Sustaining Property Verification of Synchronous Dependable Protocols Over Implementation*

Péter Bokor[1,2], Marco Serafini[1], Áron Sisak[2], András Pataricza[2] and Neeraj Suri[1]

Technical University of Darmstadt, Germany[1]
{pbokor,marco,suri}@informatik.tu-darmstadt.de

Technical University of Budapest, Hungary[2]
{petbokor,sisak,pataric}@mit.bme.hu

Abstract

It is often considered that a protocol that has been verified for its dependability properties at the protocol level maintains these proven properties over its implementation. Focussing on synchronous protocols, we demonstrate that this assumption can easily be fallacious using the example of an existing formally verified diagnostic protocol as implemented onto the targeted time-triggered architecture (TTA). The cause is identified as the overlap mismatch across the computation and communication phases in TTA, which does not match the system assumptions of the protocol. To address this mismatch problem, we develop the concept of a generic alignment (co-ordination) layer to implement the desired communication assumptions. The strength of this approach is that the verification of this alignment layer ensures that the formally proved properties of a protocol also hold over their implementation.

1. Introduction

To manage the complexity of formal verification of distributed protocols, a divide-and-conquer approach is often advocated. For synchronous round-based protocols implemented on the targeted time-triggered (TT) systems, a general approach is to first verify the protocol based on its abstract specification, and then to prove that, under certain design constraints, the TT implementation of the protocol preserves the verified properties. By first establishing that the platform implements the system assumptions made by the protocol, it is possible to prove a *simulation relation* between the abstract protocol definition and the model of the TT implementation, which shows that the two models proceed through the same sequence of system states [9]. The first step must be repeated for each single protocol, whereas the second can be done once for all, significantly reducing verification efforts.

In this paper we elaborate how the Time-Triggered Architecture (TTA) [5] is able to support the execution of a synchronous round-based protocol preserving its fault-tolerance properties. Time-triggered systems can be broadly defined as systems where all nodes have synchronized clocks and they execute specific operations at specific times following an a-priori deterministic schedule. We contrast TTA with other time-triggered systems that assume more co-ordination among the nodes, thus facilitating a property preserving implementation of synchronous round-based protocols.

In TTA, the nodes access a *shared* communication medium using a Time Division Multiple Access (TDMA) scheme to determine the time when a node is allowed to send messages on the bus. This is in contrast with *frame-based* TT systems (e.g., [4]), where nodes are assumed to send messages via dedicated channels at approximately the same time following a sequence of communication and computation phases. The latter behavior directly implements the main assumption of synchronous round-based algorithms[1]. In TTA, in order to primarily minimize the latency of control applications, the computation phase of individual nodes is staggered over the time of communication in the cluster. As a consequence, the constraints defined to prove property preservation of frame-based TT implementation of synchronous round-based protocols [9] are *not* met by TTA.

Contributions To substantiate this claim, we consider a formally verified, synchronous round-based diagnostic protocol and show that, in its original form, it fails to guarantee the desired behavior in a TTA system. We use model checking to obtain counterexamples that violate the correctness of

*Research supported in part by EC DECOS & ReSIST

[1]We use "protocol" and "algorithm" as synonyms, following the different terminology in the referred literature. Similarly, in some places, dependability and fault-tolerance are used interchangeably.

the algorithm.

We then introduce a generic *alignment layer* concept that can be used to simulate the communication properties assumed by the protocol. We show how, through the use of the layer, the original round-based algorithm can satisfy the desired correctness and fault-tolerance properties in the TTA model. The example of the diagnosis algorithm and the TTA platform are used as a case study of the general concept.

Overall, we propose a general framework to ease the verification of distributed, fault-tolerant protocols over their implementation. In particular, we prove that the alignment layer on top of TTA yields the assumptions of synchronous, round-based protocols, thus sustaining the original properties. We remark that the proposed framework is not restricted to a specific class of protocols, neither to the alignment layer (which can be of varied complexity), and nor to the underlying implementation platform. Accordingly, the time-triggered implementation of round-based protocols over frame-based systems [9] or over the TTA (as presented) can be treated as special cases.

If the alignment layer can be verified, the framework not only provides that already verified protocols automatically transform to their implementation, but also facilitates the design and verification of new algorithms to the specific platform. In fact, since the alignment layer guarantees that the protocol's assumptions are met, the implementation platform does not need to be reflected in the formal model. In particular, instead of modeling the protocol together with the model of TTA (like in [1, 8], for example), the simple model of the protocol prunes the size of the state space (e.g., in order to mitigate state space explosion in model checking), also it makes the protocol design less error-prone.

Organization In Section 2, the frame-based and TTA models of time-triggered systems are contrasted on their implementation of round-based algorithms. In Section 3, we perform formal analysis of a round-based diagnostic algorithm to elaborate its correctness in both time-triggered implementations. We also present the fault model and verify that the TTA implementation of the algorithm violates the specified requirements. The message alignment layer is proposed as a solution. In Section 4, we propose a general framework to verify the implementation of distributed protocols. We prove that message alignment is able to provide the assumptions of any round-based protocol, and validate the soundness of the proposed framework via our case study. Section 5 discusses the benefits of the proposed approach based on experimental data. Section 6 concludes the results of the paper.

2. Flavors of Time-triggered Systems: Implementing Round-based Protocols

In this section, we first define (synchronous) round-based algorithms. Next, we present the *frame-based* (Section 2.1) and *TTA* (Section 2.2) time-triggered systems as possible platforms to implement such algorithms. In both cases we elaborate how the assumptions of round-based algorithms can be consolidated onto the given platform.

In [7], Lynch identifies a class of distributed algorithms, where the nodes of the algorithm proceed through rounds. We refer to these as *round-based* algorithms. We also use the term *untimed* (as in [9]) reflecting that such systems do not use the notion of time, since perfect synchrony is assumed. This is in contrast to synchronous systems in real applications, where synchrony can be achieved only to a certain extent and the untimed abstraction must be thus, as discussed in the following, implemented. Within a round every node executes the following two phases: In the *communication phase* messages computed via a message generation function are sent and delivered to other nodes; afterwards, in the *computation phase*, the local state of the node is updated by the state transition function based on the messages sent by the other nodes. As it is assumed that nodes execute the different phases in lock-step, we call such systems *synchronous*.

Overall, time-triggered systems can generally be defined as distributed systems, where nodes execute specific operations at a-priori scheduled times, based on synchronized local clocks. The definition of synchronous round-based protocols does not take into account that even in synchronous distributed systems it is impossible to guarantee that actions are executed at exactly the same time, i.e., the lock-step assumption is violated. In the following sections we discuss how to handle this in two different classes of TT systems, i.e., frame-based and TTA.

2.1. Frame-based Time-Triggered Systems

The basic definition of TT systems encompasses multiple type of systems, using different communication and computational paradigms. The *frame-based* class of TT systems (e.g., [4]) considered by Rushby in [9] is very close to the synchronous round-based model of computation (Figure 1). *Node*s proceed through *round*s in parallel, such that in each round global *communication* and *computation phases* can be identified. In the communication phase every node is allowed to send messages to some of the other nodes, and every message from correct processes is assumed to be delivered before the end of the communication phase. The computation phase begins after all nodes have sent the messages and all messages have been delivered. Every node executes local jobs, which read the col-

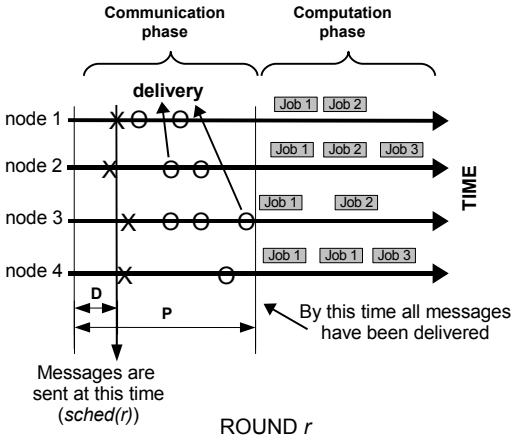

Figure 1. Frame-based model of a time-triggered system: Communication and computation are strictly successive

lection of messages currently available in the input channels and generate the messages that will be sent in the next round.

As TT systems are synchronous, there are known upper bounds on the execution time of jobs and on communication delays among correct nodes. Each correct node is equipped with a local clock of high precision, i.e., the *drift* between clock time and physical time is bounded. Although a clock synchronization protocol is used to keep the local clock of nodes close to each other, it cannot be guaranteed that they have exactly the same local clock time. Clock synchronization guarantees that the values of local clocks periodically converge and the difference of local clock time between any couple of correct nodes (called the *skew*) is bounded, though in general strictly greater than zero. As a consequence, different nodes can start the different phases at slightly different physical times (e.g., varying times of sending the messages in Figure 1).

The motivation of Rushby's work [9] is to abstract the details of (time-triggered) implementation of round-based protocols, such as synchronization and convergence of local clocks, whilst conducting formal analysis to verify correctness and fault-tolerance. Rushby concludes that the formal analysis might fail to identify the basic mechanisms of *why* the algorithm is fault-tolerant, if the formal model of the algorithm contains implementation-related details (like in [6], for example). To establish a systematic framework for formal verification of fault-tolerant time-triggered algorithms, a simulation relation is defined between the round-based algorithm and the frame-based TT implementation, i.e., it is proven that the implementation proceeds through the same sequence of "global states" (i.e., an array of local states of the nodes) as the round-based algorithm.

According to Rushby's model, each node initiates round r at local time $sched(r)$ and begins the communication and computation phases after global fixed offsets respectively called D and P (see Figure 1). To reproduce the lock-step behavior assumed by synchronous round-based protocols, these offsets must be large enough to let each node the time to complete one phase before any other node initiates the next. The paper identifies the necessary constraints (e.g., on the offsets D and P, clock drift and skew, etc.) that guarantee equivalent implementation of untimed protocols.

We show that this model, while in accordance with frame-based TT systems, does not match the communication and computation model of the TTA. As a consequence, it will be shown that a round-based protocol can violate its specification in a TTA system.

2.2. Time-Triggered Architecture (TTA)

The TTA is also a time-triggered system, however, exhibiting different communication and computation patterns than frame-based systems. In the TTA model (Figure 2) the scheduling of each node is not required to follow a global parallel sequence of communication and computation phases. The communication phase of different nodes is staggered, as the TDMA access scheme assigns a different *sending slot* to each node periodically, i.e., in every TDMA round. In fact, a *global schedule* assigns the i^{th} sending slot to node i. As the system is synchronous, all messages will be delivered to all other nodes within a bounded time, after which the communication phase of a node is completed[2]. The rationale of the TDMA communication is that TTA systems use a shared (duplicated) bus and each message is *broadcast* by the sender to all other nodes. Besides the global communication scheduling, each node has an internal *node schedule* determining when *jobs* are executed. The scheduling of the execution of jobs is independent of the TDMA scheme. A job can be any application running (distributed) on the nodes, possibly implementing a distributed protocol (e.g., clock synchronization).

Hence, the definitions and requirements identified by Rushby on the global schedule of nodes [9] do not apply to TTA systems, as $sched(r)$, D and P are not globally defined. Consequently, implementations of a synchronous round-based protocols on TTA systems do not (in general) guarantee the desired properties, as shown in the following section. In [10] we propose a *message alignment* layer, which can be used to implement any synchronous round-based protocols for any possible node schedule. The intuition is to introduce an additional *latency* in processing the messages, as at round r all messages sent in round $r-1$ are

[2]Due to the drift and skew of the local clocks, as well as the latency of the message delivery, the global schedule is determined such that correct nodes agree on the current slot.

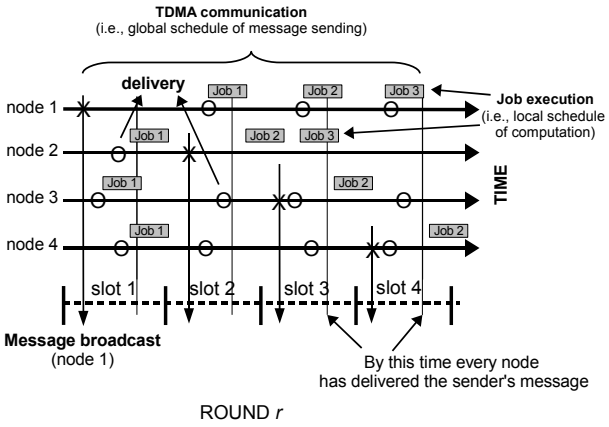

Figure 2. TTA model of a time-triggered system: Communication and computation overlap

available at every node. In the following section, after describing the operations of the layer, we prove its properties and use it to formally verify a diagnostic protocol for TTA.

3. Case Study: Verifying Inconsistency of a Round-Based Diagnostic Protocol on TTA

In [11] the authors define and formally verify the synchronous round-based diagnostic algorithm DD[3]. As discussed earlier, the fact that DD (as a round-based protocol) can be formally proven to satisfy the desired properties implies that the implementation of DD in a frame-based TT system shall guarantee the same correctness and fault tolerance [9]. To substantiate that the implementation of an abstract protocol might not exhibit the desired fault-tolerance properties, we will use DD to demonstrate that round-based protocols indeed violate correctness if implemented on the TTA.

Algorithm DD executes in two rounds, each split into communication and computation phases:

Round 1 - Local Detection

1. *Sending workload (communication):* Every node sends its workload message(s) generated by the application(s) running on the node.
2. *Local detection (computation):* Every node locally detects errors of the other nodes (through monitoring incoming messages) and forms a *local syndrome* indicating the faulty/correct status of each node.

[3]DD is chosen as an example and also given the public availability of its detailed specification and formal verification.

Round 2 - Dissemination and Analysis

1. *Global dispersion (communication):* Every node broadcasts the local syndrome, i.e., the local syndrome is sent to every other node.
2. *Global assimilation (computation):* The local syndromes received from the other nodes, together with the syndrome obtained during the local detection, are compiled into a *syndrome matrix*. The i^{th} row of the matrix corresponds to the local syndrome received from the i^{th} node.
3. *Analysis (computation):* Every node computes the majority of the values of each column[4] of the syndrome matrix to derive the *global health vector* as the output of DD. Each value of the global health vector indicates the faulty/correct status (binary value) of the relative node.

In the following, we show that DD cannot guarantee the desired properties in TTA systems without the service of a message alignment layer, which buffers messages sent and received on the TTA bus to implement the communication and computation pattern assumed by the protocol.

3.1. The Fault-tolerant Diagnosis Protocol

As we are interested in fault-tolerant protocols, the properties of the protocol need in fact to be proved against a fault model. DD assumes a *hybrid* fault model, i.e., besides pessimistic (e.g., malicious) fault modes, less severe fault manifestations are also considered [11]. As reported in many applications (e.g., [10]), the classification of hybrid fault classes contributes to a better fault coverage and a finer diagnostic resolution. The hybrid fault model of DD defines that no assumption is made to *malicious* (or Byzantine) nodes (i.e., they can send arbitrary messages), and that *benign* nodes (e.g., crash faults) can be locally detected by every correct node (based on the erroneous/missing messages sent by the faulty node). If the maximum number of malicious and benign nodes are respectively denoted by a and b, the fault hypothesis (i.e., the number of tolerated faults) of DD defines $N > 2a + b + 1$, where $a \leq 1$ and N is the overall number of nodes. For more details about the the fault classes and the fault hypothesis of DD, the reader is referred to read [11].

We remark that the same fault model will be assumed when adopting DD on the TTA platform. Symmetric faults in the TTA are justified by the bus topology, however, asymmetric faults (e.g., [1],[10]) and even Byzantine behavior [3] are possible in real TTA implementations.

[4]In fact, *hybrid majority* is computed, i.e., incorrectly delivered local syndromes and the node's opinion on itself are omitted in the voting; by definition, the output is set to faulty if no value is in majority.

Diagnostic protocols usually require correctness (i.e., correct nodes are never diagnosed as faulty), completeness (i.e., faults are eventually detected) and consistency. To prove that DD is not correct under the TTA, we show will that consistency is violated.

Consistency All correct nodes agree on the diagnosis (i.e., on the global health vector).

We note that correctness and completeness are also violated, however, for the sake of our argumentation we restrict ourselves to consistency.

3.2. Implementation Problems on the TTA

In this section we show that DD, although being formally verified under the untimed model [11], violates consistency in the TTA. We use the SAL symbolic model checker [2] to obtain counterexamples for consistency. A counterexample is an execution of the protocol violating the defined property, in particular, it is a run in DD such that a node is diagnosed respectively correct and faulty by two correct nodes (thus violating consistency).

To obtain the SAL model of the TTA-based implementation of DD, we assume that the diagnostic job is executed once in every TDMA round. Furthermore, each of the N nodes is assigned an $ID \in [1, N]$. Between two executions of a diagnostic job all nodes have the chance to send messages, which are then used in the computation of the new state. If the execution of a node's diagnostic job is scheduled after slot j, the messages sent in the current round are available from nodes $[1, k]$, where $k \leq j$, depending on how messages are delivered from the communication layer to the application (e.g., considering the delay caused by error detection mechanisms). Similarly, the new message (as the output of the execution of the diagnostic job) is available at the communication controller from slot l on, where $l \geq (j + 1)$. The latter delay is mainly determined by the execution time of the diagnostic job. In the following examples we show that differing scheduling times of the diagnostic jobs (i.e., node schedule) can result in inconsistencies of the global health vector. In fact, the node schedule of all applications running on the TTA cluster is calculated and optimized a-priori by the scheduler. Diagnosis is possibly a low priority application, thus no assumption can be made about the scheduling of diagnostic jobs. For simplicity, we assume (**Assumption 1**) that within each TDMA round the diagnostic job of each node is executed *after* the sending slot of the relative node. For example, Job 2 in Figure 2 satisfies Assumption 1.

In the SAL model of DD running on the TTA, we considered the following special case: `dj[i]` indicates the scheduling of the diagnostic job at node *i*; if `dj[i]=j` ($j \in [0, N-1]$) then the diagnostic job of node i is assumed to be executed such that it can read the messages sent in the current round by nodes $[1, j]$ and in the previous round by nodes $[j + 1, N]$. If a diagnostic job is executed after the last sending slot of a round and can read data from all nodes of the current round, we treat it as it was executed in the next round and set `dj[i]=0` accordingly. Furthermore, if `dj[i]=j`, we assume that the new message is available at the communication controller from slot $(j + 1)$ on.

The SAL model is called `DD`, and `consistency` is formulated as an invariant using the temporal logic operator **G**. The SAL theorem requires that the global health vector of any two correct nodes (`ghv[i]` and `ghv[j]`) are always consistent:

```
consistency: THEOREM
  DD |- G(FORALL(i,j,k:node):
    (non_faulty(i) AND non_faulty(j) =>
       ghv[i][k] = ghv[j][k]));
```

The SAL model checker fails to prove the theorem and produces the following counterexample (in the initial model the number of nodes is restricted to four). It can be seen that the node schedule fulfills Assumption 1, since `dj[i]` $\leq$ `i`, for all i. Furthermore, `fv[i]` indicates the actual fault status of the system: Node 2 is benign faulty in the TDMA round $(r - 1)$. For simplicity, it is assumed that faults are permanent. Consistency is violated, because (in round *r*) node 1 disagrees with node 4 on the diagnosis of node 2.

```
Counterexample 1:
=======================
...
%Node schedule of diagnostic jobs
dj[1] = 1
dj[2] = 2
dj[3] = 3
dj[4] = 0
...
%Fault vector in round r-2
fv[1] = non_faulty
fv[2] = non-faulty
fv[3] = non_faulty
fv[4] = non_faulty
...
%Fault vector in round r-1
fv[1] = non_faulty
fv[2] = benign
fv[3] = non_faulty
fv[4] = non_faulty
...
%Result of diagnosis at round r
ghv[1][1] = non_faulty
ghv[1][2] = non_faulty %INCONSISTENCY
ghv[1][3] = non_faulty
ghv[1][4] = non_faulty
...
ghv[4][1] = non_faulty
ghv[4][2] = faulty    %INCONSISTENCY
ghv[4][3] = non_faulty
ghv[4][4] = non_faulty
```

Inconsistency is caused by the overlap of communication and computation in TTA. As the diagnostic jobs running on

nodes 1 and 4 are scheduled respectively before and after the sending slot of the faulty node (2), 1 detects the fault later than 4. The run leading to an inconsistent diagnosis at round r is the following. Node 1 reads local syndromes sent in round $r-1$. From the node schedule of the diagnostic jobs, none of nodes 3 and 4 can report the fault in round $r-1$ (thus, building a majority), therefore, node 2 is deemed as non-faulty by node 1. On the other hand, node 4 has detected node 2 as faulty and receives another accusation to node 2 sent in round r by node 3. This is sufficient to deem 2 as faulty. Note that node 2's self-syndrome is omitted in the majority voting.

Read alignment As a remedy to the problem identified in Counterexample 1, we now introduce an alignment layer to delay the processing of the messages until all correct jobs can read the same set of messages. The reason why DD misbehaves in the TTA model is that the diagnostic information of two successive rounds overlaps. In fact, as diagnostic jobs of different nodes execute analysis at round r, they use diagnostic information related to different rounds. This is because the node schedule is not constrained and diagnostic jobs running of different nodes can read diagnostic messages of varying freshness.

To separate diagnosis of different rounds, a *read alignment* is used [10]. The main idea is that every node buffers the messages sent by other nodes, and computation is done based on messages that can be consistently read by all jobs running in the same round. Every diagnostic job reads the diagnostic messages $m_1, ..., m_N$ and we define the node schedule to be defined by l_i with the same semantics as `dj[i]` in the SAL model. It is assumed that messages are updated in the sending order, following the TDMA scheme. To let all diagnostic jobs executed at round r use consistent diagnostic information from the previous round (i.e., data of the same freshness), a read alignment layer is invoked providing the following service: Messages $m_1, ..., m_{l_i}$ as read in the previous round and $m_{l_i+1}, ..., m_N$ as read in the current round are collected and sent to the relative node. Accordingly, every time a diagnostic job is executed, it invokes the message alignment layer to access consistent diagnostic data, instead of reading directly the messages available after delivery.

To rectify inconsistency of DD in the TTA, we install the read alignment layer (see Figure 3) between the communication layer (i.e., TDMA scheme) and the application layer (i.e., jobs running on different nodes). Accordingly, the diagnostic jobs access data from the read alignment layer in both rounds of the protocol. During local detection, every node detects faults of other nodes based on the incoming messages of the previous TDMA round provided by the read alignment layer; local syndromes are then formed accordingly. During global assimilation, local syndromes are accessed from the layer, i.e., syndromes sent in the previous round are collected; the syndrome matrix can be then compiled and majority voting is calculated.

Further problems We augmented `DD` with the model of the read alignment layer and ran model checking again. Also, we removed Assumption 1. As a result, consistency still cannot be proven and the following counterexample is provided.

```
Counterexample 2:
=======================
...
%Node schedule of diagnostic jobs
dj[1] = 1
dj[2] = 2
dj[3] = 3
dj[4] = 4
dj[5] = 4
...
%Fault vector in round r-3
fv[1] = non_faulty
fv[2] = non_faulty
fv[3] = non_faulty
fv[4] = non_faulty
fv[5] = non_faulty
...
%Fault vector in round r-2
fv[1] = non_faulty
fv[2] = benign
fv[3] = non_faulty
fv[4] = non_faulty
fv[5] = non_faulty
...
%Fault vector in round r-1
fv[1] = malicious
fv[2] = benign
fv[3] = non_faulty
fv[4] = non_faulty
fv[5] = non_faulty
...
%Result of diagnosis at round r
...
ghv[4][1] = non_faulty
ghv[4][2] = non_faulty %INCONSISTENCY
ghv[4][3] = non_faulty
ghv[4][4] = non_faulty
ghv[4][5] = non_faulty
...
ghv[5][1] = non_faulty
ghv[5][2] = faulty     %INCONSISTENCY
ghv[5][3] = non_faulty
ghv[5][4] = non_faulty
ghv[5][5] = non_faulty
```

Prior to examining the execution of the protocol, we make two observations. First, the number of nodes has been increased to 5 to tolerate one benign and one malicious node at the same time. This is in accordance with the fault hypothesis of DD (see Section 3.1), therefore $N > 4$, if $a = 1$ and $b = 1$. Second, Assumption 1 is violated, because node 5 executes the diagnostic job such that it can send the new message (i.e., the message based on the updated local state)

in slot 5 of the current TDMA round (`dj[5]=4`). As a consequence, consistency is violated in round r, as node 4 and 5 disagree on the diagnosis of node 2. This is due to the violation of Assumption 1, since nodes can send diagnostic information of various freshness.

The counterexample is the following: At round $(r-1)$, local detection is done based on read aligned messages, which were supposed to be correctly sent at round $(r-2)$. As node 2 is faulty in that round, all correct nodes build an updated local syndrome containing the accusation on node 2 (denote it by $faulty(2)$) . According to the node schedule, nodes 3 and 4 cannot immediately send the updated local syndromes (since `dj[3]=3` and `dj[4]=4`), thus, they send the *prior* local syndromes referring to round $(r-3)$, which contain $correct(2)$. On the other hand, node 5 can immediately send the updated local syndrome, as `dj[5]=4`. Furthermore, the malicious node 1 can send local syndromes containing $correct(2)$ and $faulty(2)$ to node 4 and 5, respectively.

At round r, since the read alignment layer is used to read diagnostic data, the local syndromes sent in round $(r-1)$ are aggregated and used to calculate the majority. Accordingly, node 4 collects the values $\langle correct(2), x, correct(2), correct(2), faulty(2)\rangle$ from nodes 1 to 5 respectively (note that node 2's opinion about itself is denoted by 'x', since it is not considered in the voting). The diagnosis of node 2 obtained by node 4 through majority voting is thus $correct(2)$. On the other hand, node 5 receives $\langle faulty(2), x, correct(2), correct(2), faulty(2)\rangle$ and since no value is in majority, the default value is adopted to diagnose node 2, i.e., $faulty(2)$.

Send alignment In the example above, at round $(r-1)$, nodes 3 and 4 send local diagnosis about node 2 referring to round $(r-3)$, while node 5 sends fresher data from round $(r-2)$. To solve this problem, *send alignment* is proposed to delay the sending of fresh data if needed [10]. In fact, if all nodes can send fresh data or, equivalently, none of them can send fresh data (Assumption 1), such a mechanism is not needed, since all data refer to the same TDMA round. On the other hand, if $l_i \geq i$ (i.e., node i cannot send fresh data) and $l_j < j$ (i.e., node j can send fresh data), for some i and j, send alignment is needed to guarantee consistency: At every round, node i forms and sends the local syndrome based on the messages received from the read alignment layer in the same TDMA round, however, node j sends the syndrome based on the messages received in the previous TDMA round. Accordingly, the local syndromes assimilated and analyzed at round r were all sent at the previous round (read alignment), and they all refer to round $(r-2)$ or to round $(r-3)$ (send alignment). The layer incorporating both read and send alignment is called *message alignment layer*. Note that the minimum delay imposed by the message alignment layer is two TDMA rounds, which can be guaranteed if $l_i < i$, for all i, and send alignment is not needed. In the following section, we prove that any round-based algorithm maintains its original requirements, if it is deployed on the TTA platform augmented with the message alignment layer.

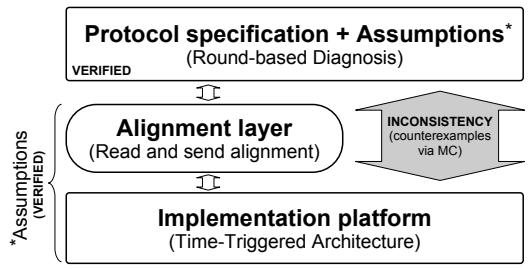

Figure 3. A framework for verification of distributed protocols over their implementation

4. The Alignment Layer: Transformation to Provide Equivalence

In this section, we propose a framework to formally verify distributed protocols over their implementation (Section 4.1). In particular, we show how the framework applies to round-based, synchronous protocols implemented over the TTA (Section 4.2). Finally, in order to validate our approach, we demonstrate that the straightforward verification of the example of our case study provides the same results as the proposed framework (Section 4.3).

4.1. A Framework To Verify Distributed FT Protocols over Implementation

Figure 3 depicts the main elements of the proposed verification framework. Given the abstract *specification* of a protocol together with the system *assumptions* (communication, synchrony, etc.), the protocol shall be implemented on a specific *platform*, such that the original specification remains fulfilled. The implementation platform possibly violates the protocol's assumption, in such cases an *alignment layer* is necessary. Instead of formally verifying the composite model of the protocol and the implementation platform, we propose to prove that the layer on top of the platform satisfies the assumptions. It is assumed that the protocol is formally *verified* against its abstract specification. Accordingly, the implementation of other protocols on the same platform can be done without verifying the alignment layer again. In Figure 3, the prototype verification settings of this paper are written in brackets being specific to model

checking a round-based diagnosis protocol over the TTA. The next section proves that read and send alignment in TTA indeed guarantee the desired assumptions.

We remark that the implementation of round-based protocols on a frame-based, time-triggered platform [9] similarly conforms to the proposed verification scheme. In [9], the alignment layer identifies the necessary constraints posed on the global time constants of the frame-based TT platform to prove that the assumptions hold. Finally, note that formal techniques applied in the framework are not specified. We use model checking to verify the abstract specification of the protocol, in [9] theorem proving is utilized for the same purpose. In both cases, the properties of the alignment layer are proven by hand.

4.2. A Proof: Message Alignment over TTA Guarantees the Lock-step Assumption

In Section 3, we pointed out the possible problems of directly implementing round-based protocols on top of TTA, and proposed a message alignment layer to guarantee the requirements of a specific diagnosis protocol. In this section, we prove that the message alignment layer ensures that the assumption of parallel alternation of communication and computation phases at different nodes is held on top of the TTA. Therefore, any round-based algorithm sustains its properties if implemented over the TTA. In fact, we use induction to show that the TTA implementation proceeds through the same sequence of states as the round-based protocol, however, with a delay of one or two TDMA rounds. This means that the TTA-implementation of a round-based protocol with r rounds terminates after $2(r-1)$ (without send alignment) or $3(r-1)$ TDMA rounds (with send alignment), depending on the node schedule[5]. The following result is analogous to that in [9] (as presented in Section 2.1), in fact, we prove that round-based protocols are simulated by the alignment layer over TTA.

We now formally define round-based protocols termed *untimed (synchronous) systems* after [9]. Compared to the general class of round-based protocols, we restrict assuming *broadcast communication*. This is in accordance with the communication in TTA systems. Furthermore, it is assumed that nodes have the same set of possible states. Note that this mathematical description does not restrict to distributed protocols where nodes execute the same program.

Definition 1. (Untimed synchronous system.) *An untimed (synchronous) system (*UT system *in short) consists of a set of* nodes *P (denote N the number of nodes), each node maintains the same set of* states *S and a set of* messages *M, node i defines a* state transition function[6] $t_i : S \times M^N \rightarrow S$, *and a* message generation function $m_i : S \rightarrow M$. *A set of* initial states *is defined by* $S^I \subseteq S$.

Every node starts from an initial state and the system makes the following two phases in lock-step, repeatedly in round *1,2,etc.: In the* communication phase *each node i generates a message (via* m_i*) that are sent to the other nodes (broadcast communication); in the* computation phase *node i updates its next state (via* t_i*) based on the messages received in the current round. The* global state *of the system at round* r *is the tuple of each node's state, and it is denoted by* $ut(r)$.

We now formally define the TTA-based implementation of an untimed system. This means that the same functions are assumed as those defined in the previous definition, however, the communication follows a TDMA scheme. It is also assumed that the TTA system implements the message alignment layer, i.e., read and message alignment are applied. We use a similar model to the one introduced in [10]: The communication controller of each node maintains the *interface variables* $m_1, ..., m_N$, where m_i is the message last read from node i. Interface variables are updated at every sending slot. The intuition of the protocol's execution is the following: At every TDMA round, each node executes the specified protocol at the scheduled time and reads messages from the other nodes via the read alignment layer; the node updates its local state based on these messages, generates a message, which will be broadcast via the send alignment layer.

Definition 2. (TTA system). *In the TTA system, P, S,* S^I*, M, and* t_i *and* m_i *of node i are defined as those for the UT system. The* node schedule *of executing* t_i *and* m_i *is defined by* $l_i \in [0, N-1]$ *for each node i.*

Every node starts from an initial state and executes the following operations, repeatedly in TDMA round *1,2,etc. If* $l_i < i$*, for all* i*,* $m_i(s_{i,r})$ *is sent by the* communication controller *(*$s_{i,r}$ *denotes the* i^{th} *node's state in the current round* r*). If* $l_i \geq i$*, send* $m_i(s_{i,r})$*, otherwise send* $m_i(s_{i,r-1})$ *(*send alignment*). The communication controller sends the message at slot* i *of the current TDMA round.*

The communication controller of each node buffers the values $m_{j,r}$*, for all* $j \in [1, N]$*, which is the message last received from node* j *at round* r*. The next state of node* i *is computed by* read alignment *if all messages are available:* $s_r = t_i(s_{i,r-1}, m_{1,r-1}, ..., m_{l_i,r-1}, m_{l_i+1,r}, ..., m_{N,r})$. *The* global state *of the system at the end of round* r *is the tuple of each node's state, and it is denoted by* $tta(r)$.

Our main result is that we prove that the UT system and the TTA system proceed through the same sequence of global states. According to the read alignment, the current

[5] In the initial round there is no receiving of messages, and similarly, no messages are sent in the final round. Therefore, the overall number of TDMA rounds is less than $2r$ and $3r$.

[6] For simplicity, we assume that a node sends the message to itself.

state of the node in the TTA system is computed based on the messages of the previous round, thus imposing a delay of one round. In addition, if send alignment is needed, certain nodes use their previous state to generate the message to send, rather than using the current one. This yields an additional round of delay. The following theorem states the equivalence which underpins the proposed verification framework for round-based protocols running on the TTA.

Theorem 1. *If the nodes start with the same initial states in the UT and the TTA system, the following holds at every round r. The additional delay is caused by the send alignment, i.e., if $l_i \geq i$, for some node i.*

$$tta(r) = ut(r-1) \text{ or } tta(r) = ut(r-2)$$

Proof. *The proof is an induction on the number of rounds.* Assume that $tta(r) = ut(r-1)$. Denote $s_{i,r-1}$ and $s'_{i,r}$ the states of the i^{th} node in the UT and TTA system, at round $r-1$ and r, respectively. The messages sent in the UT system at round $r-1$ are denoted by $m_{i,r-1}$, where $m_{i,r-1} = m_i(s_{i,r-1})$. Assume that $l_i < i$, for all i, send alignment is thus not needed. Therefore, the message $m'_{i,r}$ sent by the i^{th} node in the TTA system at round r is computed by $m_i(s'_{i,r})$. By definition, the next state $s_{i,r}$ of node i in the UT system is computed by $t_i(s_{i,r-1}, m_{1,r-1}, ..., m_{N,r-1})$. The next state $s'_{i,r+1}$ of node i in the TTA system is computed by applying t_i and the read alignment layer, $t_i(s'_{i,r}, m'_{1,r}, ..., m'_{l_i,r}, m'_{l_i+1,r+1}, ..., m'_{N,r+1})$. ¿From the definition of l_i, the interface variables $m_{l_i}, ..., m_N$ have not been updated yet, thus $m'_{j,r+1} = m'_{j,r}$. From the inductive assumption, $m_{i,r-1} = m'_{i,r}$, therefore $s_{i,r} = s'_{i,r+1}$ and the inductive proof follows.

If $l_i \geq i$ for some i, the send alignment is used and $m'_{i,r}$ is not available, thus $s'_{i,r+1}$ cannot be computed. In this case the messages sent at round r in the TTA system are computed by $m_i(s'_{i,r-1})$, i.e., based on the previous state available at all nodes. The reasoning is now similar to the previous one, and $tta(r) = ut(r-2)$ is proven. □

4.3. A Validation of Our Approach: Model Checking of DD over TTA

According to the proposed verification framework, it is not necessary to build and verify the model composing the distributed protocol under examination together with its intended implementation platform (call it *composite model*). Considering our case study, this approach can be justified if the model obtained as the composition of the models of the DD protocol, the message alignment layer and the TTA platform can be verified by model checking.

In fact, we created the model of the message alignment layer over TTA and integrated in `DD` (i.e., the SAL model of the original DD protocol). SAL's input language supports the composition of modules. A module in SAL is a functional building block of a larger model. In order to emphasize our treatment of message alignment as a layer of the overall architecture, the new SAL model is a synchronous composition of the following sub-modules: Every diagnostic job executing DD is a module; the message alignment layer as well as TDMA communication are also modeled as different modules. The composed model is called `DD_aligned` and the SAL model checker was able to prove the following theorem:

```
consistency: THEOREM
  DD_aligned |- G(FORALL(i,j,k:node):
    (non_faulty(i) AND non_faulty(j) =>
       ghv[i][k] = ghv[j][k]));
```

Accordingly, consistency can be proven to the implementation of DD over TTA, however, as discussed in Section 3.2, only after installing the message alignment layer.

N	consistency		correctness		completeness	
	RB	TTA	RB	TTA	RB	TTA
4	1s	7m	1s	3m	1s	55s
5	3s	36h 25m	2s	2h 23m	2s	1h 30m
6	10s	–	5s	13h 19m	5s	23h 57m
7	71s	–	21s	–	18s	–

Table 1. The results of model checking DD: The verification times if verifying the original protocol (RB) and the composite model (TTA)

5. Verification Efficiency based on Experiments

One advantage of the proposed verification scheme is that *existing* formally verified protocols provably sustain their properties over the implementation. Besides the reusability of established solutions, the framework also supports the design and verification of *new* protocols. Assuming that the properties of the alignment layer are proven, design and verification can be performed at the abstract level. The gain of this approach is two-fold:

(O1) The *computational complexity* of verifying the abstract specification instead of the composite model can be considerably less (even in the order of multiple magnitudes). This is due to the possibly vast number of conditional branches in the state space caused by the modeling of faults and implementation-related details.

(O2) The handling of large (and complex) models possibly yields additional complications in the *verification* and/or *design* of the protocol.

Table 1 depicts the results of model checking of the DD protocol against its FT properties (i.e., consistency, correctness and completeness). Model checking was performed based on the original round-based model (RB) and on the composite model (TTA) of the protocol. The experiments were run by using the SAL Bounded Model Checker (BMC) installed on a 2.6 Linux kernel, on a machine with a 2.66 Ghz Intel processor and 20 GB memory. In case the model checker could not prove the property due to the size of the state space (in fact, we used a timeout of 48 hours), no data is depicted in Table 1. It can be seen that the round-based model scales even if $N \geq 7$, while for the composite model the state space explodes for relative small models ($N = 6$). If comparing the times of completed proofs, the RB model exhibits a considerable gain (e.g., 3 seconds versus 36 hours in case of consistency, $N = 5$). The substantial growth of the state space of the composite model is due to the additional combinations stemming from all possible node schedules in the TTA, which have to be considered by the model checker.

Regarding (O2), Rushby already pointed out that the identification of the key fault-tolerant mechanisms of a specific round-based protocol might be tedious in a model which also contains the model of the implementing platform [9]. If deductive reasoning (e.g., theorem proving) is used for verification, the identification of such mechanisms is however crucial to complete the proof. Similarly, if the (possible complex) formal model of the implementing platform must be created, the design of the overall model might require an experienced user. Finally, the verification of the composite model possibly affects the automation of the verification. In particular, the model checking of the composite model with BMC required user interaction. The BMC approach is based on *k-induction* which uses a parameter (called depth) to prove the property. The depth can be usually approximated by the longest acyclic path in the model. In the RB model, the depth depends on the number of rounds, which is constant ($r = 2$). The default depth could be thus used. In case of TTA, on the other hand, the depth is also determined by the number of slots which varies based on N. The depth of k-induction had to be tuned accordingly.

6. Discussion and Conclusion

In this paper we analyzed the specification of a distributed diagnostic protocol with respect to different time-triggered systems as possible implementation platforms for the protocol. We used model checking to verify inconsistencies between the properties of the protocol's abstract specification and the properties of the implementation. We identified that the mismatch is because the implementation cannot directly guarantee the protocol's assumptions, such as communication, synchrony assumptions, etc.

As a solution, we proposed creating a layer which is able to provide the abstract assumptions on top of the selected implementation platform. The layer implements a transformation of the platform's characteristics and acts as a wrapper to convey the desired assumptions to the nodes running the distributed protocol. In particular, we defined message alignment to compensate the communication pattern of the TTA platform, and to simulate the lock-step assumption of the diagnostic protocol.

We showed that the previous concept can be generalized within a verification framework, which contains a distributed application (i.e., the protocol), the target implementation platform, and the transforming alignment layer in the middle of the architecture. In order to support an effective verification of a class of protocols, the properties of the alignment layer can be proven for all protocols in the class. In our case study, we proved that message alignment provides the assumptions of any round-based, synchronous protocols.

In future work, we aim at elaborating the application of a similar framework to verify the implementation of asynchronous protocols on partially synchronous systems.

References

[1] G. Bauer and M. Paulitsch. An Investigation of Membership and Clique Avoidance in TTP/C. In *Proc. of SRDS '00*, pages 118–124, 2000.

[2] L. de Moura *et al*. SAL 2. In *Proc. of CAV '04*, LNCS Vol. 3114 , pages 496–500, 2004.

[3] K. Driscoll *et al*. Byzantine Fault Tolerance, from Theory to Reality. In *Proc. of SAFECOMP '03*, pages 235–248, 2003.

[4] R. M. Kieckhafer *et al*. The MAFT Architecture for Distributed Fault Tolerance. *IEEE Trans. Comput.*, 37(4):398–405, 1988.

[5] H. Kopetz and G. Bauer. The Time-Triggered Architecture. *Proceedings of the IEEE*, 91(1):112 – 126, 2003.

[6] L. Lamport and S. Merz. Specifying and verifying fault-tolerant systems. In *Proc. of FTRTFT '94*, LNCS Vol. 863, pages 41–76, 1994.

[7] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

[8] H. Pfeifer, D. Schwier, and F. W. von Henke. Formal Verification for Time-Triggered Clock Synchronization. In *Proc. of DCCA—7*, pages 207–226, 1999.

[9] J. Rushby. Systematic Formal Verification for Fault-Tolerant Time-Triggered Algorithms. *IEEE Trans. Softw. Eng.*, 25(5):651–660, 1999.

[10] M. Serafini, N. Suri, J. Vinter, A. Ademaj, W. Brandstaetter, F. Tagliabó, and J. Koch. A Tunable Add-On Diagnostic Protocol for Time Triggered Systems. In *Proc. of DSN 2007 (To Appear)*.

[11] C. Walter, P. Lincoln, and N. Suri. Formally Verified On-Line Diagnosis. *IEEE Trans. Softw. Eng.*, 1997.