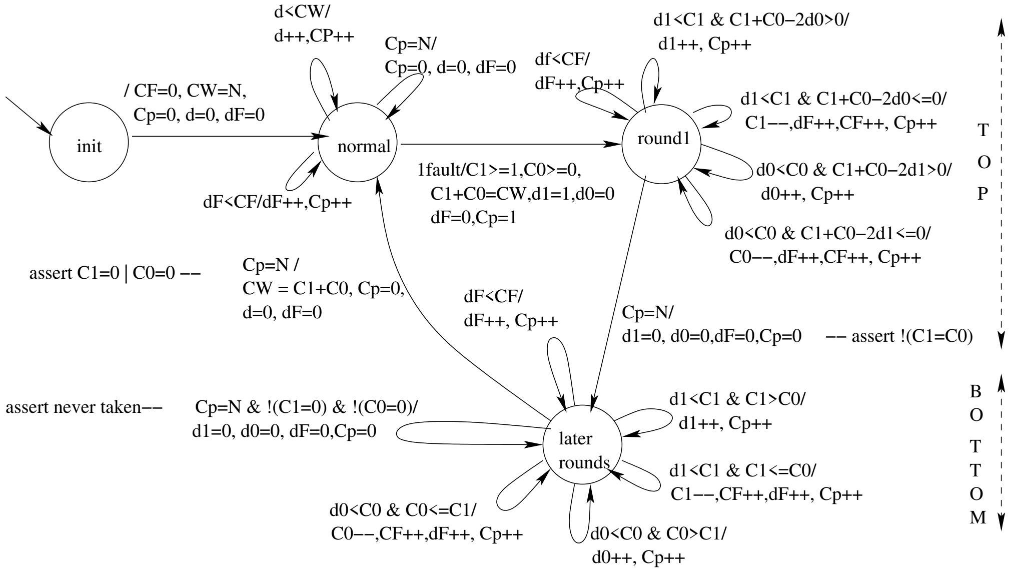

Fig. 3. The automaton with parameters and counters modeling N stations.

station recognizes itself as correct, $C0 \geq 0$ and $C1 + C0 = CW$. The counter $d1$, to count how many stations of the set S_1 have sent in the round, is set to 1 and its counterpart $d0$, for S_0, is set to 0. Finally, Cp is reset to 1 as the round is now taken from the slot of the faulty station.

Transitions of state `round1` model the sending of frames, or the failing of stations by the clique avoidance mechanism. For instance, transition with guard $d1 < C1$ & $CW - 2d0 > 0$ models a station of the set S_1 which passes the clique avoidance mechanism and sends, while $d1 < C1$ & $CW - 2d0 \leq 0$ models a station from set S_1 which leaves the `active` state because of the clique avoidance mechanism. These guards make use of Theorem 3 to check whether $CAcc_s - CFail_s > 0$. Notice again the non-determinism. State `round1` has one outgoing transition with guard $Cp = N$ to state `later rounds`. The guard is true when the first round is completed. This transition is an with an assertion that we want to prove.

4.3 First round following the fault: properties

A first property, called $P1$, that has been proved as true, is that whenever control leaves state `round1`:

$$!(C1 = C0) \qquad\qquad (P1).$$

P1 means that, when the first round after the fault is over, either $|S_1|>|S_0|$ or $|S_0|>|S_1|$, whatever the original partition $\{S_1, S_0\}$ was when the fault occurred.

We have analyzed what leads to $C1 > C0$, or $C0 > C1$ upon leaving `round1`.

First, we have shown that, if $\mid S_1 \mid > \mid S_0 \mid$ when the fault occurs, then $C1 > C0$ when control leaves `round1`, and vice-versa. Let us denote by InS_1, InS_0, the initial number of stations in the set S_1, respectively in the set S_0, when a fault occurs. Adding the constraint $InS_1 > InS_0$, we have proved that whenever control leaves state `round1`:

$$(InS_1 = C1) \qquad\qquad (P2).$$

Since counters $C1$ and $C0$ may not increase, this implies $C1 > C0$ when control enters state `later rounds`. It also implies that all stations from S_1 did send in the first round.

Then we have investigated the case $\mid S_1 \mid = \mid S_0 \mid$ when the fault occurs. If set S_1 comes first in the statical order, then $C_1 > C_0$, and vice versa if S_0 comes first. Adding the constraint $InS_1 = InS_0$ we have proved:

$$AG\ (d1 = InS_1 \texttt{ and } d0 < InS_1)\ \Rightarrow\ AG(C1 = InS_1)) \qquad (P3),$$

$$AG\ ((d1 = InS_1 \texttt{ and } d0 < InS_1)\ \Rightarrow\ (C1 + C0 - 2 * d1 <= 0)) \qquad (P4).$$

Again, this implies $C1 > C0$ when control leaves state `round1`. It also implies that all stations from S_1 did send in the first round.

4.4 Second and later rounds following the fault: abstraction

From Theorem 3, we deduce $CAcc_{s^1}$ and $CFail_{s^1}$ for the faulty station s^1 at the end of the first round.

Corollary 10 *At the end of the first round $CAcc_{s^1} = \mid S_1 \mid$ and $CFail_{s^1} = \mid S_0 \mid$.*

From Corollaries 8 and 9, we deduce that at the end of the first round, all stations in S_1 have the same membership vector, namely S_1. A similar result holds for stations in S_0.

Corollary 11 *Let s be any station and consider m_s at the end of the first round following the fault.*

1. *If $s \in S_1$ then $m_s[s'] = 1$ iff $s' \in S_1$.*
2. *If $s \in S_0$ then $m_s[s'] = 1$ iff $s' \in S_0$.*

Let us notice that S_1 and S_0 form two cliques in the graph theoretical sense. If none of this set is empty, at the end of the first round, stations are split in two cliques. This is illustrated at the end of section 2, after the time slot of s_3.

Using Properties $P1$ as well as Corollaries 10 and 11, one can deduce that $\mid S_1 \mid$ and $\mid S_0 \mid$ give good approximations of *CAcc* and *CFail* in subsequent rounds.

Indeed, suppose first that $|S_1|>|S_0|$ at the end of the first round. Stations in S_1 continue sending, while stations in S_0 are not able to send. $CAcc_s$ stays stable for any station $s \in S_1$ during the second round, while $CFail_s$ decreases. For a station $s' \in S_0$, the contrary happens. A dual result holds when $|S_1|<|S_0|$.

Lemma 12 *Consider the sets A and F with $A = S_1$ and $F = S_0$ at the end of the first round and let s be any station about to send in the second round.*

1. *Suppose $|A|>|F|$.*
   (a) If $s \in S_1$ then $CAcc_s \geq |A|$ and $CFail_s \leq |F|$.
   (b) Let $s \in S_0$ then $CAcc_s \leq |F|$ and $CFail_s \geq |A|$.
2. *Suppose $|F|>|A|$.*
   (a) If $s \in S_0$ then $CAcc_s \geq |F|$ and $CFail_s \leq |A|$.
   (b) If $s \in S_1$ then $CAcc_s \leq |A|$ and $CFail_s \geq |F|$.

4.5 Second and later rounds: the model and properties

We use lemma 12 to model the full protocol in later rounds as shown in the `BOTTOM` part of Figure 3. Transition with guard $d1 < C1$ & $C1 > C0$ models stations from S_1 that are sending, while $d1 < C1$ & $C1 \leq C0$ corresponds to stations from S_1 which leave the `active` state because of the clique avoidance mechanism. Similar transitions correspond to stations from S_0.

If membership vectors are coherent again at the end of the second round, transition with guard $Cp = N$ & $!(C1 = 0)$ & $!(C0 = 0)$, that models the start of a new round with both sets non empty, should never be taken. Indeed, the invariant below at state `later rounds` is true:

$$AG\ !(!(C1 = 0)\ \texttt{and}\ !(C0 = 0)\ \texttt{and}\ (Cp = N)) \qquad (P6).$$

$P6$ is annotated as an assertion in Figure 3. $P6$ proves that the corresponding transition is never enabled, thus control leaves state `later rounds` as soon as $Cp = N$, i.e., after 1 round.

Further the following property is also verified as true when control moves from state `later rounds` to state `normal`:

$$AG\ (C1 = 0\ \texttt{or}\ C0 = 0) \qquad (P7).$$

$P7$ means that either S_1 or S_0 is empty at the end of the second round. Hence, all active stations have the same membership vectors at the end of the second round and form again a single clique in the graph theoretical sense.

4.6 Implementation

The automaton in Figure 3 has been translated in the formalism of Action Language [11] and If, and automatically verified using the corresponding verifier [13], [21] respectively. Verification has been conducted modularly. First, the `TOP` part with properties $P1$ to $P4$ have been verified, then the `BOTTOM` part with properties $P6$ and $P7$. The transition system construction took 0.84 sec. and the verification of $P1$ took 0.41 sec. using 16572416 bytes of memory with ALV, and LASH needed 27643969 byte(s) for the same property.

5 Automatic verification: the k faults case

To model the protocol for an arbitrary number N of stations and a given number k of faults, as for the case of 1 fault, we use a fixed (but proportional to k) number of counters to abstract the individual membership vectors and the individual *CAcc* and *CFail*. However, we will see that the case of k faults is not a mere generalization, it is more complex than the case of 1 fault.

5.1 First round

Let $1 \leq i \leq k$. By Proposition 2, after the occurrence of fault i, W is partitioned into sets S_w with $w \in \{0,1\}^i$. We find it handy for the following to indicate the length of the string w with the superscript i. We associate two counters Cw^i and dw^i to each set S_{w^i} that is formed after the occurrence of any fault i. The counters Cw^i counts how many stations belong to set S_{w^i} when fault i occurs. The counters dw^i count how many stations from the set S_{w^i} have sent between fault i and fault $i+1$ in case $i < k$, and counts how many stations from the set S_{w^i} have sent so far in the first round following fault k in case $i = k$. Again because of Proposition 2, we assume that, for any $w \in \{0,1\}^{i-1}$, $Cw^i1 + Cw^i0 = Cw^i$, $Cw^i1 \geq 1$ and $dw^i1 \geq 1$. For each fault i, we associate a counter $Cp(i)$ that counts how many time slots have elapsed since fault i.

These counters are almost enough to know $CAcc_s$ and $CFail_s$ for any station s in the first round following fault k. Indeed, let s be a station ready to send. s belongs to some set S_{w^k}. In the rounds preceding fault k and during the round following fault k, s recognizes as correct frames sent by stations from $S_{w'}$, where w' is a prefix of w^k, and recognizes as incorrect all other frames. This information is recorded with the counters dw^i, $w \in \{0,1\}^i$ and $1 \leq i \leq k$.

There is one more subtlety. The clique avoidance mechanism needs that $CAcc_s$ and $CFail_s$ count one round only, the round being relative to the position of the sending station s. To do so properly, we distinguish two cases.

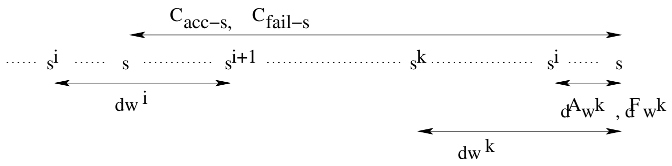

Fig. 4. Evaluating $CAcc_s$ and $CFail_s$ after fault k.

The first case is when fault k occurs in the first round following fault 1 and the time slot of s still belongs to that round. One must take into account that s has recognized as correct all stations that have sent before fault 1, which is a generalization of Theorem 3.

The second case is when the time slot of the sending station s lies between station s^i and s^{i+1} and fault k occurs in the first round following fault i, $i \geq 1$.

After fault i, s belongs to some set S_{w^i} and the number of frames accepted as correct by s is given by dw^i. However, to count correctly $CAcc_s, dw^i$ is too much. One has to withdraw all stations accepted by s whose time slots are between s^i and s. This is illustrated in Figure 4. We introduce auxiliary counters $d^A w^k$ and $d^F w^k$. These counters are set to 0 when fault k occurs. Counter $d^A w^k$ counts how many stations from set S_{w^k} have sent so far, as counters dw^k do, and counter $d^F w^k$ counts how many stations from set S_{w^k} were preventing from sending so far by the clique avoidance mechanism and moved to the set of non-working stations. The difference with dw^k is that these counters are reset to 0 each time a counter $Cp(i)$ reaches N after fault k. Thus $dw^i - \Sigma_{w'^k} d^A w'^k - \Sigma_{w'^k} d^F w'^k$, with w^i a prefix of w'^k, gives exactly how many frames between s and s^{i+1} the station s has recognized as correct in the round, and $dw^i - \Sigma_{w'^k} d^A w'^k - \Sigma_{w'^k} d^F w'^k + dw^{i+1} + \cdots + dw^k$ gives exactly how many frames in total s has recognized as correct in the round, i.e., $CAcc_s$. A similar idea works for $CFail_s$.

Proposition 13 *Let $s \in S_{w^k}$ a station ready to send in the round following fault k.*

1. *If $Cp(1) \leq N$ at the time slot of s, then:*
   $CAcc_s = \mid W \mid - \Sigma_{w'} dw'$, *and* $CFail_s = \Sigma_{w'} dw'$,
   where w' must not be prefix of w^k.
2. *Let $i < k$ such that $Cp(i) \geq N$ and $Cp(i+1) < N$ at the time slot of s. Then:*
   $CAcc_s = (\Sigma_{j=i}^{j=k} dw^j) - \Sigma_{w'^k} d^A w'^k - \Sigma_{w'^k} d^F w'^k$, *$w^i$ is a prefix of w^k and of w'^k, and*
   $CFail_s = (\Sigma_{j=i}^{j=k} \Sigma_{w'^j} dw'^j) - \Sigma_{w''^k} d^A w''^k - \Sigma_{w''^k} d^F w''^k$,
   where w'^j, w''^k must not be a prefix of w^k, and w'^i must be a prefix of w''^k, $i \leq j \leq k$.

5.2 Later rounds and the model

At the end of the first round, counters dw^k are kept as they are while counters $d^F w^k$ are reset to 0 and incremented during the second round as before, i.e., $d^F w^k$ counts how many stations from set S_{w^k} were preventing from sending so far by the clique avoidance mechanism and moved to the set of non-working stations. For the second round following fault k, $CAcc_s$ and $CFail_s$ are calculated with these counters only.

Proposition 14 *Let $s \in S_w^k$ a station ready to send in the second round following fault k. Then:*
$CAcc_s = dw^k - d^F w^k$, *and*
$CFail_s = \Sigma_{w'^k} dw'^k - \Sigma_{w'^k} d^F w'^k$ *with* $w'^k \neq w^k$.

The proof of this lemma uses the fact that if a station sends in the second round following fault k, then it has sent also in the first round following fault k.

Using all these counters, an automaton similar to the one given in Figure 3 can be designed and, in theory [1], automatically verified. Properties analogous to $P6$ and $P7$ have to be checked to prove that after the second round following fault k, there is only 1 clique.

6 Conclusion

We have proposed an approach for verifying automatically a complex algorithm which is industrially relevant. The complexity of this algorithm is due to the very subtle dynamic which is hard to model. We have shown that this dynamic can be captured faithfully by means of unbounded (parametric) counter automata. Even if the verification problem for these infinite-state models is undecidable in general, there exists many symbolic reachability analysis techniques and tools which allow to handle such models.

Our approach allows to build a model (counter automaton) for the algorithm with an arbitrary number of stations, but for a given number of faults. We have experimented our approach by verifying in a fully automatic way the model in the case of one fault, using the ALV tool and the LASH tool.

Related Work: [5] provides a manual proof of the algorithm in the 1 fault case. Theorem 1 generalizes this result to the case of any number of faults. On the other hand, [5] considers the first and second successor feature in the implicit acknowledgment algorithm, which is omitted in our work.

As far as we know, all the existing works on automated proofs or verifications of the membership algorithm of TTP concern the case of one fault, and only symmetric fault occurrences are assumed. In our work, we consider the more general framework where several faults can occur, and moreover, these faults can be asymmetric. In [22], a mechanised proof using PVS is provided. [17, 6, 15] adopt an approach based on combining abstraction and finite-state model-checking. [17] has checked the algorithm for 6 stations. [6, 15] consider the parametric verification of n stations; [15] provides an abstraction proved manually whereas [6] uses an automatic abstraction generation technique, both abstractions leading to a finite-state abstraction of the parameterized network. To tackle this more general framework, we provide an abstraction which yields a counter automaton and reduce the verification of the algorithm to the symbolic reachability analysis of the obtained infinite-state abstract model. Moreover, our abstraction is exact in the sense that it models faithfully the emission of frames by stations.

Future Work: One future work is to consider the feature involving first and second successor in the implicit acknowledgment algorithm. Our work indicates that this feature could be left out, as far as clique avoidance and coherence of membership vectors is concerned, since without it, stabilization occurs after two rounds following the last fault in the general case of k asymmetric faults. However, this feature allows a quicker detection of send-faulty stations. We conjecture that all results of Section 3 go through. However the abstraction issue

[1] In the case of 2 faults, we got memory problems, both with ALV and LASH.

seems more tricky. Another interesting direction is to automatize, for instance using a theorem prover, the abstraction proof which allows to build the counter automaton modeling the algorithm. More generally, an important issue is to design automatic abstraction techniques allowing to produce infinite-state models given by extended automata. Finally, a challenging problem is to design an algorithmic technique allowing to verify automatically the algorithm by taking into account simultaneously both of its parameters, i.e., for any number of stations *and for any number of faults*.

References

1. P.A. Abdulla, A. Annichini, S. Bensalem, A. Bouajjani, and Y. Lakhnech P. Habermehl. Verification of Infinite-State Systems by Combining Abstraction and Reachability Analysis. In *CAV'99*. LNCS 1633, 1999.
2. Parosh Aziz Abdulla and Bengt Jonsson. Channel representations in protocol verification. In *CONCUR'01*, volume 2154 of *Lecture Notes in Computer Science*, pages 1–15, 2001.
3. Parosh Aziz Abdulla and Bengt Jonsson. Ensuring completeness of symbolic verification methods for infinite-state systems. *Theoretical Computer Science*, 256:145–167, 2001.
4. A. Annichini, E. Asarin, and A. Bouajjani. Symbolic techniques for parametric reasoning about counter and clock systems. In *12th Intern. Conf. on Computer Aided Verification (CAV'00)*. to appear in LNCS, Springer-Verlag, 2000.
5. G. Bauer and M. Paulitsch. An investigation of membership and clique avoidance in ttp/c. In *Proceedings 19th IEEE Symposium on Reliable Distributed Systems (SRDS'00)*, Nürnberg, Germany, pages 118–124. IEEE Computer Society, Oct. 2000.
6. K. Baukus, Y. Lakhnech, , and K. Stahl. Verifying universal properties of parameterized networks. In *Proceedings of the 5th International Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems FTRTFT 2000*, Pune, India, 2000.
7. B. Boigelot and P. Wolper. Symbolic Verification with Periodic Sets. In *CAV'94*. LNCS 818, 1994.
8. A. Bouajjani, J. Esparza, and O. Maler. Reachability Analysis of Pushdown Automata: Application to Model Checking. In *CONCUR'97*. LNCS 1243, 1997.
9. A. Bouajjani and P. Habermehl. Symbolic Reachability Analysis of FIFO-Channel Systems with Nonregular Sets of Configurations. In *ICALP'97*. LNCS 1256, 1997. Full version in TCS **221** (1/2), pp 221-250, 1999.
10. A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *CAV'00*. LNCS 1855, 2000.
11. T. Bultan. Action language: a specification language for model checking reactive systems. In *Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000)*, University of Limerick, Ireland, pages 335–344, June 2000.
12. T. Bultan, R. Gerber, and C. League. Verifying systems with integer constraints and Boolean predicates: A composite approach. In *Proc. of the Intern. Symp. on Software Testing and Analysis*. ACM Press, 1998.
13. T. Bultan and Tuba Yavuz-Kahveci. Action language verifier. In *Proceedings of the 16th IEEE International Conference on Automated Software Engineering (ASE*

2001), Coronado Island, California, pages 382–386. IEEE Computer Society, Nov. 2001.
14. Patrick Cousot and Nicholas Halbwachs. Automatic discovery of linear restraints among variables of a program. In *POPL'78*. ACM, 1978.
15. Sadie Creese and A.W. Roscoe. Ttp: a case study in combining induction and data independence. Technical report tr-1-99, Oxford University Programming Research Group, 1999.
16. S. Graf and H. Saidi. Construction of abstract state graphs with pvs. In *CAV'97*. LNCS 1254, 1997.
17. S Katz, P. Lincoln, and J. Rushby. Low-overhead time-triggered group membership. In *7th International Workshop on Distributed Algorithms (WDAG'97)*, volume 1320 of *Lecture Notes in Computer Science*, pages 155–169. Springer-Verlag, 1997.
18. Y. Kesten, O. Maler, M. Marcus, A. Pnueli, and E. Shahar. Symbolic Model Checking with Rich Assertional Languages. In *CAV'97*. LNCS 1254, 1997. Full version in TCS **256**, pp 93-112, 2001.
19. H. Kopetz. Ttp/c protocol - specification of the ttp/c protocol. www.tttech.com, 1999.
20. H. Kopetz and G. Grünsteidl. A time triggered protocol for fault-tolerant real-time systems. In *IEEE Computer*, volume January, pages 14–23. 1994.
21. The Liège Automata-based Symbolic Handler (LASH). Available at `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.
22. H. Pfeifer. Formal verification of the ttp group membership algorithm. In *IFIP TC6/WG6.1 International Conference on Formal Description Techniques for Distributed Systems and Communication protocols and Protocol Specification, Testing and Verification, FORTE/PSTV 2000*, Pisa, Italy, Oct. 2000.
23. H. Saidi and N. Shankar. Abstract and Model Check while you Prove. In *CAV'99*. LNCS 1633, 1999.
24. P. Wolper and B. Boigelot. Verifying systems with infinite but regular state spaces. In *CAV'98*, 1998.